Use of Discrete Choice Models with Recommender Systems

by

Bassam H. Chaptini

Bachelor of Engineering, American University of Beirut (2000)
Master of Science in Information Technology (2002)

Submitted to the Department of Civil and Environmental Engineering in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

In the Field of

Information Technology

at the

Massachusetts Institute of Technology

June 2005

Signature of Author: [illegible]
Department of Civil and Environmental Engineering
March 30, 2005

Certified by: [illegible]
Steven R. Lerman
Class of 1922 Professor of Civil and Environmental Engineering
Thesis Supervisor

Accepted by [illegible]
Andrew Whittle
Chairman, Departmental Committee on Graduate Studies

Use of Discrete Choice Models with Recommender Systems

by

Bassam H. Chaptini

Submitted to the Department of Civil and Environmental Engineering
on March 30, 2005, in partial fulfillment of the Requirements for
the Degree of Doctor of Philosophy in the Field of Information Technology

ABSTRACT

Recommender systems, also known as personalization systems, are a popular technique for reducing information overload and finding items that are of interest to the user. Increasingly, people are turning to these systems to help them find the information that is most valuable to them. A variety of techniques have been proposed for performing recommendation, including content-based, collaborative, knowledge-based and other techniques. All of the known recommendation techniques have strengths and weaknesses, and many researchers have chosen to combine techniques in different ways.

In this dissertation, we investigate the use of discrete choice models as a radically new technique for giving personalized recommendations. Discrete choice modeling allows the integration of item and user specific data as well as contextual information that may be crucial in some applications. By giving a general multidimensional model that depends on a range of inputs, discrete choice subsumes other techniques used in the literature.

We present a software package that allows the adaptation of generalized discrete choice models to the recommendation task. Using a generalized framework that integrates recent advances and extensions of discrete choice allows the estimation of complex models that give a realistic representation of the behavior inherent in the choice process, and consequently a better understanding of behavior and improvements in predictions. Statistical learning, an important part of personalization, is realized using Bayesian procedures to update the model as more observations are collected.

As a test bed for investigating the effectiveness of this approach, we explore the application of discrete choice as a solution to the problem of recommending academic courses to students. The goal is to facilitate the course selection task by recommending subjects that would satisfy students' personal preferences and suit their abilities and interests. A generalized mixed logit model is used to analyze survey and course evaluation data. The resulting model identifies factors that make an academic subject "recommendable". It is used as the backbone for the recommender system application. The dissertation finally presents the software architecture of this system to highlight the software package's adaptability and extensibility to other applications.

Thesis Supervisor: Steven R. Lerman
Title: Professor of Civil and Environmental Engineering

Acknowledgments

I am indebted to a great number of people who generously offered advise, encouragement, inspiration and friendship through my time at MIT.

First of all, I wish to express my deepest gratitude to my advisor and mentor Professor Steven Lerman for his guidance, his support, for the opportunities he has provided me and for the invaluable insights he offered me.

Thank you to the members of my doctoral committee. Professor Moshe Ben-Akiva, who provided vital input and instrumental guidance for my research. Professor Dan Ariely, who provided insight and perspective on different aspects of my dissertation. Thanks are also extended to Dr. Joan Walker who generously offered her help and whose Ph.D. dissertation and feedback on my thesis proved to be invaluable for my research. I also would like to thank all the RA's and staff at the Center for Educational Computing Initiatives for making CECI such a wonderful place to work.

Thank you to all my friends. Rachelle for her kindness, her wisdom, her continuous encouragement, her interest in my endeavors, and for providing great escapes from MIT and making the process a whole lot more enjoyable. Karim for his friendship and the conversations about statistics and, thankfully, other topics as well. Jad and Ziad for their friendship, the laughter, the stories and the wonderful conversations and inspiration. Fouad and Georges for their ever lasting friendship. Georges for being the most wonderful listener and for his closeness despite him being in France. Fouad for his sincerity, his perspective, and his endless support. For all other friends for making my stay at MIT a memorable experience.

In countless ways I have received support and love from my family. I would like to take this opportunity to thank them for all the love, affection, encouragement and wonderful moments they shared with me over the years. To my cousin Cyril, for always being there for me. To my siblings, Nayla and Zeina, for providing me support and entertainment. To my mom, for her endless love and caring whom I hope I have given back a fraction of what I have received. To my dad who has been influencing and supporting my academic pursuits for as long as I can remember, and who has a tremendous influence on who I am.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOTATION

n	denotes an individual, $n = 1 ,\dots, N$.
N	total number of individuals
i,j	denote alternatives, $i, j = 1 ,\dots, J_n$.
J_n	is the number of alternatives in the choice set C_n.
C_n	is the choice set faced by individual n.
t	indexes a response across observations of a given respondent n, where $t=1,\dots,T$.
y_{in}, y_{int}	is a choice indicator (equals to 1 if alternative i is chosen, and 0 otherwise).
U_n	is the utility as perceived by individual n.
U_{in}	is a the utility of alternative i as perceived by individual n.
U_{nt}	is the utility as perceived by individual n for observation t.
X_n	is a $(1 \times K)$ vector of explanatory variables describing n.
X_{in}	is a $(1 \times K)$ vector of explanatory variables describing n and i.
X_{nt}	is a $(1 \times K)$ vector of explanatory variables describing n for observation t.
β	is a $(K \times 1)$ vector of unknown parameters.
β_n	is a $(K \times 1)$ vector of unknown parameters for person n.
Σ_β	covariance matrix of vector β_n.
θ	represent the mean and covariance matrix Σ_β of the normally distributed β_n.
$\varepsilon_n, \varepsilon_{in}, \varepsilon_{nt}$	are i.i.d. Gumbel random variables.
μ	scale parameter of $\varepsilon_n, \varepsilon_{in}, \varepsilon_{\text{int}}$.
ν_n	is a panel data random disturbances; $\nu_n \sim N(0, \sigma_\nu^2)$.
$\tilde{\nu}_n$	is a standard normal disturbance $\tilde{\nu}_n \sim N(0,1)$
Σ, σ	covariances of random disturbance terms.

RP and SP Notation

r	indexes an RP response across RP observations of a given respondent *n*, where *r=1,...,R*.
R	is the total number of RP observations given by respondent *n*
s	indexes an SP response across SP observations of a given respondent *n*, where *s=1,...,S*.
S	is the total number of SP observations given by respondent *n*
X_{nt}	is the $(1 \times K)$ vector of attributes and characteristics present in both RP and SP setting,
W_{nt}	is the $(1 \times K^{RP})$ matrix of variables present only in the RP setting.
δ	is a $(K^{RP} \times 1)$ vector of unknown parameters *n*.
v_n^{RP}	is a panel data random effects for the RP situation; $v_n^{RP} \sim N(0, \sigma_v^{RP^2})$
ε_{nt}^{RP}	is an i.i.d. Gumbel random variable with scale parameter μ^{RP}
Z_{nt}	is the $(1 \times K^{SP})$ matrix of variables present only in the SP setting,
χ	is a $(K^{SP} \times 1)$ vector of unknown parameters *n*.
τ^{SP}	is a constant present only in the SP setting,
v_n^{SP}	is a panel data random effects for the RP situation; $v_n^{SP} \sim N(0, \sigma_v^{SP^2})$
ε_{nt}^{SP}	is an i.i.d. Gumbel random variable with scale parameter μ^{SP}

Latent Variable Notation

l	denotes the index of a latent variable, *l=1,...,L*
L	is the total number of latent variables.
$X_{\ln t}^*$	is the latent variable *l* describing latent characteristics of individual *n* for observation *t*
$X_{\ln t}$	is a $(1 \times K_l)$ vector of explanatory variables describing *n* for observation *t* and latent variable *l*,
X_{nt}^*	is a $(1 \times L)$ vector of stacked $X_{\ln t}^*$.
λ_l	is a $(K_l \times 1)$ vector of unknown parameters.

ω_{lnt}	is the disturbance term for latent variable *l*.
ω_{nt}	is a multivariate normal distribution with mean 0 and covariance matrix Σ_ω
ζ_{nt}	is an $(L \times 1)$ vector of standard independent normally distributed variables.
F_l	is the 1th row of an $(L \times L)$ identity matrix.
Γ	is the $(L \times L)$ lower triangular Cholesky matrix such that $\Gamma\Gamma' = \Sigma_\omega$
γ_l	is an unknown parameter acting as the loading factor for X^*_{lnt}.
γ	is an $(L \times 1)$ vector of stacked γ_l.
m	denotes the index of the indicator *m=1,2,...,M*,
I_{mnt}	is an indicator of X^*_{nt}
α_m	is an $(L \times 1)$ vector of coefficient to be estimated.
ϕ	is the standard normal density function.

Chapter 1: Introduction

This dissertation examines the use of discrete choice models as a core element of recommender systems. The framework set is generic and will be used in the context of academic advising by investigating the application of discrete choice models as a solution to the problem of recommending academic courses to students.
This introductory chapter will serve to define two keywords – recommender systems and discrete choice models – by giving an overview of recommender systems, what they are, and how they are used, and by briefly describing what constitutes discrete choice models. Both areas will be covered in detail in separate chapters with a description of the different technologies, frameworks and theories used. The chapter will also serve to set a general framework that identifies the steps needed to develop a recommender system using discrete choice theory. Throughout this dissertation, we will examine this framework under the test bed application of academic advising that this chapter introduces and defines.

1.1. Research Motivation

Typically, a recommender system works by asking you a series of questions about things you liked or didn't like. It compares your answers to those of others, and finds people who have similar opinions. Chances are if they liked an item, you would enjoy it too. This technique in providing item recommendations or predictions based on the opinions of like-minded users is called "Collaborative Filtering" and is the most successful recommendation technique to date. Another technique, called content-based filtering, searches over a corpus of items based on a query identifying intrinsic features of the items sought. In content-based filtering, one tries to recommend items similar to those a given user has liked in the past. The assumption in collaborative and content-based filtering is that items are going to be recommended based on similarities among the users or items themselves. These approaches suffer from a lack of theoretical understanding of the behavioral process that led to a particular choice.

Discrete choice models are based on behavioral theory and are rooted in classic economic theory, which states that consumers are rational decision makers who, when faced with a set of possible consumption bundle of goods, assign preferences to each of the various bundles and then choose the most preferred bundle from the set of affordable alternatives. Discrete choice models have proven successful in many different areas, including transportation, energy, housing and marketing – to name only a few. They are still subject to continuous research to extend and enrich their capabilities. Although the literature is very extensive, they have been typically used to predict choices on an aggregate level. Recent efforts in statistical marketing research, however, have focused on using choice models to predict individual choices that can provide a critical foundation of market segmentation and as input to market simulators. This research, on the other hand, is interested in investigating the use of discrete choice to predict choices on an individual level to offer personalized recommendations.

To test the framework of using discrete choice modeling with recommender systems, we tackled the problem of designing and developing an online academic advisor that recommends students academic courses they would enjoy. This application is innovative as no other system that we know of was developed to tackle this problem. The application also makes use of valuable course evaluation data that is available online for students, but is not efficiently used.

1.2. What are Recommender Systems?

Recommender Systems, also known as personalization systems, are a popular technique for reducing information overload and finding items that are of interest to the user. Increasingly, people are turning to these systems to help them find the information that is most valuable to them. The process involves gathering user-information during interaction with the user, which is then used to deliver appropriate content and services, tailor-made to the user's needs. The aim is to improve the user's experience of a service. Recommender systems support a broad range of applications, including recommending movies, books, music, and relevant search results. They are an ever-growing feature of online services that is manifested in different ways and contexts, harnessing a series of developing technologies. They are of particular interest for the e-business industry where the purposes to provide personalization are to[1]:

- Better serve the customer by anticipating needs
- Make the interaction efficient and satisfying for both parties
- Build a relationship that encourages the customer to return for subsequent purchases

User satisfaction is the ultimate aim of a recommender system. Beyond the common goal, however, there is a great diversity in how personalization can be achieved. Information about the user can be obtained from a history of previous sessions, or through interaction in real time. "Needs" may be those stated by the customer as well as those perceived by the business. Once the user's needs are established, rules and techniques, such as "collaborative filtering", are used to decide what content might be appropriate.

A distinction is often made between customization and personalization. Customization occurs when the user can configure an interface and creates a profile manually, adding and removing elements in the profile. The control of the look and/or content is user-driven. In personalization, on the other hand, the user is seen as being less in control. It is the recommender system that monitors, analyses and reacts to behavior (e.g. content offered can be based on tracking surfing decision).

The following are two examples that show the different kinds of personalized services encountered on the web. Amazon.com provides an example of how personalized recommendations are employed as a marketing tool. MyBestBets is a specific application that gives recommendations on what to watch on TV.

[1] Reference: The Personalization Consortium (http://www.personalization.org/)

1.2.1. Amazon.com

Amongst its other features, Amazon.com[2] will make suggestions for products that should be of interest to the customer whilst he/she is browsing the site. Amazon.com determines a user's interest from previous purchases as well as ratings given to titles. The user's interests are compared with those of other customers to generate titles which are then recommended during the web session. Recommendations for books that are already owned by the customer can be removed from the recommendation list if the customer rates the title. Removing titles from recommendations list by giving ratings helps to generate new recommendations.

1.2.2. MyBestBets

The MyBestBets[3] personalization platform, powered by ChoiceStream[4], provides personalization capabilities that make it easier for consumers to navigate vast content spaces to find those choices that they'll really enjoy. The Platform's recommendations understand not just what people like, but "why" they like it. By knowing how consumers think about content, the recommender system matches each individual consumer's needs and interests with the content they are most likely to enjoy. For example, when personalizing movie recommendations, MyBestBets determines not just that a particular user likes "romantic comedies", but that he/she likes thoughtful, modern, romantic comedies that are slightly edgy. Using this insight regarding the underlying attributes that appeal to a user, ChoiceStream identifies movies with similar attributes, matching the user's interests.

The ability to deliver the experiences described above rests on the acquisition of a profile of the user. The user has attributes, interests, needs – some or all of which need to be captures and processed. The techniques used to complete the profiling of the user are varied. Furthermore, there are differences in how the appropriate content that matches the user's needs is determined and delivered. In Chapter 2 of this dissertation, we will

[2] http://www.amazon.com
[3] http://www.mybestbets.com
[4] http://choicestream.com

explain some of the technologies in use, describing some of their advantages and disadvantages.

1.3. What are Discrete Choice Models?

The standard tool for modeling individual choice behavior is the discrete choice model based on random utility hypothesis. According to [Ben-Akiva and Lerman, 1985], these models are based on behavioral theory that is: (i) descriptive, in the sense that it postulate how human beings behave and does not prescribe how they ought to behave, (ii) abstract, in the sense that it can be formalized in terms which are not specific to a particular circumstance, (iii) operational, in the sense that it results in models with parameters and variables that can be measured. Formally stated, a specific theory of choice is a collection of procedures that defines the following elements [Ben-Akiva and Lerman, 1985]:

1. *Decision maker*: this can be an individual person or a group of persons. Individuals face different choices and have widely different tastes. Therefore, the differences in decision-making processes among individuals must be explicitly treated.

2. *Alternatives*: A choice is by definition made from a non-empty set of alternatives. Alternatives must be feasible to the decision maker and known during the decision process.

3. *Attribute of Alternatives:* The attractiveness of an alternative is evaluated in terms of a vector of attributes. The attribute values are measured on a scale of attractiveness.

4. *Decision rule:* The mechanisms used by the decision maker to process the information available and arrive at a unique choice. Random utility theory is one of a variety of decision rules that have been proposed and is the most used discrete choice model.

Utility assumes that the attractiveness of an alternative is measured by the combination of a vector of attributes values. Hence utility is reducible to a scalar expressing the attraction of an alternative in terms of its attributes. A utility function associates a real number with each possible alternative such that it summarizes the preference ordering of the decision maker. The concept of random utility introduces the concept that a modeler's inferences about individual choice behavior are probabilistic. The individual is still assumed to select the alternative with the highest utility. However the analyst does not know the utilities with certainty. Therefore, utility is modeled as a random variable, consisting of an observed measurable component and an unobserved random component. Chapter 3 of this dissertation reviews random utility concepts and basic discrete choice theory, while Chapter 5 investigates extensions of the basic model under the test bed of the online academic advisor.

1.4. General Framework for Designing and Implementing Recommender Systems using Discrete Choice Theory

The first step of constructing a recommender system is to understand the problem at hand and its scope, similar to what was described in the previous section. Namely, the components of the problem under the collection of procedures defined by the choice theory needs to be thoroughly understood (decision maker, alternatives, attributes, and decision rule). Once the problem has been defined, the rest of the framework includes data collection, statistical modeling, and software implementation. Formally stated, a general framework for developing recommender systems using discrete choice includes the following steps:

1. Defining the problem under the specific theory of choice formed by the collection of procedures that defines the decision maker, the alternatives, the attributes of the alternative and finally the decision rule.

2. Designing surveys and collecting data to better understand the factors involved in the choice process and constructing a database of reliable data that would lead to robust models. For optimal results, this step should heavily rely on the use of experimental design techniques to construct surveys that draw on the advantages of using different types of data thereby reducing bias and improving efficiency of the model estimate.

3. Constructing and estimating choice models that best fit the collected data and that are behaviorally realistic to the problem at hand. The basic technique for integrating complex statistical models is to start with the formulation of a basic discrete choice model, and then add extensions that relax simplifying assumptions and enrich the capabilities of the basic model.

4. Incorporating the choice model as part of a software package that hosts the recommender system and whose function is to incorporate the estimated model, collect more data and observations, construct user profiles, personalize the estimated model to fit those profiles, and finally provide personalized recommendations.

The framework given by these four steps is generic to any recommendation system and will be used to both construct the "Online Academic Advisor" and to structure this dissertation. The next section will tackle the first step of this framework which is to introduce and define the problem of academic advising. Later chapters will deal with the remaining steps.

1.5. Test Bed: Online Academic Advisor

Choosing the appropriate classes is a crucial task that students have to face at the start of every semester. Students are flooded with an extensive list of course offerings, and when presented with a number of unfamiliar alternatives, they tend to seek out recommendations that often come either from their advisors or fellow students.

The goal of an academic advising application is to facilitate the class selection task by recommending subjects that would satisfy the students' personal preferences and suit their abilities and interests. Accomplishing the complex task of advising should include assisting students in choosing which courses to take together and when to take them, what electives to choose, and how to satisfy departmental requirements. Most of the complexity of this problem arises from the fact that the recommender system not only needs to consider a set of courses for the next semester, but also needs to have an extended list of courses that leads to graduation.

1.5.1. Overview of the Choice Problem

The aim of this research is not fully automate the complex problem of advising, but rather to have a software agent that assists students in assessing how "enjoyable" a class would be for them to take and hence help them decide which term to take a required subject and which elective to take. In other words, this dissertation explores the application of discrete choice to help a given student find the classes he/she is interested in by producing a predicted likeliness score for a class or a list of top *N* recommended classes. The software agent will only focus on predicting what courses are recommended the most in the upcoming semester, with a minimal effort spent on studying the interaction of courses with each other, and on satisfying departmental requirements.

In order to achieve this objective, the factors that influence students' overall impression of a class need to be understood. The hypothesis is that students tend to form an overall impression of a class based on factors or attributes such an individual students' characteristics (e.g. area of concentration, gender or year towards graduation), the class content, the class character (e.g. enrolment size, lab), logistics (e.g. schedule, location), and effectiveness of the instructors only to name a few. Once those attributes are defined, discrete choice models can be used to estimate the overall utility or "how recommendable" a class is.

1.5.2. The Decision Maker

The decision maker in the context of this research is an undergraduate student (sophomore, junior, senior or M.Eng.[5]) in the department of Electrical Engineering and Computer Science (EECS) at MIT. The EECS department was chosen because it has the largest enrollment at MIT and thus provides a good sample population for the studies that need to be conducted.

Students within the EECS department can seek one of three different undergraduate degrees: Bachelor of Science in Electrical Engineering (BS in EE), Bachelor of Science in Electrical Engineering and Computer Science (BS in EECS), and Bachelor of Science

[5] The EECS Master of Engineering program is a five-year program available only to M.I.T. EECS undergraduates. It is an integrated undergraduate/graduate professional degree program with subject requirements ensuring breadth and depth. Students write a single 24-unit thesis, which is to be completed in no more than three terms.

in Computer Science (BS in CS). Students are also generally affiliated with one of the 7 areas (also known as concentrations) that the department offers (Communication, Control, and Signal Processing; Artificial Intelligence and Applications; Devices, Circuits, and Systems; Computer Systems and Architecture Engineering; Electrodynamics and Energy Systems; Theoretical Computer Science; Bioelectrical Engineering). Having this kind of student segmentation offers a good test bed to study the effect of heterogeneity and clustering in applying choice models for recommender systems.

1.5.3. The Alternatives

This research will only consider the courses that are offered in the EECS department. Any student considers a subset of this set, termed their choice set. This latter includes courses that are feasible during the decision process. Although EECS course offerings includes dozens of potential classes, the choice set for a particular student is usually considerably reduced because of constraints such as whether the class is being offered in any given semester, scheduling, prerequisites and academic requirements for graduation. As it was previously mentioned, the goal of this research is to predict a level of enjoyment for a given subject and present the student with a list of the most enjoyable classes for the next semester. Under these circumstances, the "choice problem" becomes in reality a "rating problem" where the choice set is simply the scale or "level of recommendation" for a given class. In presenting the list of classes, the recommender system will not take into consideration academic requirements which are department specific. On the other hand, it will account for class offering and prerequisites.

1.5.4. Evaluation of Attributes of the Alternatives

As was stated previously, one of the main hypotheses is that an academic course can be represented by a set of attributes that would define its attractiveness to a particular student. Courses are considered to be heterogeneous alternatives where decision makers may have different choice sets, evaluate different attributes, and assign diverse values for the same attribute of the same alternative. As a consequence, we need to work with a general characterization of each course by its attributes. A significant part of this research will focus on identifying the factors that influence students' choices in selecting their academic courses, and evaluate their relative importance in the choice process.

1.5.5. The Decision Rule

The generalized discrete choice framework is a flexible, tractable, theoretically grounded, empirically verifiable, and intuitive set of methods for incorporating and integrating complex behavioral processes in the choice model. It obviates the limitations of standard models by allowing for random taste variations and correlations in unobserved factors over time. For these reasons, generalized discrete choice models will be used as our decision rule to model student rating behavior. A thorough review of the basic choice models is included in Chapter 3, and extensions of their basic functionality in Chapter 5.

1.6. Dissertation Structure

This dissertation will be structured following the framework defined in section 1.4, which will take the reader through the steps of designing and constructing a recommender system for the specific task of academic advising. Presenting and explaining the framework under a specific problem will serve two purposes: help understand the different steps involved while stressing on implementation, thus showing how the framework can be replicated to any other application; prove the applicability of discrete choice with recommender systems by actually developing a working prototype.

Now that the first step of the framework applied to academic advising has been tackled in section 1.5, the remainder of this dissertation is organized as follows. Chapter 2 and 3 will serve as a review of recommender systems and discrete choice literature. Chapter 2 presents an overview of the latest advances in recommender systems. Chapter 3 presents a theoretical background of basic discrete choice models. Chapter 4, being the second step of the framework, is dedicated to the studies, surveys and data collection to understand and model the problem of creating an automated academic advisor. Chapter 5 is the third step and focuses on the student rating model by first constructing a modeling framework that includes advanced methods in choice modeling. It then describes the estimation procedure and finally presents the results. Chapter 6 is the final step and focuses on the architectural design of the online academic advisor by describing the functionality of the different modules of the software package. It also provides performance metrics of the resulting recommender system. Chapter 7 provides a summary and directions for further research.

Chapter 2: Recommender Systems

Recommender systems (also known as personalization system) apply data analysis techniques to the problem of helping users find the items they are interested in by producing a predicted likeliness score or a list of top-*N* recommended items for a given user. Item recommendations can be made using different methods. Recommendations can be based on demographics of the users, overall top chosen items, or past choice habits of users as a predictor of future items. Collaborative Filtering (CF) is the most successful recommendation technique to date ([Ungar and Foster, 1998] , [Shardanand and Maes, 1995]). Typically, these systems do not use any information regarding the actual content of the items, but are rather based on usage or preference patterns of other users. CF is built on the assumption that a good way to find interesting content is to find other people who have similar interests, and then recommend items that those similar users like. In fact, most people are familiar with the most basic form of CF: "word of mouth". For instance, it is a form of CF when someone consults with friends to gather opinions about a new restaurant before reserving a table. In the context of recommender systems, CF takes this common way people gather information to inform their decisions to the next level by allowing computers to help each of us be filters to someone else, even for people that we don't know ([Miller et al, 2004]).

The opinions of users can be obtained explicitly from the users or by using some implicit measures. Explicit voting refers to a user consciously expressing his or her preference for an item, usually on a discrete numerical scale. Implicit voting refers to interpreting user behavior or selections to input a vote or preference. Implicit votes can be based on browsing data (e.g. Web applications), purchase history (e.g. online or traditional stores), or other types of information access patterns. The computer's role is then to predict the rating a user would give for an item that he or she has not yet seen.

2.1. Goals and Components of a Recommender System

The overall goals of a recommender system can be summarized as following:

- It must deliver relevant, precise recommendations based on each individual's tastes and preferences.
- It must determine these preferences with minimal involvement from consumers.
- And it must deliver recommendations in real time, enabling consumers to act on them immediately.

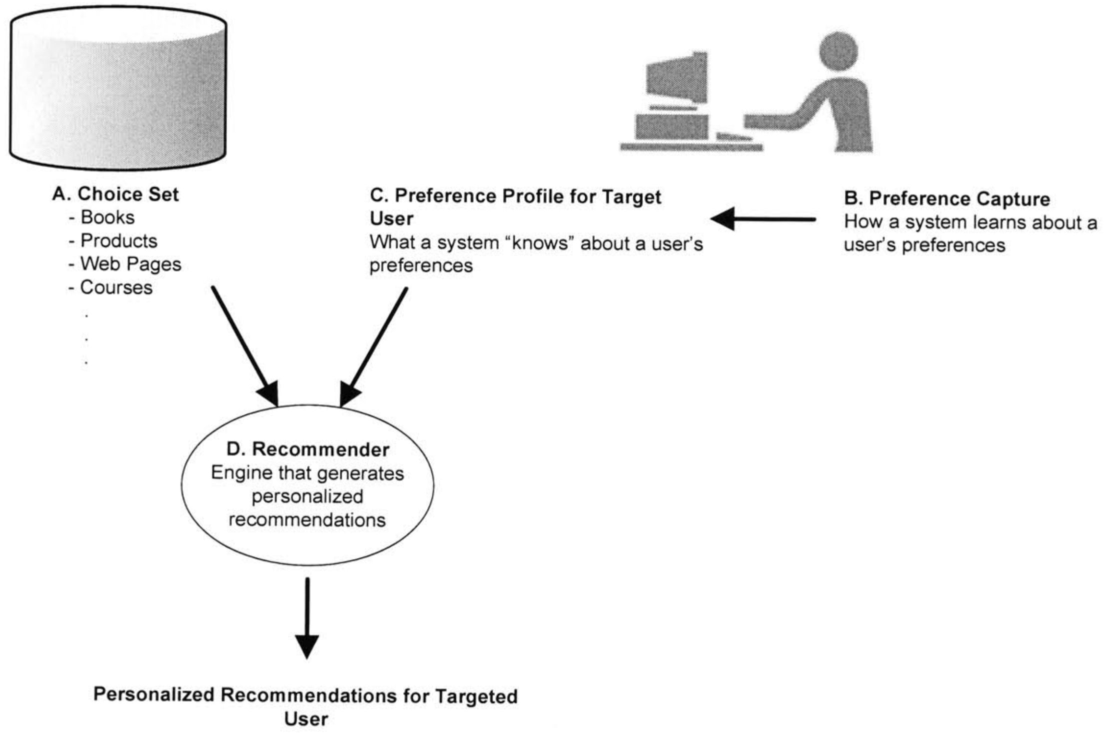

A. Choice Set
- Books
- Products
- Web Pages
- Courses
C. Preference Profile for Target User
What a system "knows" about a user's preferences
B. Preference Capture
How a system learns about a user's preferences
D. Recommender
Engine that generates personalized recommendations
Personalized Recommendations for Targeted User

Figure 2-1. Components of a Recommender System ([ChoiceStream])

Technologies designed to meet those goals vary widely in terms of their specific implementation. The ChoiceStream Technology Brief[6] defines the core of recommender systems as being made up of the following component (see Figure 2-1):

[6] http://www.choicestream.com/pdf/ChoiceStream_TechBrief.pdf

- Choice Set. The choice set represents the universe of content, products, media, etc. that are available to be recommended to users.
- Preference Capture. User preferences for content can be captured in a number of ways. Users can rate content, indicating their level of interest in products or content that are recommended to them. Users can fill out questionnaires, providing general preference information that can be analyzed and applied across a content domain(s). And, where privacy policies allow, a personalization system can observe a user's choices and/or purchases and infer preferences from those choices.
- Preference Profile. The user preference profile contains all the information that a personalization system 'knows' about a user. The profile can be as simple as a list of choices, or ratings, made by each user. A more sophisticated profile might provide a summary of each user's tastes and preferences for various attributes of the content in the choice set.
- Recommender. The recommender algorithm uses the information regarding the items in a choice set and a user's preferences for those items to generate personalized recommendations. The quality of recommendations depends on how accurately the system captures a user's preferences as well as its ability to accurately match those preferences with content in the choice set.

We now turn our focus on the last part of the recommender system's components: The recommender algorithm. As was stated earlier, collaborative filtering (CF) is the most popular technique in use. There are two general classes of CF algorithms. User-Based algorithms operate over the entire user database to make predictions. Model-based collaborative filtering, in contrast, uses the user database to estimate or train a model, which is then used for predictions [Balabanovic and Shoham, 1997].

2.2. User-Based Collaborative Filtering

User-Based algorithms utilize the entire user-item database to generate a prediction. These systems employ statistical techniques to find a set of users, known as *neighbors*, that have a history of agreeing with the target user (i.e., they either rate different items similarly or they tend to buy similar sets of items). Once a neighborhood of users is formed, these systems use different algorithms to combine the preferences of neighbors to produce a prediction or top-N recommendation for the active user. The techniques, also known as nearest-neighbor or user-based collaborative filtering, are widely used in practice. The basic user-based collaborative filtering algorithm, described in [Resnick et al., 1994], can be divided into roughly three main phases: neighborhood formation, pairwise prediction, and prediction aggregation. As an example, Figure 2-2 shows six person shapes representing six users. In particular we are interested in calculating predictions for user "A". The distance between each person indicates how similar each user is to "A". The closer the persons on the figure the more similar the users are. In neighborhood formation, the technique is to select the right subset of users who are most similar to "A". Once the algorithm has selected which neighborhood, represented in Figure 2-2 by the users in the circle, it can make an estimate of how much "A" will value a particular item. In pairwise prediction, the algorithm learns how much each user in the neighborhood rated a particular item. The final step - prediction aggregation - is to do a weighted average of all the ratings to come up with a final prediction (see [Miller et al, 2004] for a more detailed description of his example, particularly on how it is internally represented in a CF system).

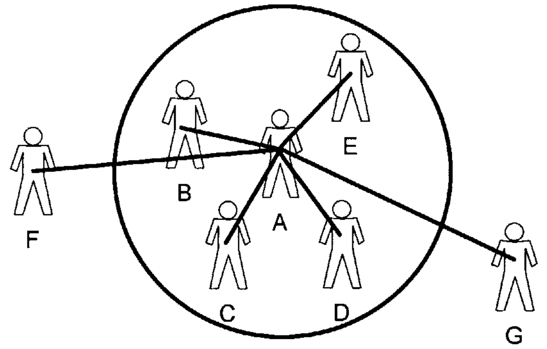


Figure 2-2. Basic Collaborative Filtering Algorithm [Miller et al, 2004]

2.2.1. Challenges and Limitations

The CF systems described above has been very successful in the past, but its widespread use has revealed some real challenges [Claypool et al., 1999]:

- Early rater problem: Pure CF cannot provide a prediction for an item when it first appears since there are no users ratings on which to base the predictions. Moreover, early predictions for the item will often be inaccurate because there are few ratings on which to base the predictions. Similarly, even an established system will provide poor predictions for each and every new user that enters the systems. As extreme case of the early rater problem, when a CF system first begins, every user suffers from the early rater problem for every item.
- Scarcity problem: In many information domains, the number of item far exceeds what any individual can hope to absorb, thus matrices containing the ratings of all items for all users are very sparse. Relatively dense information filtering domains will often still be 98-99% sparse, making it hard to find items that have been rated by enough people on which to base collaborative filtering predictions.
- Gray Sheep: In a small or even medium community of users, there are individuals who would not benefit from pure collaborative filtering systems because their opinions do not consistently agree or disagree with any group of people. These individuals will rarely, if ever, receive accurate collaborative filtering predictions, even after the initial start up phase for the user and the system.
- Scalability: As the number of users and items grows, the process of finding neighbors becomes very time consuming. The computation load is approximately linear with the number of users making it difficult for website with high volumes and large user base to do a lot of personalization.

2.3. Model-based Collaborative Filtering Algorithms

Model-based collaborative filtering algorithms provide item recommendations by first developing a model of user ratings. Algorithms in this category take a probabilistic approach and represent the collaborative filtering process as computing the expected value of a user prediction, given his or her ratings of other items. The model building process is performed by different machine learning algorithms such as Bayesian networks, clustering, and rule-based approaches. The Bayesian network model [Breese et al, 1998] formulates a probabilistic model for collaborative filtering problem. The clustering model treats collaborative filtering as a classification problem ([Basu et al, 1998], [Breese et al, 1998], [Ungar and Foster, 1998]) and works by clustering similar users in a same class and estimating the probability that a particular user is in a particular class, and from there computes the conditional probability of ratings. The rule-based approach applies association rule discovery algorithms to find association between co-purchased items. Essentially these rule discovery algorithms work by discovering association between two sets of products such that the presence of some products in a particular transaction implies that products from the other set are also present in the same transaction. The system then generates item recommendation based on the strength of the association between items [Sarwar et al, 2000]. All the above mentioned approaches have one thing in common; each approach separates the collaborative filtering computation into two parts. In the first part, which can be done offline, a model is build that captures the relationship between users and items. The second part, typically done in real time during a web session, uses the model to compute a recommendation. Most of the work is generally done in building the model making the recommendation computation very fast.

2.3.1. Item-Based Collaborative Filtering

Item-based CF is one example of a model-based approach. It is based on the observation that the purchase of one item will often lead to the purchase of another item (see [Aggarwal et al., 1999], [Billsus and Pazani, 1998], and [Breese et al., 1999]). To capture this phenomenon, a model is build that capture the relationship between items ([Karypis, 2001] and [Sarwar et al., 2001] call this approach *item-item CF*).

Item-based CF were developed to create personalization systems with lower computation costs than those relying on user-based CF. And while item-based systems are generally more scalable than user-based ones, the two approaches to CF share many of the same deficiencies, including poor or inconsistent recommendation quality and the inability to recommend new or changing content (i.e. the cold start problem). Like user-based CF systems, item-based CF solutions recognize patterns. However, instead of identifying patterns of similarity between users' choices, this technique identifies patterns of similarity between the items themselves.

A very simple item-based approach can be built by simply counting the number of times that a pair of products is purchased by the same user ([Miller et al., 2004]). In general terms, item-based CF looks at each item on a target user's list of chosen/rated items and finds other content in the choice set that it deems similar to that item. Determination of similarity can be made by scoring items based on explicit content attributes (e.g. movie genre, lead actors, director, etc.) or by calculating correlations of user ratings between items.

Reduced to a simple formula, item-based CF says that if the target user likes A, the system will recommend items B and C if those items are determined to be the most similar to item A based on their correlations or attributes. The main advantage of an item-based system over a user-based one is its scalability. Item-based solutions do not have to scour databases containing potentially millions of users in real time in order to find users with similar tastes. Instead, they can pre-score content based on user ratings and/or their attributes and then make recommendations without incurring high computation costs.

2.4. Content-based Filtering

A number of authors and system designers have experimented with enhancing CF with content-based extensions [Balabanovic and Shoham, 1997].

Content-based search over a corpus of items is based on a query identifying intrinsic features of the items sought. Search for textual documents (e.g. Web pages) uses queries containing words or describing concepts that are desired in the returned documents. Search for titles of compact discs, for example, might require identification of desired artist, genre, or time period. Most content retrieval methodologies use some type of similarity score to match a query describing the content with the individual titles or items, and then present the user with a ranked list of suggestions [Breese et al, 1998].

So in *content-based* recommendation one tries to recommend items similar to those a given user has liked in the past, whereas in *collaborative* recommendation one identifies users whose tastes are similar to those of the given user and recommends items they have liked. A pure content-based recommendation system is considered to be one in which recommendations are made for a user based solely on a profile built up by analyzing the content of items which that user has rated in the past. A pure content-based system has

several shortcomings. Generally, content-based techniques have difficulty in distinguishing between high-quality and low-quality information that is on the same topic. And as the number of items grows, the number of items in the same content-based category increases, further decreasing the effectiveness of content-based approaches.
A second problem, which has been studied extensively both in this domain and in others, is over-specialization. When the system can only recommend items scoring highly against a user's profile, the user is restricted to seeing items similar to those already rated. Often this is addressed by injecting a degree of randomness into the scoring of similarity [Balabanovic and Shoham, 1997].

2.5. Hybrid Systems: Content-based and Collaborative Filtering

Experiments have shown collaborative filtering systems can be enhanced by adding content based filters ([Alspector et al, 1998], [Balabanovic et al, 1997], [Claypool et al, 1999]). In one approach to create a hybrid content-based, collaborative system [Claypool et al, 1999], user profiles are maintained based on content analysis, and directly compare these profiles to determine similar users for collaborative recommendation. Users receive items both when they score highly against their own profile and when items are rated highly by a user with a similar profile.
Another approach to building hybrid recommender systems is to implement separate collaborative and content-based recommender systems. Then two different scenarios are possible. First, the outputs (ratings) obtained from individual recommender systems are combined into one final recommendation using either a linear combination of ratings ([Claypool et al., 1999]) or voting scheme ([Pazzani, 1999]). Alternatively, one of the individual recommender systems can be used at any given moment, choosing to use the one that is "better" than others based on some recommendation "quality" metric ([Billsus and Pazzani, 2000] and [Tran and Cohen, 2000]).
Hybrid recommendation systems can also be augmented by knowledge-based techniques ([Burke 2000]), such as case-based reasoning, in order to improve recommendation accuracy and to address some of the limitations of traditional recommender systems.

2.6. Extending Capabilities of Recommender Systems

The current generation of recommendation technologies performed well in several applications, including the ones for recommending books, CDs, and new articles ([Mooney, 1999] and [Schafer et al., 2001]). However, these methods need to be extended for more complex types of applications. For example, [Adomavicius et al., 2003] showed that the multidimensional approach to recommending movies outperformed simple collaborative filtering by taking into the consideration additional information, such as when the movie is seen, with whom, and where.

By using discrete choice models, this research uses a multidimensional approach to model individual choice behavior. The approach is not specifically intended to overcome the weaknesses of CF and content-based filtering, but is aimed at investigating and adapting a radically new model for recommender systems. As was pointed out earlier, most of the recommendation methods produce ratings that are based on a limited understanding of users and items as captured by user and item profiles and do not take full advantage of the available data. Discrete choice modeling bridge this gap by fully using and integrating in one model item and user specific data as well as contextual information, such as time and place that may be crucial in some applications. By giving a general model that depends on a whole range of inputs, discrete choice models subsumes collaborative, content-based and hybrid methods discussed in the previous sections.

The next chapter will provide an overview of the basic theory and mathematics underlying discrete choice models and will present the recent advances in discrete choice models and their potential adaptation to recommender systems.

Chapter 3: Discrete Choice Models

3.1. Random Utility Models

Classical consumer theory assumes deterministic behavior, which states that utility of alternatives is known with certainty, and that the individual is always assumed to select the alternative with the highest utility. However, these assumptions have significant limitations for practical applications. Indeed, the complexity of human behavior suggests that a choice model should explicitly capture some level of uncertainty. The classical consumer theory fails to do so.

The concept of random utility introduces the concept that individual choice behavior is probabilistic. The individual is still assumed to select the alternative with the highest utility. However the analyst does not know the utilities with certainty. Therefore, utility is modeled as a random variable, consisting of an observed measurable component and an unobserved random component.

3.1.1. Deterministic and Random Utility components

The utility that individual *n* is associating with alternative *i* is given by:

[3-1] $U_{in} = V_{in} + \varepsilon_{in}$

Where

V_{in} is the deterministic part and

ε_{in} is the stochastic part (or disturbance)

There are four sources of uncertainty: unobserved alternative attributes, unobserved individual attributes (or taste variations), measurement errors, and proxy (or instrumental) variables [Ben-Akiva and Lerman, 1985].

The alternative with the highest utility is supposed to be chosen. Therefore, the probability that alternative *i* is chosen by decision-maker *n* within choice set C_n is:

[3-2] $P_C^n(i) = P[U_{in} \geq U_{jn}, \forall j \in C_n]$

Using Equation [3-1], Equation [3-2] can be rewritten as:

$$P_C^n(i) = P[V_{in} + \varepsilon_{in} \geq V_{jn} + \varepsilon_{jn}, \forall j \in C_n]$$

[3-3] $P_C^n(i) = P[V_{in} - V_{jn} \geq \varepsilon_{jn} - \varepsilon_{in}, \forall j \in C_n]$

Note that the utility is an arbitrarily defined scale. Thus adding a constant to all utilities does not affect the choice probablities even though it shifts the functions V_{in} and V_{jn}.

The derivation of random utility models is based on a specification of the utility as defined by Equation [3-1]. Different assumptions about the random term ε_{in} and the deterministic term V_{in} will produce specific models.

3.2. Specification of the Deterministic Part

The utility of each alternative must be a function of the attributes of the alternative itself and of the decision-maker. We can write the deterministic part of the utility that individual *n* is associating with alternative *i* as:

$$V_{in} = V_{in}(x_{in})$$

where x_{in} is a an attribute either of individual *n* or attribute *i*. The function defined is often assumed linear, that is if *K* attributes are considered:

[3-4] $V_{in}(x_{in}) = \sum_{k=1}^{K} \beta_k x_{in}$

where β_k are parameters to be estimated. Linearity in parameters is not equivalent to linearity in the attributes since the x_{in} s can themselves be functions of attributes.

3.3. Specification of the Disturbance

We can discuss the specification of the choice model by only considering the difference $\varepsilon_{jn} - \varepsilon_{in}$ rather than each term separately. For practical purposes, the mean of the random term is usually supposed to be zero. It can be shown that this assumption is not restrictive [Ben-Akiva and Lerman, 1985]. To derive assumptions about the variance of the random term, we observe that the scale of the utility may be arbitrarily specified. The arbitrary decision about the scale is equivalent to assuming a particular variance of the distribution of the error term.

3.4. Specific Choice Models

Once assumptions about the mean and the variance of the error term distribution have been defined, the focus is now on the actual functional form of this distribution. Many models have been explored. [Train, 2002] presents a good overview of a number of models. We will present a general overview of three of of them: The logit and probit model are the workhorses of discrete choice, but they rely on simplistic assumptions; Mixed logit is a more flexible model that is gaining popularity in the recent litterature.

3.4.1. Logit Model

Logit is by far the simplest and most widely used discrete choice model. It is derived under the assumption that ε_{in} is independent and identically distributed (i.i.d.) extreme value for all i. The critical part of the assumption is that the unobserved factors are uncorrelated over alternatives, as well as having the same variance for all alternatives. This assumption, while restrictive, provides a very convenient form for the choice probability. However, the assumption of independence can be inappropriate in some situations and the development of other models has arisen largely to avoid the independence assumption within logit. The logit choice probability is given by:

[3-5] $$P_{in} = \frac{e^{V_{in}}}{\sum_{j \in C_n} e^{V_{jn}}}$$

3.4.2. Probit Model

Probit is based on the assumption that the disturbances ε_{in} are distributed jointly normal. With a full covariance matrix, any pattern of correlation can be accommodated. The flexibility of the probit model in handling correlations over alternatives is its main advantages. One limitation of probit models is that they require normal distributions for all unobserved components of utility. In some situations, normal distributions are inappropriate and can lead to perverse predictions [Train, 2002]. Another limitation is that simulation of the choice probabilities is computationally expensive.

3.4.3. Mixed Logit

Mixed logit is a highly flexible model that allows the unobserved factors to follow any distribution. It obviates the limitations of standard logit by allowing for random taste variations and correlations in unobserved factors over time. Unlike probit, it is not restricted to normal distributions. Its derivation is straightforward, and simulation of its choice probabilities is computationally simple.

3.5. Predicting Choices

Our goal is to understand the behavioral process that leads to the decision maker's choice. The observed factors are labeled x, and the unobserved factors ε. The factors relate to the decision maker's choice through a function

$$i = h(x, \varepsilon)$$

This function is called the behavioral process and can be, for instance, any of the specific choice models described in the previous section. It is deterministic in the sense that given x and ε, the choice of the decision maker is fully determined. Since ε is unobserved, the decision maker's choice is not deterministic and cannon be predicted. Instead, as given in Equation [3-3], the probability of any particular outcome is derived. The unobserved terms are considered random with density $f(\varepsilon)$. The probability that the decision maker chooses a particular outcome from the set of all possible outcomes is simply the probability that the unobserved factors are such that the behavioral process results in that outcome:

$$P(i \mid x) = P(\varepsilon \text{ such that } h(x,\varepsilon) = i)$$

Define an indicator function

[3-6] $I[h(x,\varepsilon) = i]$

that takes the value of 1 when the statement in brackets is true and 0 when the statement is false. That is, $I[\cdot] = 1$ if the value of ε, combined with x, induces the agent to choose outcome i, and $I[\cdot] = 0$ if the value of ε, combined with x, induces the agent to choose some other outcome. Then the probability that the agent chooses outcome i is simply the expected value of this indicator function, where the expectation is over all possible values of the unobserved factors:

$$P(i \mid x) = P(I[h(x,\varepsilon) = i] = 1) = \int I[h(x,\varepsilon) = i] f(\varepsilon) d\varepsilon$$

Stated in this form, the probability is an integral – specifically an integral of an indicator for the outcome of the behavioral process over all possible values of the unobserved factors. To calculate this probability, the integral must be evaluated. There are three possibilities, each considered in a subsection below ([Train, 2002]).

3.5.1. Complete Closed-form Expression

For certain specifications of h and f, the integral can be expressed in closed form. In these cases, the choice probability can be calculated exactly from the closed-form formula. Logit is the most prominent example of a model estimated analytically.

3.5.2. Complete Simulation

If a closed-form solution does not exist for the integral, simulation is applicable in one form or another to practically any specification of h and f. Simulation relies on the fact that integration over a density is a form of averaging. Probit is the most prominent example of a model estimated by complete simulation.

3.5.3. Partial Simulation – Partial Closed-form

Suppose the random terms can be decomposed into two parts labeled ε_1 and ε_2 such that the integral over ε_2 given ε_1 is calculated exactly, while the integral over ε_1 is simulated. There are clear advantages to this approach over complete simulation. Analytic integrals are both more accurate and easier to calculate than simulated integrals. Therefore it is useful, when possible, to decompose the random terms so that some of them can be integrated analytically, even if the rest must be simulated. Mixed logit is a prominent example of a model that uses this decomposition effectively.

3.6. Maximum Likelihood Estimation

We turn to the problem of inferring the parameters $\beta_1,\ldots,\ \beta_k$ of Equation [3-4] from a sample of observations. Each observation consist of the following [Ben-Akiva and Lerman, 1985]:

1. An indicator variable defined in Equation [3-6].
2. Vectors of attributes x containing k values of the relevant variables.

Given a sample of N observations, the problem then becomes one of finding estimates $\hat{\beta}_1, \hat{\beta}_2, \ldots, \hat{\beta}_k$ that have some or all of the desirable properties of statistical estimators. The most widely used estimation procedure is the maximum likelihood. Conceptually, maximum likelihood estimation is straightforward, and will not be explained in this dissertation (refer to [Ben-Akiva and Lerman, 1985] for a detailed explanation). But it is worth noting that in some instances, the maximum likelihood estimation procedure can become computationally burdensome and relies on simulation techniques (see section 3.5.2 and 3.5.3).

3.7. Discrete Choice Models and Recommender Systems

This section will describe recent works done in discrete choice models that this research is going to use. [Walker, 2001] presents in her dissertation a generalized methodological framework that integrates extensions of discrete choice. Another recent advance in discrete choice is estimating individual level parameters to improve the modeling of decision-maker heterogeneity.

3.7.1. Framework of the Generalized Discrete Choice Model

The basic technique for integrating the methods is to start with the multinomial logit formulation, and then add extensions that relax simplifying assumptions and enrich the capabilities of the basic model. The extensions include [Walker, 2001]:

- *Specifying factor analytic (probit-like) disturbances* in order to provide a flexible covariance structure, thereby relaxing the Independence from Irrelevant Alternatives (IIA)[7] condition and enabling estimation of unobserved heterogeneity through techniques such as random parameters.
- *Combining revealed and stated* preferences in order to draw on the advantages of both types of data, thereby reducing bias and improving efficiency of the parameter estimates. As will be discussed in chapter 4, this extension will be particularly useful to the application of recommender systems in the context of academic advising.
- *Incorporating latent variables* in order to provide a richer explanation of behavior by explicitly representing the formation and effects of latent constructs such as attitudes and perceptions.
- *Stipulating latent classes* in order to capture latent segmentation, for example, in terms of taste parameters, choice sets, and decision protocols.

These generalized models often result in functional forms composed of complex multidimensional integrals. Therefore a key aspect of the framework is its 'logit kernel' formulation in which the disturbance of the choice model includes a logit like

[7] IIA states that if some alternatives are removed from a choice set, the relative choice probabilities from the reduced choice set are unchanged [Ben-Akiva and Lerman, 1985].

disturbance. This formulation can replicate all known error structures and it leads to a straightforward probability simulator (of a multinomial logit form) for use in maximum simulated likelihood estimation [Walker, 2001]. This proposed framework leads to a flexible, theoretically grounded, method for incorporating and integrating complex behavioral processes for use in recommender systems.

3.7.2. Estimating Individual Level Parameters

An exciting development in modeling has been the ability to estimate reliable individual level parameters for choice models. These individual parameters have been very useful in segmentation, identifying extreme individuals[8], and in creating appropriate choice simulators. Maximum likelihood and hierarchical Bayes techniques has both been used to infer the tastes of each sampled decision maker from estimates of the distribution of tastes in the population [Huber and Train, 2001]. The aim is to improve the modeling of consumer heterogeneity in order to create more accurate models on the aggregate level.

This research is interested in using these recent advances to develop a discrete choice model framework that is adapted for use in recommender systems. More details on the generalized discrete choice framework is provided in Chapter 5, and details about estimating and updating individual level parameters is provided in Chapter 6.

[8] Extreme individuals are individuals with tastes (estimated parameters) significantly different from the average tastes of the population.

Chapter 4: Designing the Survey and Collecting Data

Chapter 2 and 3 focused on giving a background and a framework for recommender systems and discrete choice models. We turn our attention to tackling the problem of designing and building the academic advising agent that will be used as a test bed for investigating the application of discrete choice models as a solution to the problem of recommending academic courses to students. This chapter is concerned with the second step of the framework set in Chapter 1, namely understanding the problem, identifying the attributes and collecting data. In order to achieve this objective, the factors that influence students' overall impressions of a class need to be understood. The hypothesis is that students tend to form an overall impression of a class based on factors or attributes relating to their own demographics or to specific class characteristics. Once those attributes are defined, discrete choice models can be used to estimate the overall utility or "how recommendable" a class is.

The chapter focuses first on describing the studies done to identify important attributes that should be included in our choice model. It then presents the different steps that led to the design of the final survey that was used to collect data from students. Finally, the last section is dedicated to the actual data collection and database design.

4.1. The Survey

4.1.1. Identifying the Attributes

One of our main hypotheses is that an academic course can be represented by a set of attributes that would define its attractiveness to a particular student. A preliminary questionnaire and a focus group were conducted during the Spring 2003 semester. Based on those findings, an online survey was designed and tested at the end of October 2003. The final version of the survey was released in April 2004.

The Underground Guide to Course VI

Course evaluation surveys are presented to students at the end of each term to collect the overall impression of each class. The 'Underground Guide to Course VI' (EECS department) compiles all the EECS course evaluations and is published each term. Its goal is to provide a document that accurately describes the contents, logistics, and character of EECS subjects, as well as the effectiveness of the instructors.

The 'Underground Guide' is used by most students as the definitive resource for finding out the contents and quality of courses. Students actively use it in deciding which term to take a required course and which electives to take.

A sample of the evaluation forms that is made available to MIT students is included in Appendix A. The attributes considered in these surveys are expected to have an important influence on students' ratings of the overall enjoyment of the course.

Preliminary Questionnaire

A Conjoint Analysis survey was designed to estimate the importance of five attributes: teaching style, instructor rating, workload, content, and convenience. A detailed description of the attributes is included in Appendix B. The study was based on a pen and paper survey of a representative group of 31 students in the MIT Sloan School of Management. Results showed that the perception of the content of a course has the highest utility in the conjoint analysis and thus is the most important factor in students' course selection. The relative importance of each of the evaluated attributes was as follows:

- Content (34.6%)
- Skills of the instructor (29.6%)
- Workload (14.1%)
- Convenience (12.2%)
- Teaching style (9.4%)

This preliminary study was helpful to confirm the hypothesis of the paramount importance of both the content and instructor attribute in course rating.

Focus group

The aim of the focus group was to identify attributes and strategies that students use to select courses. Two focus group meetings, each lasting one hour and each having 10 EECS students, were conducted in February 2003. The main findings are summarized as follows:

- Students rely heavily on the advice of upper classmates to get feedback on classes.
- Students use the 'Underground Guide' as a main source to get information on classes.
- Most of the students tend to choose and set EECS courses first, and then choose their humanities requirements to "fill the gaps" in their schedules.
- Future plans after graduation influence what courses students might like or might take, especially when it comes to choosing courses relevant to an area of concentration.
- Students who took a class with a professor that they like will probably look for other classes that this professor is giving.

4.1.2. Design of Stated Preference Experiment

Once the attributes are identified, the next task in a stated preference experiment is to decide on the combinations of attribute levels that should be presented to the survey takers. Each combination of the attributes describes a unique class. [Louviere et al (2000)] offers a comprehensive review on the methods of designing stated choice experiments.

Factorial Designs are designs in which each level of each attribute is combined with every level of all other attributes. For instance, the pilot study that will be presented in the next section deals with six 3 level attributes and three 2 level attributes. A full factorial design would therefore consist of $3^6 \times 2^3$ or 5832 total classes. Full factorial designs have very attractive statistical properties from the standpoint of estimating a model's parameters. In particular, complete factorial designs guarantee that all attributes effects of interests are truly independent. In our example, the complete factorial involves 5832 total combinations each of which requires a minimum of one observation in order to estimate all the possible effects.

However, the combination of profiles can be spread among survey takers so that every student doesn't have to evaluate 5832 different choices. If we estimate that about 100 students would take this pilot survey, we can spread the design into 100 different questionnaires. But even with this gain, a full factorial design would still require every student to evaluate 58 classes. Fractional factorial analysis can be used to design a subset of complete factorials.

If preliminary testing showed that a certain number of evaluations, say 7, is a comfortable number for survey takers to answer, the overall design needs to be restricted to 7×100 or 700 combinations of classes spread in 100 different surveys. Instead of sampling randomly from the full factorial design, fractional factorial analysis is used so that particular effects of interest are estimated as efficiently as possible.

Using the SAS (Statistical Analysis System) OPTEX procedure

The OPTEX procedure searches for optimal fractional factorial designs. Once a set of candidate design points and a linear model are specified, the procedure chooses points so that the terms in the model can be estimated as efficiently as possible. When a linear model is fit with an orthogonal design, the parameter estimates are uncorrelated, which means each estimate is independent of the other terms in the model. More importantly, orthogonality usually implies that the coefficients will have minimum variance. However, for many practical problems, orthogonal designs are simply not available particularly when the number of runs must be limited. In those situations, nonorthogonal designs must be used. In such cases OPTEX can be used to find optimal designs. Such designs are typically nonorthogonal; however they are efficient in the sense that the variances and covariances of the parameter estimates are minimized.

For example, suppose one wants a design for seven two-level factors that is limited to 32 runs. Among standard orthogonal arrays, the smallest appropriate 2^7 design has 128 runs. To generate an efficient non-orthogonal design the OPTEX procedure is invoked on the 128 candidate run, constraining OPTEX to choose 32 runs that would minimize the variance of the parameter estimates. The resulting would be the optimal fractional factorial designs consisting of 32 runs. More details on the OPTEX procedure and the algorithms used can be found in the SAS online documentation[9].

4.1.3. Pilot Survey

This online survey was divided into two parts: Demographic information and a stated preference questionnaire where survey takers were presented with seven pairs of hypothetical courses and were asked to choose the one they would prefer to take. The stated choice experiment is included in Appendix C. Each course option includes a list of nine attributes that characterize it (see Table C.1 in Appendix C).

Preliminary testing of the survey included surveys with 5, 7, 8, 10 and 12 hypothetical choice situations. Testing showed that most students felt that 7 evaluations per survey was a comfortable number and were able to finish the survey within reasonable amount of time. Based on an estimated 100 students taking the survey, a partial factorial design of 700 were spread in 100 different surveys. Effectively, 1400 profiles were created. Half of these profiles were randomly combined with the other half to form 700 different choice sets. Cases where one profile would be dominant (all the attributes have a more favorable rating) were avoided by randomly redrawing and repairing profiles until all dominant combinations were eliminated.

The survey was released during the last week of October 2003 and lasted for two weeks. It was conducted as an online survey. This allowed easy access to the EECS student

[9] http://v8doc.sas.com/sashtml/

population and facilitated the administration of 100 different surveys. The number of respondents turned out to be almost exactly as predicted (103 respondents). Analysis of demographic data showed a representative distribution of major concentration and academic years (sophomore, junior, seniors, and M.Eng) among survey takers although no sampling effort was made to have an adequate distribution. A mixed logit model was estimated for the given data. The results showed that topics, performance and instructor turned out to be the most important attributes when it comes to selecting courses. Demographics such as gender, major and academic year play an important role as well. The pilot test showed a certain number of shortcomings:

- The topic attribute should be more effectively quantified and should correspond to the different concentrations offered by the EECS department.
- The attributes levels chosen should be consistent with the ones used by the EECS' evaluation forms.
- Convenience as far as scheduling is concerned, is not of particular importance.
- Expected Performance (defined as a measure of how well the student think he/she will perform), although an important attribute in the SP setting, is hard to predict in RP situations for courses not already taken. It will be assumed that performance is a result and not a cause of a student enjoying a course, and hence will not be included in the final survey.

4.1.4. Final Survey

The online survey was divided into three parts: demographic information, revealed Preference (RP) and a stated preference (SP). The aim was to design a survey that would take less than 20 minutes to complete. Testing showed that evaluating a maximum of eight RP questions and five (SP) questions was a comfortable number to finish the survey within the allowed time frame.

Revealed Preference

Students were asked to evaluate a maximum of eight EECS classes that they've taken so far or that they took and eventually dropped. The attributes were selected to match the ones collected in the evaluation forms in the 'Underground Guide' (See Appendix D, Table D.1 for a description of those attributes). A sample revealed preference rating is shown in Appendix D, Figure D.1.

One difficulty was to choose one rating that captures the overall "recommendation" level of a course. An intuitive choice would be the "overall rating" of a subject, which is included in both the survey and EECS Evaluation forms. This rating, however, tries to capture the quality of the course in terms of teaching, workload and difficulty. On the other hand, a high quality course doesn't necessarily result in a high recommendation by a given student because of factors such as interest in the content of the lecture material, the timing when the course is taken, and the overall workload of the semester. Focus group discussions showed that most of the students take recommendations on classes to take from their peers. As a result, it was decided that the survey will mimic this behavior by choosing the one metric as being a scenario of rating the following question:

> "On a scale of 1 to 7, would you recommend this class to a fellow student who resembles you when you took it (same background, abilities, interests, and temperament)? "

Stated Preference

The survey takers were presented with five sets of two hypothetical courses and were asked to choose the one they would more likely take. Since the aim is to ultimately combine SP and RP questions in the final model, the attributes used in the SP scenarios and their levels where chosen in a way to be as close as possible to the RP questions. Table D.2 and Figure D.2 in appendix D describe a typical SP scenario.

Based on an estimated 300 students taking the survey and five SP questions per survey and using the methodology described in 4.1.2, a partial factorial design of 3000 profiles was created. Half of these profiles were randomly combined with the other half to form 1500 choice sets spread in 300 surveys.

4.2. Data Collection

4.2.1. Administrating the Survey

MIT's Information System's Web Survey Service (http://web.mit.edu/surveys/) advises researchers that surveys be administered to students between the middle of February and the month following Spring Break. These periods tend to have the best response rates. The survey was released two weeks after the Spring Break (12th April 2004) and lasted for two weeks. It was conducted as an online survey. As an incentive, a lottery drawing for a prize of $1000 was to be awarded to one of the EECS students who completed the survey. The total number of completed surveys turned out to be 328.

4.2.2. Collecting data from the 'Underground Guide'

The subject ratings found in the 'Underground Guide' are average values given by a usually large sample of the enrolled students in a class for a given term. A software program was created to systematically go through all the terms of the 'Undergraduate Guide' that are available online. For each term, a database of the subjects offered is constructed. The following information is collected for each subject: Lecturers' Ratings, Overall Rating, Difficulty, Workload, Response Rate and the list of Prerequisites

4.2.3. Creating the Database

The next step is to merge the revealed preference data from the survey with the course evaluation information collected from the 'Underground Guide'. This was accomplished by mapping the subjects collected from the survey to the corresponding subject and term from the evaluation forms. The revealed preference data that couldn't be mapped was simply dropped from the study.

A number of attributes which are not directly available from the raw data needed to be derived or calculated. All the processes describe below were automated in software.

Required or Elective: A class that is taken as a requirement might be rated differently than one taken as an elective. Hence, it would be useful to classify the RP rated classes into one of these two categories. In order to decide which category a course belongs to, the degree that a student is pursing (obtained from the demographic information in the survey) was matched with the list of the required classes for that degree (the lists for the three different degrees can be found in the 'Underground Guide'). This process is performed for all the classes in the RP data.

Concentration: The preliminary studies showed that the level of relevancy of the content of a given subject to the student's interests is expected to be an important attribute. In order to quantify this relevancy, students were asked to indicate on a scale of 1 to 4 the importance, as far as their interests are concerned, of each of the seven engineering concentrations that the EECS department offers. This indicator can be used to quantify the relevance of a given class depending on the concentration it belongs to. The 'Underground Guide' classifies all the classes under one of the seven concentrations. The software program uses this information to associate all the classes in the RP data with the relevancy indicator of the corresponding concentration.

Timing: The model does not directly deal with planning subjects over semesters. On the other hand, the timing when a subject is taken (e.g. during the spring term of a student's sophomore year) is relevant to how much a student will recommend the subject in question. For instance, an advanced class that is taken "too early" might not be as recommendable as if it were to be taken later on even if in both cases the prerequisites were satisfied. In the given RP data, the timing of a class was calculated relative to the fall semester of a student's sophomore year (which is identified as "Term 0"). A subject taken in the spring of a student's sophomore year would therefore be identified as "Term 1", and so on. The average timing that a given subject is taken was calculated from the timings given by the RP data. Four variables where created:

- "Term Late"/"Term Early": a binary variable indicating if a class was taken after/before its average term
- "How Late"/"How Early": two variables indicating how late/early the class was taken with respect to its average term.

Stated Preference Data: The stated preference part of the online survey asked the respondent to rate how likely he/she would take one the two presented hypothetical classes. In order to match the RP data with the SP data, the SP choice scenario with two subjects needed to be transformed into a rating scenario with one subject. This was performed by taking the differences in the attributes of the two hypothetical subjects. As in the RP data, the concentration field was first replaced with the concentration relevancy indicator. Once this field was quantified, the difference in the concentration indicators was taken.

In this chapter, we presented the general methodology that was used to identify the important attributes in the decision process of rating courses, to design an online survey that was taken by EECS students, and finally to collect and store the data so it can be easily used to construct our choice model. The next chapter uses this collected data to estimate the student model which will be used to process the information available and arrive at a unique prediction of how "recommendable" a course would be to a given student.

Chapter 5: Student Rating Model

This chapter tackles the problem of estimating a student rating model that will be used as the backbone of the recommender system engine. The chapter corresponds to the third step of the recommender system framework set in Chapter 1 and is structured as follows. Section 5.1 presents the theoretical background of the generalized discrete choice framework and sets its general notation. Section 5.2 develops a framework for the student rating model. Section 5.3 develops the equations that are used to estimate the student rating model of Section 5.2. Those equations are derived from a simplification of the more general notation given in section 5.1. These simplifications are described at the beginning of section 5.3. The model estimation and discussion of the results obtained concludes the chapter.

5.1. Theoretical Background

5.1.1. The Discrete Choice Model

The most common discrete choice model is the linear in parameters, utility maximizing, multinomial logit model (MNL), developed by [McFadden, 1974], which is specified as:

[5-1] $U_{in} = X_{in}\beta + \varepsilon_{in}$,

[5-2] $y_{in} = \begin{cases} 1, & \textit{if } U_{in} = \max\limits_{j}\{U_{jn}\} \\ 0, & \textit{otherwise} \end{cases}$

where: n denotes an individual, $n = 1, \ldots, N$;

i,j denote alternatives, $i, j = 1, \ldots, J_n$;

U_{in} is the utility of alternative i as perceived by individual n;

y_{in} is the choice indicator (equals to 1 if alternative i is chosen, and 0 otherwise);

β is a $(K \times 1)$ vector of unknown parameters;

ε_{in} is an i.i.d. Gumbel distributed random variable;

X_{in} is a $(1 \times K)$ vector of explanatory variables describing n and i.

Equation [5-1] and [5-2] lead to the following multinomial logit formulation:

$$[5\text{-}3] \qquad P_{in}(y_{in}=1 \mid X_{in};\beta,\mu)=\frac{e^{\mu(X_{in}\beta)}}{\sum_{j\in C_n} e^{\mu(X_{jn}\beta)}},$$

where: C_n is the choice set faced by individual n, comprised of J_n alternatives;

μ is the scale parameter of the Gumbel distributed variate ε_{in}

[Ben-Akiva and Lerman, 1985] offer a background of the random utility model and a detailed description of the multinomial logit model. [Walker, 2001] presents a framework for extending this classic discrete choice model. This framework will be used as the basis for our student model. We will start with the base model, and gradually extend it in the subsequent sections to arrive to the generalized discrete choice model that will be used in this dissertation.

5.1.2. Base Model: Ordinal Logit Model

In surveys, respondents are often asked to provide ratings. In these types of questions, the responses are ordered. A natural way to represent this decision process is to think of the respondent as having some level of utility associated with the question. In this dissertation, a student is assumed to associates a utility U with a given course. The higher levels of U mean that the student thinks that the course is recommendable. The student chooses a response on the basis of the level of U. On a 7 point preference rating, where level 1 is "Not Recommended" and level 7 is "Highly Recommended", if U is above some cutoff, which we label τ_6, then the answer is level 7 ("Highly

Recommended"). If the answer is below τ_6 but above another cutoff, τ_5, then the answer is level 6. And so on. Since only the difference between the utilities matters, the utilities are specified in a differenced form, and threshold values (τ) are specified in the utility scale such as:

$$\begin{aligned}
P_n(1) &= P(\tau_0 < U_n \le \tau_1),\\
P_n(2) &= P(\tau_1 < U_n \le \tau_2),\\
P_n(3) &= P(\tau_2 < U_n \le \tau_3),\\
P_n(4) &= P(\tau_3 < U_n \le \tau_4),\\
P_n(5) &= P(\tau_4 < U_n \le \tau_5),\\
P_n(6) &= P(\tau_5 < U_n \le \tau_6),\\
P_n(7) &= P(\tau_6 < U_n \le \tau_7),
\end{aligned}$$

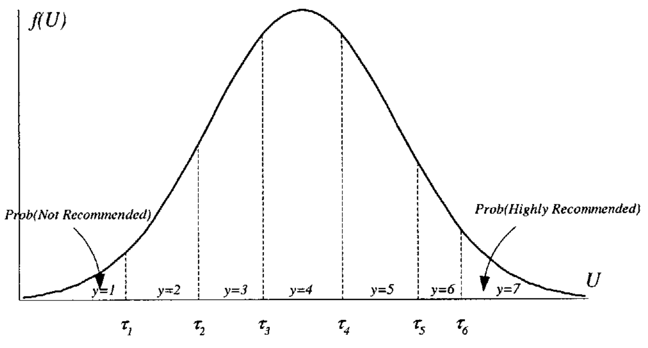


Figure 5-1. Distribution of preference

where: $\tau_0 = -\infty$ and $\tau_7 = \infty$

We define the ordinal choice indicator as:

[5-4] $y_n = i, \text{ if } \tau_{i-1} < U_n \le \tau_i, \qquad i = 1,\ldots,7.$

and

$$y_{in} = \begin{cases} 1, & \text{if } y_n = i \\ 0, & \text{otherwise} \end{cases}.$$

Using Equation [5-1], U_n is then defined as:

[5-5] $U_n = X_n\beta + \varepsilon_n,$

where the alternative specific subscript i has been dropped. The likelihood for each ordinal preference rating is then:

[5-6] $$P_n(y_n \mid X_n; \beta, \tau, \mu) = \prod_{i=1}^{7}\left(\frac{1}{1+e^{\mu(X_n\beta-\tau_i)}} - \frac{1}{1+e^{\mu(X_n\beta-\tau_{i-1})}}\right)^{y_{in}}$$

where τ on the left hand side represents a vector holding the values of τ_i.

5.1.3. Mixed Ordered Logit

Mixed logit is a highly flexible model that can approximate any random utility (see [McFadden and Train, 2000]). It eliminates the limitation of standard logit by allowing for random taste variation and correlation in unobserved factors over time. Its derivation is straightforward, and simulation of its choice probabilities is computationally simple. [Train, 2003] and [Walker, 2001] present a comprehensive description of the mixed logit model. [Train, 2003] defines mixed logit probabilities as integrals of standard logit probabilities over a density of parameters. Stated more explicitly, a mixed logit model is any model whose choice probabilities can be expressed in the form:

$$P_{in} = \int L_{in}(\beta) f(\beta) d\beta$$

where $L_{in}(\beta)$ is the logit probability evaluated at parameters β and $f(\beta)$ is a multivariate density function called the mixing distribution.

Random Coefficients Model

One application of mixed logit that is widely used is based on random coefficients. In this context, Equation [5-5] can be rewritten as:

[5-7] $U_n = X_n \beta_n + \varepsilon_n$ where $\beta_n \sim N(\beta, \Sigma_\beta)$,

β_n is a $(K \times 1)$ vector of random coefficients for person n representing that person's tastes. The equation is written for a normal distribution, although other distributions (lognormal, etc.) are possible. The coefficients vary over decision makers in the population with density $f(\beta)$. This density is a function of parameters θ that represent, in the case of $f(\beta)$ being a normal distribution, the mean β and covariance Σ_β of β_n.

The probability conditional on β_n for individual n is the standard ordered logit given in Equation [5-6], with β substituted by β_n. However β_n is typically unknown. The unconditional choice probability is therefore the integral over all possible values of β_n. Under this extension, the likelihood function becomes:

[5-8]
$$
\begin{aligned}
P_n(y_n \mid X_n;\theta,\tau,\mu) &= \int_\beta P_n(y_n \mid X_n,\beta;\tau,\mu) f(\beta \mid \theta) d\beta \\
&= \int_\beta \left(\prod_{i=1}^{7} \left(\frac{1}{1+e^{\mu(X_n\beta-\tau_i)}} - \frac{1}{1+e^{\mu(X_n\beta-\tau_{i-1})}} \right)^{y_{in}} \right) f(\beta \mid \theta) d\beta
\end{aligned}
$$

where as before θ represents the mean β and covariance Σ_β of a normally distributed β_n.

Panel Data

The specification for mixed ordered logit data can be generalized for repeated choices by each sampled respondents. Each observation in the data represents one of the observations for one of the respondents. For instance, if we had 10 observations for each of 50 individuals, our dataset would contain $10 \times 50 = 500$ observations in total.

In the case of panel data estimators, the observations are not independent, but groups of observations (grouped by respondents) are independent. The utility function given in Equation [5-7] can be rewritten as:

[5-9] $$U_{nt} = X_{nt}\beta_n + v_n + \varepsilon_{nt}, \qquad v_n \sim N(0,\sigma_v^2) \qquad \varepsilon_{nt} \sim i.i.d.\,Gumbel$$

where n identifies the respondents, and t indexes responses across observations given by respondent n. The random effects v_n are assumed $N(0,\sigma_v^2)$, and ε_{nt} are assumed i.i.d. Gumbel, and v_n and ε_{nt} are assumed uncorrelated. Rewriting Equation [5-9] with a standard normal disturbance term for the panel data we obtain:

[5-10] $U_{nt} = X_{nt}\beta_n + \sigma_v \tilde{v}_n + \varepsilon_{nt}$, where $\tilde{v}_n \sim N(0,1)$,

Conditional on $\tilde{v}_n$ and β_n, the probability that a given respondent makes a sequence $t=1,\dots,T$ of ratings is the product of ordered logit formulas:

[5-11] $$P_n(y_n \mid X_n, \beta_n, \tilde{v}_n; \tau, \mu, \sigma_v) = \prod_{t=1}^{T} \prod_{i=1}^{7} \left(\frac{1}{1+e^{\mu(X_{nt}\beta_n + \sigma_v \tilde{v}_n - \tau_i)}} - \frac{1}{1+e^{\mu(X_{nt}\beta_n + \sigma_v \tilde{v}_n - \tau_{i-1})}} \right)^{y_{int}}$$

where T is a constant[10] denoting the total number of observations given by respondent n, and

$$y_{nt} = i, if\ \tau_{i-1} < U_{nt} \le \tau_i, \qquad i = 1,\dots,7.$$

and

$$y_{\text{int}} = \begin{cases} 1, & if\ y_{nt} = i \\ 0, & otherwise \end{cases}.$$

The unconditional probability is the integral of these products over all values of $\tilde{v}_n$ and β_n. The likelihood function for random effects ordered logit can be written as:

$$\begin{aligned} P(y \mid X;\theta,\sigma_v,\tau,\mu) &= \prod_{n=1}^{N} P_n(y_n \mid X_n;\theta,\sigma_v,\tau,\mu) \\ &= \prod_{n=1}^{N} \iint_{\beta,v} P_n(y_n \mid X_n,\beta,\tilde{v};\tau,\mu,\sigma_v) f(\beta \mid \theta)\phi(\tilde{v})\, d\tilde{v}\, d\beta \end{aligned} \qquad [5\text{-}12]$$

where N is the total number of respondents and ϕ is the standard normal density function.

5.1.4. Combining Revealed and Stated Preference Data

Revealed preference (RP) data reflect people's actual choices in real-world situations. Stated preference (SP) data are collected in experimental situations where respondents are presented with hypothetical choice situations. There are advantages and limitations to each type of data (see Table 5-1).

Table 5-1. RP vs. SP Data

	Revealed Preference	*Stated Preference*
Advantages	• Reflect actual choices	• Contains as much variation in each attribute as needed
Limitations	• Limited to currently available choice situations and attributes • Insufficient variation in relevant factors	• What people say they would do might not reflect what they actually do.

[10] For simplicity of notation, we are assuming that T is constant across respondents. One can easily generalize for a different number of observations across respondents by substituting T by T_n.

By combining SP and RP data, the advantages of each can be obtained while mitigating their respective limitations. Procedures for estimating combined SP and RP discrete choice models are described by [Ben-Akiva and Morikawa, 1990], [Hensher and Bradeley, 1993], and [Hensher et al., 1999]. As will be described in a later section, for every respondent we have ratings based on classes that have been taken (RP), and ratings based on hypothetical scenarios (SP). The rating for the RP and SP questions is the same – that is rating a course on a scale of 7, 1 being "Not Recommended" and 7 being "Highly Recommended". However, responses to the RP and SP questions aren't perfectly comparable because of differences in the choice settings:

- SP questions presents students with hypothetical courses.
- Students didn't actually take the courses in the SP cases, thus potentially affecting how they perceive the course.

To capture those effects, we construct separate models for the RP and SP data where we divide *T*, the total number of observations per respondent, into:

- RP observations: denoted by *r=1,...,R*
- SP observations: denoted by *s=1,...,S*

Once again for notational simplicity *R* and *S* are assumed to be constant across respondents and can be generalized by using R_n and S_n. For revealed preferences, we rewrite the utility in Equation [5-10] as:

[5-13] $U_{nr}^{RP} = X_{nr}\beta_n + W_{nr}\delta + \sigma_v^{RP}\tilde{v}_n + \varepsilon_{nr}^{RP}$

where: X_{nr} is the $(1 \times K)$ vector of attributes and characteristics present in both RP and SP setting,

W_{nr} is the $(1 \times K^{RP})$ vector of variables present only in the RP setting,

δ is a $(K^{RP} \times 1)$ vector of unknown parameters,

$\tilde{v}_n$ is standard normal $\tilde{v}_n \sim N(0,1)$,

ε_{nr}^{RP} is an i.i.d. Gumbel random variable with scale parameter μ^{RP}

For simplicity δ was assumed to be fixed, but one could also assume it to be randomly distributed using the same approach as the one used for β_n. Equation [5-11] for the RP model can be rewritten as:

[5-14]
$$P_n^{RP}\left(y_n^{RP} \mid X_n, W_n, \beta_n, \tilde{v}_n; \tau, \delta, \mu^{RP}, \sigma_v^{RP}\right) = \prod_{r=1}^{R}\prod_{i=1}^{7}\left(\frac{1}{1+e^{\mu^{RP}(X_{nr}\beta_n + W_{nr}\delta + \sigma_v^{RP}\tilde{v}_n - \tau_i)}} - \frac{1}{1+e^{\mu^{RP}(X_{nr}\beta_n + W_{nr}\delta + \sigma_v^{RP}\tilde{v}_n - \tau_{i-1})}}\right)^{y_{inr}^{RP}}$$

For the SP model, we write the utility as:

[5-15] $$U_{ns}^{SP} = X_{ns}\beta_n + Z_{ns}\chi + \sigma_v^{SP}\tilde{v}_n + \tau^{SP} + \varepsilon_{ns}^{SP}$$

where:

	Z_{ns}	is the $(1 \times K^{SP})$ vector of variables present only in the SP setting,
	χ	is a $(K^{SP} \times 1)$ vector of unknown parameters,
	τ^{SP}	is a constant present only in the SP setting,
	$\tilde{v}_n$	is the same random value as in Equation [5-13],
	ε_{ns}^{SP}	is an i.i.d. Gumbel random variable with scale parameter μ^{SP}

Equation [5-11] for the SP model can be rewritten as:

[5-16]
$$P_n^{SP}\left(y_n^{SP} \mid X_n, Z_n, \beta_n, \tilde{v}_n; \tau, \tau^{SP}, \chi, \mu^{SP}, \sigma_v^{SP}\right) = \prod_{s=1}^{S}\prod_{i=1}^{7}\left(\frac{1}{1+e^{\mu^{SP}(X_{ns}\beta_n + Z_{ns}\chi + \sigma_v^{SP}\tilde{v}_n + \tau^{SP} - \tau_i)}} - \frac{1}{1+e^{\mu^{SP}(X_{ns}\beta_n + Z_{ns}\chi + \sigma_v^{SP}\tilde{v}_n + \tau^{SP} - \tau_{i-1})}}\right)^{y_{ins}^{SP}}$$

Combining equations [5-14] and [5-16] for the RP and SP data, we can rewrite the likelihood function given in Equation [5-12] as:

[5-17]
$$P\left(y^{RP}, y^{SP} \mid X, W, Z; \theta, \sigma_v^{RP}, \sigma_v^{SP}, \tau, \tau^{SP}, \delta, \chi, \mu^{SP}\right) = \prod_{n=1}^{N}\left\{ \iint_{\beta, v} P_n^{RP}\left(y_n^{RP} \mid X_n, W_n, \beta, \tilde{v}; \tau, \delta, \sigma_v^{RP}\right) \times P_n^{SP}\left(y_n^{SP} \mid X_n, Z_n, \beta, \tilde{v}; \tau, \tau^{SP}, \chi, \mu^{SP}, \sigma_v^{SP}\right) f(\beta \mid \theta)\phi(\tilde{v})\, d\tilde{v}\, d\beta \right\}$$

For identification purposes, the scale of μ^{RP} is set to be equal to 1 without loss of any generality.

5.1.5. Latent Variables Models

A latent variable can be defined as a random variable whose realizations are hidden from us (see [Skrondal and Rabe-Heskth, 2004]). This is in contrast to explicit or manifest variables where the realizations are observed. The idea is to explicitly incorporate latent constructs such as attitudes and perceptions, or any amorphous concept affecting choice, in an effort to produce more behaviorally realistic models.

A general approach to synthesizing models with latent variables has been advanced by a number of researchers who developed the structural and measurement equation framework and methodology for specifying latent variable models (see [Keesling, 1972], [Wiley, 1973], and [Bentler, 1980]).

[Ben-Akiva et al., 2002] gives an excellent presentation of a general specification and estimation method for the integrated model. This methodology will be used to extend our model to incorporate latent variables as explanatory factors. It incorporates indicators of the latent variables provided by responses to survey questions to aid in estimating the model. A simultaneous estimator is used, which results in latent variables that provide the best fit (in the maximum likelihood sense) to both the choice and the latent variables indicators. Figure 2 presents the integrated model as given by [Ben-Akiva et al., 2002].

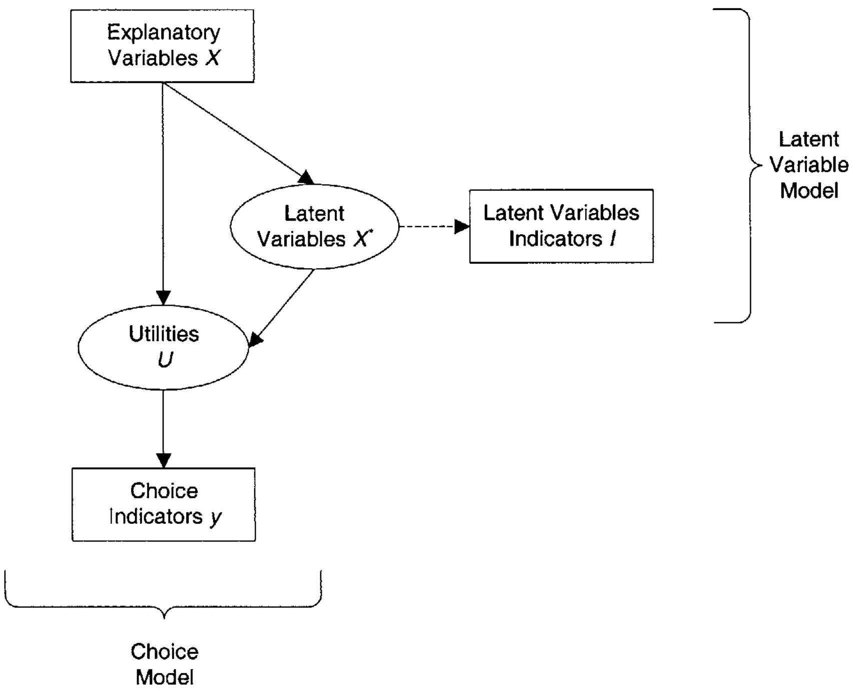


Figure 5-2. Integrated Choice and Latent Variable Model ([Ben-Akiva et al (2002)])

Using Equation [5-10], the general choice model with latent attributes is specified by the following. The RP/SP data distinction will be ignored for simplicity of the notation and hence the subscript *t* will be used again to denote an observation.

[5-18] $U_{nt} = X_{nt}\beta_n + X^*_{nt}\gamma + \sigma_v \tilde{v}_n + \varepsilon_{nt}$,

where: X^*_{nt} is an $(1 \times L)$ vector of stacked latent variables X^*_{lnt}, $l=1,\ldots,L$ denotes the index of a latent variable,

γ is an $(L \times 1)$ vector of unknown parameters.

Structural Model

For the latent variable model:

[5-19] $X^*_{lnt} = X_{lnt}\lambda_l + \omega_{lnt}$, $\quad l = 1,2,\ldots,L$

where: X^*_{lnt} is the latent variable *l* describing latent characteristics of individual *n* and observation *t*,

X_{lnt} is a $(1\times K_l)$ vector of explanatory variables describing n for observation t and latent variable l,

λ_l is a $(K_l\times 1)$ vector of unknown parameters

ω_{lnt} is the disturbance term for latent variable l

The multivariate distribution of the latent variables given the observed variables will be noted as:

[5-20] $f_{X^*}\left(X^*_{nt} \mid X_{nt};\lambda,\Sigma_\omega\right)$

where f_{X^*} is a multivariate normal distribution with a covariance matrix Σ_ω.

Let ω_{nt} be an $(L\times 1)$ vector of ω_{lnt}, such that $\omega_{nt}\sim N(0,\Sigma_\omega)$. The hypothesis is that the errors ω_{lnt} are not independent and hence Σ_ω is not diagonal, and that the errors have no serial correlation. For estimation, it is desirable to specify the factors such that they are independent, and we therefore decompose ω_{nt} as follows:

$$\omega_{nt} = \Gamma\zeta_{nt},$$

where Γ is the $(L\times L)$ lower triangular Cholesky matrix such that $\Gamma\Gamma' = \Sigma_\omega$ and ζ_{nt} is a an $(L\times 1)$ vector of standard independently distributed variables ($\zeta_{nt}\sim iid\ N(0,I)$). Define F_l as being the l^{th} row of an $(L\times L)$ identity matrix. Using this notation leads to:

$$\omega_{lnt} = F_l\Gamma\zeta_{nt}$$

which we use to obtain:

[5-21] $X^*_{lnt} = X_{lnt}\lambda_l + F_l\Gamma\zeta_{nt}$

Using this decomposition, the multivariate distribution of X^*_{nt} given by Equation [5-20] is replaced by a distribution of ζ_{nt} given by:

$$f_\zeta\left(\zeta_{nt} \mid X_{nt};\lambda,\Sigma_\omega\right) = \prod_{l=1}^{L}\phi(\zeta_{lnt})$$

where ϕ is the standard normal density function.

The choice model given in Equation [5-18] becomes:

$$[5\text{-}22] \quad U_{nt} = X_{nt}\beta_n + \sum_{l=1}^{L}(X_{lnt}\lambda_l + F_l\Gamma\zeta_{nt})\gamma_l + \sigma_v\,\tilde{v}_n + \varepsilon_{nt},$$

where γ_l is an unknown parameter acting as the loading factor for X^*_{lnt}.

Measurement Model

The measurement equations of the latent variable model are written as:

$$[5\text{-}23] \quad I_{mnt} = X^*_{nt}\alpha_m + \eta_{mnt} \qquad \eta_{nt} \sim N(0, \Sigma_\eta\ diagonal)$$

where: M denotes the index of the indicator $m=1,2,...,M$,

I_{mnt} is an indicator of X^*_{nt}

α_m is an $(L\times 1)$ vector of coefficients to be estimated.

The disturbance of the measurement equations of the latent variable model were assumed to be normally and independently distributed ($\eta_{mnt} \sim N(0,\sigma^2_{\eta_m})$) with no serial correlation, and the indicators are assumed to be conditionally (on X^*_{nt}) independent. This would result in the following density:

$$[5\text{-}24] \quad f_l(I_{nt} \mid X^*_{nt};\alpha,\Sigma_\eta) = \prod_{m=1}^{M}\frac{1}{\sigma_{\eta_m}}\phi\left(\frac{I_{mnt}-X^*_{nt}\alpha_m}{\sigma_{\eta_m}}\right),$$

where ϕ is the standard normal density function.

Integrated Model

The most intuitive way to create the likelihood function for the integrated model is to start with the likelihood without the latent variables. Writing again Equation [5-17]:

$$P(y^{RP}, y^{SP} \mid X,W,Z;\theta,\sigma_v^{RP},\sigma_v^{SP},\tau,\tau^{SP},\delta,\chi,\mu^{SP})$$

$$[5\text{-}25] \qquad = \prod_{n=1}^{N}\left\{ \iint_{\beta,v} P_n^{RP}(y_n^{RP} \mid X_n,W_n,\beta,\tilde{v};\tau,\delta,\sigma_v^{RP}) \times P_n^{SP}(y_n^{SP} \mid X_n,Z_n,\beta,\tilde{v};\tau,\tau^{SP},\chi,\mu^{SP},\sigma_v^{SP})\, f(\beta\mid\theta)\phi(\tilde{v})\,d\tilde{v}\; d\beta\right\}$$

Taking the RP likelihood part (which is given in Equation [5-14]) in the previous equation and using Equation [5-18] to add the latent variables results in the following:

[5-26]
$$P_n^{RP}\left(y_n^{RP} \mid X_n, X_n^{*RP}, W_n, \beta_n, \tilde{\nu}_n; \tau, \delta, \gamma, \sigma_\nu^{RP}\right) = \prod_{r=1}^{R}\prod_{i=1}^{7}\left(\frac{1}{1+e^{(X_{nr}\beta_n + W_{nr}\delta + X_{nr}^{*RP}\gamma + \sigma_\nu^{RP}\tilde{\nu}_n - \tau_i)}} - \frac{1}{1+e^{(X_{nr}\beta_n + W_{nr}\delta + X_{nr}^{*RP}\gamma + \sigma_\nu^{RP}\tilde{\nu}_n - \tau_{i-1})}}\right)^{y_{inr}^{RP}}$$

The resulting likelihood function for the RP model is then the integral of the previous equation over the distribution of the latent constructs given by Equation [5-20]:

[5-27]
$$P_n^{RP}\left(y_n^{RP} \mid X_n, W_n, \beta_n, \tilde{\nu}_n; \tau, \delta, \gamma, \sigma_\nu^{RP}, \lambda, \Sigma_\omega\right) = \prod_{r}^{R} \int_{X_{nr}^{*RP}} P_{nr}^{RP}\left(y_{nr}^{RP} \mid X_{nr}, X_{nr}^{*RP}, W_{nr}, \beta_n, \tilde{\nu}_n; \tau, \delta, \gamma, \sigma_\nu^{RP}\right) f_{X^*}\left(X_{nr}^{*RP} \mid X_{nr}; \lambda, \Sigma_\omega\right) dX_{nr}^{*RP}$$

We introduce indicators to both improve the accuracy of estimates of the structural parameters as well as to allow for their identification. Assuming the error components $(\omega, \varepsilon, \eta)$ are independent, the joint probability distribution of the observable variable y_n^{RP} and I_n^{RP}, conditional on the exogenous variables X_n and W_n, is:

[5-28]
$$\begin{aligned} P_n^{RP}&\left(y_n^{RP}, I_n^{RP} \mid X_n, W_n, \beta_n, \tilde{\nu}_n; \tau, \delta, \gamma, \sigma_\nu^{RP}, \lambda, \Sigma_\omega, \alpha^{RP}, \Sigma_\eta^{RP}\right) \\ &= \prod_{r=1}^{R} \int_{X_{nr}^{*RP}} P_{nr}^{RP}\left(y_{nr}^{RP} \mid X_{nr}, X_{nr}^{*RP}, W_{nr}, \beta_n, \tilde{\nu}_n; \tau, \delta, \gamma, \sigma_\nu^{RP}\right) \\ &\qquad \times f_I^{RP}\left(I_{nr}^{RP} \mid X_{nr}^{*RP}; \alpha^{RP}, \Sigma_\eta^{RP}\right) f_{X^*}\left(X_{nr}^{*RP} \mid X_{nr}; \lambda, \Sigma_\omega\right) dX_{nr}^{*RP} \end{aligned}$$

Using the same procedure, we obtain the following SP model:

[5-29]
$$\begin{aligned} P_n^{SP}&\left(y_n^{SP} \mid X_n, X_n^{*SP}, Z_n, \beta_n, \tilde{\nu}_n; \tau, \tau^{SP}, \chi, \mu^{SP}, \gamma, \sigma_\nu^{SP}\right) \\ &= \prod_{s=1}^{S}\prod_{i=1}^{7}\left(\frac{1}{1+e^{\mu^{SP}(X_{ns}\beta_n + Z_{ns}\chi + X_{ns}^{*SP}\gamma + \sigma_\nu^{SP}\tilde{\nu}_n + \tau^{SP} - \tau_i)}}\right., \\ &\qquad \left. - \frac{1}{1+e^{\mu^{SP}(X_{ns}\beta_n + Z_{ns}\chi + X_{ns}^{*SP}\gamma + \sigma_\nu^{SP}\tilde{\nu}_n + \tau^{SP} - \tau_{i-1})}}\right)^{y_{ins}^{SP}} \end{aligned}$$

from which the resulting likelihood function for the SP model is derived:

$$P_n^{SP}\left(y_n^{SP} \mid X_n^{SP}, Z_n, \beta_n, \tilde{v}_n; \tau, \chi, \gamma, \sigma_v^{SP}, \mu^{SP}, \lambda, \Sigma_\omega\right)$$

[5-30] $$= \prod_{s=1}^{S} \int_{X_{ns}^{*SP}} P_{ns}^{SP}\left(y_{ns}^{SP} \mid X_{ns}, X_{ns}^{*SP}, Z_{ns}, \beta_n, \tilde{v}_n; \tau, \tau^{SP}, \chi, \mu^{SP}, \gamma, \sigma_v^{SP}\right) \times f_{X^*}\left(X_{ns}^{*SP} \mid X_{ns}; \lambda, \Sigma_\omega\right) dX_{ns}^{*SP}$$

And finally, introducing indicators leads to:

$$P_n^{SP}\left(y_n^{SP}, I_n^{SP} \mid X_n, Z_n, \beta_n, \tilde{v}_n; \tau, \chi, \gamma, \sigma_v^{SP}, \mu^{SP}, \lambda, \Sigma_\omega, \alpha^{SP}, \Sigma_\eta^{SP}\right)$$

[5-31] $$= \prod_{s=1}^{S} \int_{X_{ns}^{*SP}} P_{ns}^{SP}\left(y_{ns}^{SP} \mid X_{ns}, X_{ns}^{*SP}, Z_{ns}, \beta_n, \tilde{v}_n; \tau, \tau^{SP}, \chi, \mu^{SP}, \gamma, \sigma_v^{SP}\right) \times f_I^{SP}\left(I_{ns}^{SP} \mid X_{ns}^{*SP}; \alpha^{SP}, \Sigma_\eta^{SP}\right) f_{X^*}\left(X_{ns}^{*SP} \mid X_{ns}; \lambda, \Sigma_\omega\right) dX_{ns}^{*SP}$$

Plugging in the derived RP and SP likelihood given by Equations [5-28] and [5-31] back into Equation [5-25], we obtain the likelihood for the combined model:

$$P\left(y, I^{RP}, I^{SP} \mid X, W, Z; \theta, \sigma_v^{RP}, \sigma_v^{SP}, \tau, \tau^{SP}, \delta, \chi, \mu^{SP}, \gamma, \lambda, \Sigma_\omega, \alpha^{RP}, \Sigma_\eta^{RP}, \alpha^{SP}, \Sigma_\eta^{SP}\right)$$

[5-32] $$= \prod_{n=1}^{N} \left\{ \iint_{\beta, v} P_n^{RP}\left(y_n^{RP}, I_n^{RP} \mid X_n^{RP}, W_n, \beta, \tilde{v}; \tau, \delta, \gamma, \sigma_v^{RP}, \lambda, \Sigma_\omega, \alpha^{RP}, \Sigma_\eta^{RP}\right) \times P_n^{SP}\left(y_n^{SP}, I_n^{SP} \mid X_n^{SP}, Z_n, \beta, \tilde{v}; \tau, \chi, \mu^{SP}, \gamma, \sigma_v^{SP}, \lambda, \Sigma_\omega, \alpha^{SP}, \Sigma_\eta^{SP}\right) \times f(\beta \mid \theta) \phi(\tilde{v})\, d\tilde{v}\, d\beta \right\}$$

Equation [5-32] can be written more compactly as:

[5-33] $$L = \prod_{n=1}^{N} P_n(y_n, I_n \mid X_n, W_n, Z_n; \theta, \tau, \tau^{SP}, \delta, \chi, \mu^{SP}, \gamma, \lambda, \alpha, \Sigma)$$

where:

where:	Σ	denotes the variances and covariances $\sigma_v^{RP}, \sigma_v^{SP}$, Σ_ω, Σ_η^{SP}, and Σ_η^{RP};
	I_n	denotes RP (I_n^{RP}) and SP (I_n^{SP}) indicators.
	α	denotes α^{RP} and α^{SP} coefficients.

5.1.6. Estimation

Maximum likelihood techniques are used to estimate the unknown parameters of the integrated model. The model estimation process maximizes the logarithm of the sample likelihood function over the unknown parameters:

$$[5\text{-}34] \quad \max_{\theta,\tau,\tau^{SP},\delta,\chi,\mu,\gamma,\lambda,\alpha,\Sigma} \sum_{n=1}^{N} \ln\left(P_n(y_n, I_n \mid X_n, W_n, Z_n; \theta, \tau, \tau^{SP}, \delta, \chi, \mu^{SP}, \gamma, \lambda, \alpha, \Sigma)\right)$$

The integral in the probability like the one given in Equation [5-32] includes complicated multi-dimensional integrals, with dimensionality equal to that of the integral of the underlying choice model plus the number of latent variables. This integral generally does not have a closed formed, and so it is approximated through a numerical integration or simulation. As the dimensionality of the integral increases, the numerical integration methods quickly become computationally infeasible and simulation methods must be employed (see [Walker, 2001]). [Bolduc, 2003] states that numerical integration techniques in situations involving more than three dimensions is too demanding in computing time and not accurate enough if approximations are used.

In this research, we will use the methods of maximum simulated likelihood which employ random draws of the random coefficients and latent variables from their probability distribution. [Train, 2003] provides a comprehensive methodology review of the use of simulation techniques to estimate behavioral models. In particular, *D* draws of β, ν and X^* are respectively taken from the density $f(\beta)$, $f(\nu)$ and $f(X^*)$. For each draw, the product of logit in Equation [5-32] is calculated, and the results are averaged over draws. The parameters are estimated by maximizing the simulated likelihood over the unknown parameters. A more detailed description of the simulation and maximization techniques will follow in the problem-specific context of our student model framework.

5.2. Student Model Framework

The model uses data collected in the final survey presented in the previous chapter and from rating found in the Underground Guide's evaluation forms. Figure 5-3 provides a full path diagram of the model for a given RP observation, noting the relationships among variables. Figure 5-4 does the same for a given SP observation.

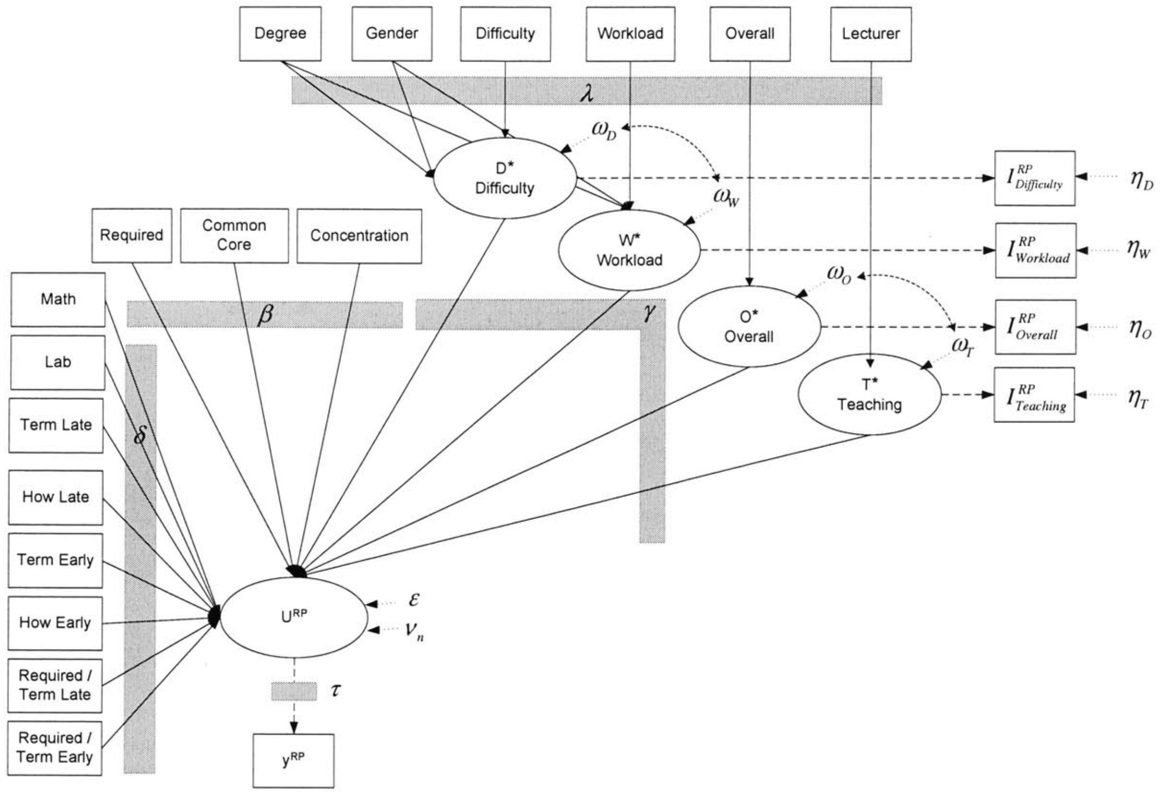


Figure 5-3. Student Model for a Given RP Observation

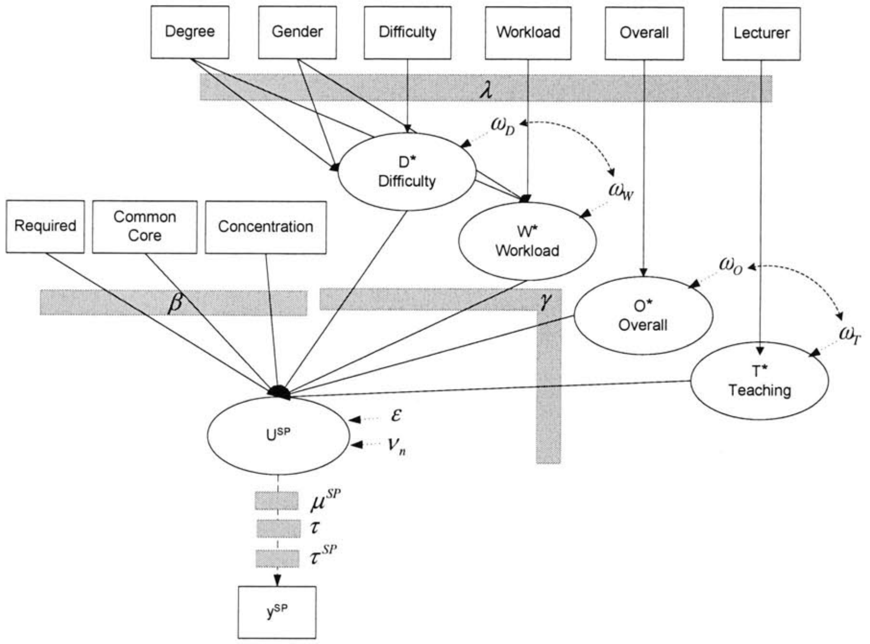


Figure 5-4. Student Model for a Given SP Observation

The behavioral hypothesis is that students rate how recommendable a course is based on attributes and characteristics relating to their demographics and to the RP and SP situations.

5.2.1. Latent Attributes

For a given survey respondent, ratings for teaching, overall impression, difficulty and amount of workload required is obtained from two different sources: (i) average ratings coming from the evaluation forms of 'The Underground Guide'; (ii) individual ratings which are collected by the online survey.

Typically, ratings from these two sources differ since the former one represents an aggregate measure of the impression of an entire class, while the later one is just one element of this aggregate measure. Moreover, it was found the averages of the personal ratings for a given class in a given term differed from the averages given by the evaluation forms for the same class (bias not only exists on an individual level, but is also found on an aggregate level). This difference in averages can be explained by two reasons: the survey takers are a subset of the overall students who gave their ratings in the evaluation forms; the survey takers rated the attributes differently than when they rated the same ones in the evaluation forms.

The model is ultimately intended to predict the level of recommendation for courses that have not already been taken. Under this circumstance, individual ratings for course attributes are unknown for future classes. Only ratings from the evaluation forms can be explicitly included in making this kind of prediction. On the other hand, individual ratings for courses already taken are useful for estimating the bias of a given survey taker with respect to the average ratings coming from the evaluation forms. In terms of modeling, teaching, overall impression, difficulty and workload were identified as four different latent variables with average ratings being used as observable variables and individual ratings being used as indicators of those four latent variables. Note that a given respondent gives both RP and SP data. Figure 5-3 shows RP observations, and Figure 5-4 shows SP observations (typically a given respondent would have many RP and many SP observations). Since SP data are not actual courses that have been taken and hence don't have ratings from evaluation forms, there are no indicators for the SP model as shown in Figure 5-4.

The timing variables are RP specific as shown on Figure 5-3. The two other RP specific factors are dummy variables indicating whether a course is classified as an EECS math course (6.041 and 6.042) or an EECS lab (from 6.100 through 6.182). The "Common Core" included in the RP and SP model is a dummy variable indicating whether a course belongs to the 4 EECS core courses (6.001, 6.002, 6.003, and 6.004) that all EECS students have to take independently of the degree they seek.

A different scaling factor is calculated for the RP and SP data (μ^{RP} is normalized to 1), resulting in:

$$Variance(\varepsilon^{RP}) = (\mu^{SP})^2 Variance(\varepsilon^{SP}).$$

Finally an SP specific constant is estimated resulting in a shifted threshold τ^{SP} with respect to the RP thresholds τ^{RP}.

The preliminary studies and analysis of the survey data shows a high correlation between the teaching and overall rating attributes on one hand and between the difficulty and workload attributes on the other hand. Based on these findings, the covariances of the error terms between ω_T and ω_O, and between ω_D and ω_W are included in the model estimation (shown as a two way arrow in Figure 5-3 and Figure 5-4).

5.2.2. Intermediate Models

The joint model presented in both Figure 5-3 and Figure 5-4 was reached after improving on simpler models that didn't include latent variables. One of those basic models and its improvement by including the latent variables are presented in Appendix E. The inclusion of latent variables turned out to significantly improve the goodness of fit of the model as shown when comparing the rho-bar-squared[11] in Table E.1 and Table E.2. Moreover, the final model only incorporated parameters (coefficients or variances) that

[11] $\bar{\rho}^2$ is an index similar to ρ^2, but adjusted for number of parameters estimated by the model. In general, we should prefer models that are parsimonious, i.e., where we do not include variables that contribute very little to explaining the observed choices. ρ^2 in is an index of incremental explanation provided by the selected model in comparison to a naive model in which all parameters are set to 0.

turned out to be statistically significant. The estimation results for the final model are presented at the end of this chapter in section 5.4.

5.3. Student Model Estimation

5.3.1. Likelihood Function

The general formulation of the likelihood function given in Equation [5-32], when applied to the joint model given in Figure 5-3 and Figure 5-4, should incorporate the following simplifications:

- The model has four latent variables (indexed *l=1,..,4*) with one indicator for each one of them. Since we have one indicator for every latent variable, all the α coefficients were set to *1* for identification purposes. Moreover the indicator index *m* is replaced by *l*. Indicators are only available in RP situations, hence dropping the measurement equation for the SP model.
- The coefficients β were estimated as being fixed (as opposed to having β_n and a distribution of $f(\beta)$ as in Equation [5-32]).
- ν_n^{RP} and ν_n^{SP} were estimated as having the same distribution $\nu_n \sim N(0,\sigma_\nu^2)$.
- The timing variables and the "Math" and "Lab" dummy variables are only present in the RP setting making them RP only variables. Besides τ^{SP}, there are no SP only variables (no χ to estimate).

Latent Variables

Equation [5-18] defined the structural equation for latent variables as:

$$X_{lnt}^* = X_{lnt}\lambda_l + F_l\Gamma\zeta_{nt}.$$

The model shown in Figure 5-3 and Figure 5-4 contains four latent variables with a covariance matrix Σ_ω given by:

$$\Sigma_\omega = \begin{bmatrix} \sigma_{11} & \sigma_{12} & 0 & 0 \\ \sigma_{21} & \sigma_{22} & 0 & 0 \\ 0 & 0 & \sigma_{33} & \sigma_{34} \\ 0 & 0 & \sigma_{43} & \sigma_{44} \end{bmatrix} \text{, where}$$

Index l	*Latent Variable*
1	Teaching
2	Overall
3	Difficulty
4	Workload

and $\sigma_{12} = \sigma_{21}$, and $\sigma_{34} = \sigma_{43}$. The restricted form of Σ_ω was explained in section 5.2.1 where it was discussed that the preliminary studies and analysis of the survey data only shows a high correlation between the teaching and overall rating attributes on one hand and between the difficulty and workload attributes on the other hand. In this case, the parameters of the Cholesky decomposition of Σ_ω are:

$$\Gamma = \begin{bmatrix} \Gamma_{11} & 0 & 0 & 0 \\ \Gamma_{21} & \Gamma_{22} & 0 & 0 \\ 0 & 0 & \Gamma_{33} & 0 \\ 0 & 0 & \Gamma_{43} & \Gamma_{44} \end{bmatrix} \text{, where } \Gamma\Gamma' = \Sigma_\omega$$

ζ_{nt} is still defined as an $(L \times 1)$ vector of standard independent normally distributed factors:

$$\zeta_{nt} = \begin{bmatrix} \zeta_{1nt} \\ \zeta_{2nt} \\ \zeta_{3nt} \\ \zeta_{4nt} \end{bmatrix} \text{, where } \zeta_{lnt} \sim N(0,1)$$

and F_l is the l^{th} row of an $(L \times L)$ identity matrix. Taking a concrete example of l=2 ("Overall" latent variable), the structural equation given by $X^*_{lnt} = X_{lnt}\lambda_l + F_l\Gamma\zeta_{nt}$ becomes:

$$X_{2nt}^{*} = X_{2nt}\lambda_2 + F_2\Gamma\zeta_{nt} = X_{2nt}\lambda_2 + \begin{bmatrix}0 & 1 & 0 & 0\end{bmatrix}\begin{bmatrix}\Gamma_{11} & 0 & 0 & 0\\ \Gamma_{21} & \Gamma_{22} & 0 & 0\\ 0 & 0 & \Gamma_{33} & 0\\ 0 & 0 & \Gamma_{43} & \Gamma_{44}\end{bmatrix}\begin{bmatrix}\zeta_{1nt}\\ \zeta_{2nt}\\ \zeta_{3nt}\\ \zeta_{4nt}\end{bmatrix}$$

$$= X_{2nt}\lambda_2 + \begin{bmatrix}0 & 1 & 0 & 0\end{bmatrix}\begin{bmatrix}\Gamma_{11}\zeta_{1nt}\\ \Gamma_{21}\zeta_{1nt} + \Gamma_{22}\zeta_{2nt}\\ \Gamma_{33}\zeta_{3nt}\\ \Gamma_{43}\zeta_{3nt} + \Gamma_{44}\zeta_{4nt}\end{bmatrix}$$

$$= X_{2nt}\lambda_2 + \Gamma_{21}\zeta_{1nt} + \Gamma_{22}\zeta_{2nt}$$

Having defined our latent variable structural equations for our model, we can use the utility of the choice model given by Equation [5-22] for $L = 4$:

$$U_{nt} = X_{nt}\beta_n + \sum_{l=1}^{4}(X_{lnt}\lambda_l + F_l\Gamma\zeta_{nt})\gamma_l + \sigma_v\,\tilde{v}_n + \varepsilon_{nt},$$

When adapted to the model given in Figure 5-3, the RP choice model given in Equation [5-26] for a given observation r (remember that the subscript t is replaced by r for RP observations and s for SP observations) becomes:

$$P_{nr}^{RP}\left(y_{nr}^{RP} \mid X_{nr}, X_{nr}^{*}, W_n, \tilde{v}_n; \beta, \tau, \delta, \gamma, \sigma_v\right)$$

[5-35]
$$= \prod_{i=1}^{7}\left(\frac{1}{1+e^{(X_{nr}\beta + W_{nr}\delta + \sum_{l=1}^{4}(X_{lnr}\lambda_l + F_l\Gamma\zeta_{nr})\gamma_l + \sigma_v\tilde{v}_n - \tau_i)}} - \frac{1}{1+e^{(X_{nr}\beta + W_{nr}\delta + \sum_{l=1}^{4}(X_{lnr}\lambda_l + F_l\Gamma\zeta_{nr})\gamma_l + \sigma_v\tilde{v}_n - \tau_{i-1})}}\right)^{y_{inr}^{RP}}$$

Introducing indicators and integrating over the latent variables leads to the following RP likelihood:

$$P_n^{RP}\left(y_n^{RP}, I_n^{RP} \mid X_n, W_n, \tilde{v}_n; \beta, \tau, \delta, \gamma, \sigma_v, \lambda, \Sigma_\omega, \Sigma_\eta\right)$$

[5-36]
$$= \prod_{r=1}^{R} \iint\iint P_{nr}^{RP}\left(y_{nr}^{RP} \mid X_{nr}, \zeta_{nr}^{RP}, W_n, \tilde{v}_n; \beta, \tau, \delta, \gamma, \sigma_v, \lambda, \Sigma_\omega\right) \times \prod_{l=1}^{4}\frac{1}{\sigma_{\eta_l}}\phi\left(\frac{I_{lnr}^{RP} - \left(X_{lnr}\lambda_l + F_l\Gamma\zeta_{nr}^{RP}\right)}{\sigma_{\eta_l}}\right) \times \prod_{l=1}^{4}\phi(\zeta_{lnr}^{RP})\,d\zeta$$

Following the same steps and noting that there are no indicators for the SP situations, we obtain the following SP likelihood equation for the model given in Figure 5-4:

[5-37]
$$P_n^{SP}\left(y_n^{SP} \mid X_n, \tilde{v}_n; \beta, \tau, \tau^{SP}, \gamma, \sigma_v, \mu^{SP}, \lambda, \Sigma_\omega\right) = \prod_{s=1}^{S} \iint \iint P_{ns}^{SP}\left(y_{ns}^{SP} \mid X_{ns}, \zeta_{ns}^{SP}, W_n, \tilde{v}_n; \beta, \tau, \delta, \gamma, \sigma_v, \lambda, \Sigma_\omega\right) \times \prod_{l=1}^{4} \phi(\zeta_{lns}^{SP})\, d\zeta$$

The last step is to combine both the RP and SP model and integrate over the panel data so that the likelihood function for the model can be written as:

[5-38]
$$\begin{aligned} &P\left(y^{RP}, y^{SP}, I^{RP} \mid X, W; \beta, \tau, \tau^{SP}, \delta, \mu^{SP}, \gamma, \lambda, \sigma_v, \Sigma_\omega, \Sigma_\eta\right) \\ &= \prod_{n=1}^{N} \Big\{ \int P_n^{RP}\left(y_n^{RP}, I_n^{RP} \mid X_n, W_n, \tilde{v}; \beta, \tau, \delta, \gamma, \sigma_v, \lambda, \Sigma_\omega, \Sigma_\eta\right) \\ &\quad \times P_n^{SP}\left(y_n^{SP} \mid X_n, \tilde{v}; \beta, \tau, \tau^{SP}, \gamma, \sigma_v, \mu^{SP}, \lambda, \Sigma_\omega\right) \times \phi(\tilde{v})\, d\tilde{v} \Big\} \end{aligned}$$

5.3.2. Maximum Simulated Likelihood

Each factor ζ and $\tilde{v}$ introduces a dimension to the integral, making the dimension of our likelihood function to be 5. We use the Maximum Simulated Likelihood (MSL) methodology to estimate the unknown parameters of the integrated model. The simulation procedure will include drawing from the distribution of ζ to simulate the latent variables, and drawing from the distribution of $\tilde{v}$ to simulate the panel data effect.

Latent Variables

Simulation is performed by taking *D* random draws from the distributions of ζ_{1nt}, ζ_{2nt}, ζ_{3nt} and ζ_{4nt} for each observation in the sample (for each RP and each SP observations hence using the subscript *t*), denoted $\zeta_{1nt}^d, \zeta_{2nt}^d, \zeta_{3nt}^d$ and ζ_{4nt}^d, *d=1,...,D*. The structural equation for latent variable *l* is then written as:

$$X_{lnt}^{*d} = X_{lnt}\lambda_l + F_l T \zeta_{nt}^d, \text{ where } \zeta_{nt}^d = \begin{bmatrix} \zeta_{1nt}^d \\ \zeta_{2nt}^d \\ \zeta_{3nt}^d \\ \zeta_{4nt}^d \end{bmatrix}$$

The simulated structural equation is then replaced in the choice and measurement model.

Panel Data

Simulation is performed by taking random draws from the distribution $\tilde{\nu}_n$ for each respondent in the sample denoted $\tilde{\nu}_n^d$, *d=1,...,D*. Note that only one draw is taken for all the observations coming from a given respondent *n* as opposed to taking one draw for every observation as was the case for the latent variables. By using this procedure, we ensure that the panel effect across observations given by a respondent *n* is taken into consideration. The utility of the choice model given by Equation [5-22] is rewritten as:

$$U_{nt}^d = X_{nt}\beta + \sum_{l=1}^{4}\left(X_{lnt}\lambda_l + F_l\Gamma\zeta_{nt}^d\right)\gamma_l + \sigma_\nu\tilde{\nu}_n^d + \varepsilon_{nt}$$

and the overall likelihood function given in Equation [5-38] for a given draw *d* is written as:

$$P^d\left(y^{RP}, y^{SP}, I^{RP}, | X, W, \zeta^d, \tilde{\nu}^d; \beta, \tau, \tau^{SP}, \delta, \mu^{SP}, \gamma, \lambda, \sigma_\nu, \Sigma_\omega, \Sigma_\eta\right)$$

Simulated likelihood

The following is then an unbiased simulator for Equation [5-38]:

[5-39]
$$\begin{aligned}&\tilde{P}\left(y^{RP}, y^{SP}, I^{RP}, | X, W; \beta, \tau, \tau^{SP}, \delta, \mu^{SP}, \gamma, \lambda, \sigma_\nu, \Sigma_\omega, \Sigma_\eta\right)\\ &\quad = \frac{1}{D}\sum_{d=1}^{D} P^d\left(y^{RP}, y^{SP}, I^{RP}, | X, W, \zeta^d, \tilde{\nu}^d; \beta, \tau, \tau^{SP}, \delta, \mu^{SP}, \gamma, \lambda, \sigma_\nu, \Sigma_\omega, \Sigma_\eta\right)\end{aligned}$$

The parameters are estimated by maximizing the simulated likelihood over the unknown parameters. There has been a lot of research concerning how to best generate the set of random points (see [Bhat, 2000] for a summary and references). Numerous procedures have been proposed in the numerical analysis literature for taking "intelligent" draws from a distribution rather than random ones. The procedures offer the potential to reduce the number of draws that are needed for mixed logit estimation, thereby reducing run times, and/or to reduce the simulation error that is associated with a given number of draws. Using Halton sequences for mixed logit estimation is one such procedure that we used to simulate our likelihood function. For more information on Halton draws see [Bhat, 2000] and [Train, 1999].

5.3.3. Estimation with STATA

There are many existing software programs that deal with estimating advanced discrete choice models. BIOGEME (http://roso.epfl.ch/biogeme) is one software package designed for the maximum likelihood estimation of Generalized Extreme Value Models. BIOGEME, however, cannot estimate ordered models. STATA (http://www.stata.com/) and SAS (http://www.sas.com/) are statistical packages that have routines for estimating regressions and discrete choice models, but they don't include prepackaged routines to estimate such a complex model as ours. GLLAMM (http://www.gllamm.org/) is a program that runs in STATA and estimates "Generalized Linear Latent And Mixed Models". This program can estimate models with latent variables, random coefficients, and ordered data. The program, however, uses numerical integration to estimate the likelihood function. This makes it unusable to estimate models as the one given in Equation [5-38].

STATA is extensible and can be programmed to compute user-defined maximum likelihood estimators. We developed a STATA routine to specifically estimate the likelihood equation given in Equation [5-38]. The routine applies the simulated maximum likelihood method using Halton sequences as described in the previous section. The code for the algorithm is included in Appendix F. Details on how the numerical maximization is performed in STATA can be found in [Gould et al., 2003]. Appendix F contains the code for the following two core STATA files:

combined_model.do

This file loads the data and launches the program by calling other routines. The data is loaded by first reading the survey database file described in section 4.2.3. It then loads the Halton sequence that the simulation uses. We wrote a Java program that generates 5000 draws for as many Halton sequences as needed and stores the sequences in a text file. The STATA program reads this created text file to load the Halton Sequences. Other routines that are called are:

- *data_corrections.do* eliminates observations coming from students with no RP data and corrects the ratings for two courses.

- *rp_data.do, sp_data.do, demographics.do* setup the variables for the RP, SP and demographic data respectively.
- *interactions.do* creates interaction variables such as a match between subject concentration and student's interest in the concentration.
- *combined.do* combines the data from the RP and SP experiments
- *ologit_halton.do* is the main routine that simulates the likelihood function

ologit_halton.do

This file implements the core engine of the estimation. The model is specified in the *combined_model.do* file when calling *ologit_halton.do*. The *ologit_halton.do* routine uses the *ml* command that allows STATA to fit user defined likelihood function. Details on how to use and program the *ml* command can be found in [Gould et al., 2003].

5.3.4. Identification

Identification of this complex model is difficult. Applying the sufficient, but not necessary technique of conditionally identifying each sub-module is not enough to be certain that the model is fully identified. We used a set of recommended empirical identification indicators to mitigate this uncertainty (see [Walker, 2001]):

- The parameters converged to the same point and likelihood when given different starting values.
- STATA automatically verifies that the Hessian of the log-likelihood is non-singular (a test of local identification). STATA did not issue a "Hessian warning" when it started crawling toward a solution.
- The parameters were stable as the number of simulation draws increased.

Moreover, STATA's maximum likelihood estimation allows the monitoring of convergence. The later iterations of maximizing the likelihood of this model followed the expected pattern of moving smoothly toward a maximum, taking smaller and smaller steps until the log likelihood function flattened out.

5.4. Results

There are a total of 43 parameters to be estimated. That includes: β (3 parameters), δ (8 parameters), γ (4 parameters), τ (6 parameters), τ^{SP} (1 parameter), μ^{SP} (1 parameter), σ_v (1 parameter), λ (9 parameters), Σ_ω (6 parameters including 2 off diagonal elements), and Σ_η *diagonal* (4 parameters). The results are shown in Table 5-2. The dataset included 1866 RP observations and 1380 SP observations for a total of 3246 observations. Those observations were obtained from 314 respondents. Simulation was performed using 5000 Halton draws.

This model demonstrates the impact of different course and student characteristics on students' course ratings. Most of the hypothesis and findings in previous studies were confirmed. Looking at the choice model (top panel):

- The teaching and overall latent variables both have a significant positive effect on students' ratings.
- Difficult courses are perceived as being less recommendable.
- Workload, surprisingly, turned out to have a positive coefficient. One would expect that a higher workload for a given course would lead to a lower rating (and hence one would expect a negative coefficient), but results showed otherwise. This can be explained by the fact that course workload is acting as a proxy to the quantity of information or usefulness of the course, and hence makes the rating more "recommendable".
- The more the content of a course covers the areas of interests, or "Concentrations", the more recommendable it is.
- Required courses have a positive effect on rating since required courses are defined depending on the chosen degree and concentration, and hence are relevant to a student's interest.

- Common core courses, which are requirements for all students, had a positive effect as well. They are perceived as being essential and useful (and hence should be recommended to other students).
- Math and Department Lab courses have a significant positive effect on students' ratings. A note should be made that once these variables, which are RP specific, were included, the SP constant τ^{SP} became statistically insignificant (as opposed to be significant with a value of -0.49 in Table E.2). Hence, students particularly enjoy math and lab courses. Since these factors are not included in SP situations, student on average rated SP courses 0.49 points lower than RP courses when those factors were not taken into consideration in RP situation (as was the case in the model presented in Table E.2).
- As expected, higher ratings of the prerequisites of a given course lead to a higher rating for that course.
- As far as timing is concerned:

- an elective course that is taken in a later term than its expected average term tends to have a positive effect on rating, but then this effect decreases proportionally to how late the course is actually taken. In other words: a given elective course taken on a later term becomes more enjoyable as students would have the skills and knowledge maturity to enjoy the course more thoroughly. But taking the course "too late" is not as recommendable as other courses might depend on it.
- An elective course that is taken earlier than expected tends to get a higher rating. Those courses can be seen as being particularly relevant to a certain interest area or concentration, and hence are recommended to be taken as early as possible.
- Finally a required course not taken when it is expected to be has a negative effect on rating.

Looking at the Latent Variable model (middle panel), the estimated covariance between teaching and overall rating on one hand, and between workload and difficulty on the other hand turned to be statistically significant. Results indicate that gender and type of degree tend to have a significant effect on how workload and difficulty are perceived.

Female students tend to rate the workload of a given course higher than male students would. EE[12] and EECS[13] students tend to rate workload and difficulty of a given course higher than CS[14] students would.

The model, shown in Table 5-2, concludes the third step of our recommender system framework. As was shown in the result discussion, the model is intuitive, behaviorally realistic and led to insights on which factors are important to students. Attributes that were included in the choice model are either easy to collect or are readily available online for upcoming courses making the model particularly suitable to predict courses that haven't been taken yet. The next chapter uses this model as the backbone of the recommender engine that the software package uses to predict student ratings.

[12] Electrical Engineering
[13] Electrical Engineering and Computer Science
[14] Computer Science

Table 5-2. Estimation Results (5000 Halton Draws)

Choice Model

Course Utility	**Est. $\beta, \delta, \gamma, \tau$**	**t-stat**
T^*: Teaching (Latent)	0.39	8.41
O^*: Overall	0.54	9.98
W^*: Workload	0.37	2.75
D^*: Difficulty	-0.11	-2.33
Concentration	0.20	7.69
Required	0.24	7.76
Common Core	0.26	-3.94
Term Late	0.94	3.81
How Late	-0.18	-1.95
How Early	0.15	1.98
Required x Term Early	-0.59	-3.69
Required x Term Late	-1.06	-4.72
Mathematics	0.81	4.49
Department Lab	0.79	8.63
Prerequisites	0.06	3.13
τ^{SP} (SP constant)	-0.11	-0.67
μ^{SP} (SP constant)	0.60	25.25
ν_n (Variance of Panel Data)	0.11	2.24
τ_1	2.28	5.78
τ_2	3.60	9.04
τ_3	4.43	10.97
τ_4	5.34	12.99
τ_5	6.43	15.27
τ_6	7.73	17.82
Rho-bar Squared		0.452

Latent Variable Model

Structural Models		**Est. λ, Σ_ω**	**t-stat**
Teaching	Lecturer	0.93	117.48
	Variance ω_T	0.31	13.66
Overall	Overall	0.95	121.75
	Variance ω_O	0.31	10.88
Covariance ω_{TO}		0.32	15.28
Workload	Workload	0.44	31.84
	Female	0.15	4.82
	Degree EE	0.33	5.81
	Degree EECS	0.46	13.77
	Variance ω_W	0.13	27.86
Difficulty	Difficulty	0.87	56.93
	Degree EE	0.13	1.24
	Degree EECS	0.58	5.50
	Variance ω_D	0.35	11.23
Covariance ω_{WD}		0.16	6.96

Measurement Model		**Est. Σ_η**	**t-stat**
$I_{Lecturer}$	Variance η_T	1.35	59.69
$I_{Overall}$	Variance η_O	1.28	60.08
$I_{Workload}$	Variance η_W	0.27	55.31
$I_{Difficulty}$	Variance η_D	1.08	57.31

Chapter 6: Online Academic Advisor

The fourth and final step of the recommender system framework is to implement the choice model estimated in Chapter 5 as part of the software package that delivers recommendations. One of the many modules of this software package hosts this initial model, while other modules are targeted at collecting data, updating the model, calculating predictions and presenting recommendations. The chapter focuses on describing this modular architecture by explaining the functionalities and mechanisms of the different implemented modules in the specific case of our academic advising application. The chapter also highlights the software package's adaptability and extensibility to other applications.

6.1. Components of the Online Academic Advisor

Chapter 2 presented the goals and components of a successful recommender system.

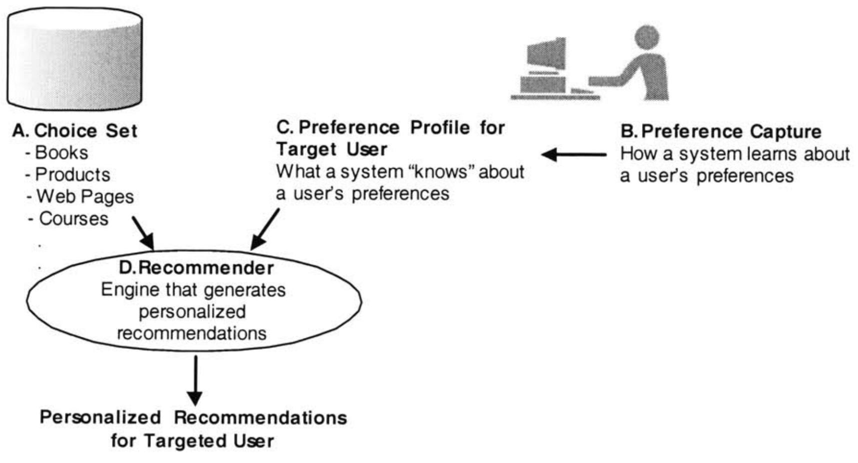
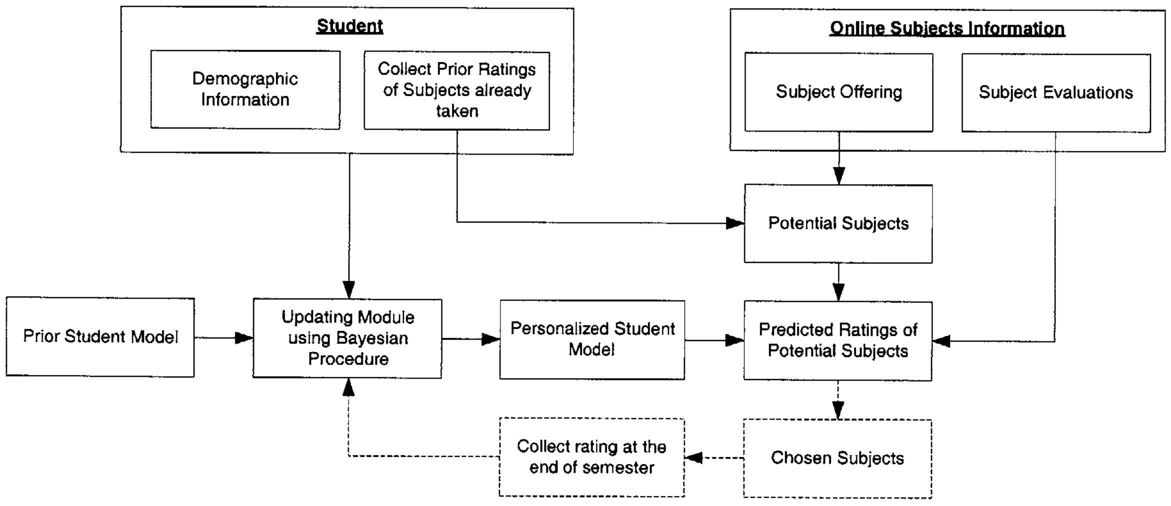

A. Choice Set
- Books
- Products
- Web Pages
- Courses
C. Preference Profile for Target User
What a system "knows" about a user's preferences
B. Preference Capture
How a system learns about a user's preferences
D. Recommender
Engine that generates personalized recommendations
Personalized Recommendations for Targeted User

Figure 6-1. Components of a Recommender System ([ChoiceStream])

Although different technologies are used to meet those goals, the core of a recommender system is made up of the components presented in Figure 6-1 which forms the basis of our Online Academic Advisor. The next step of this dissertation is to design and implement a software agent that incorporates those core components.

A high-level description of this software agent is best depicted in Figure 6-2. In this framework, The "Online Subjects Information" is a web agent that actively looks into MIT's online database for subject offerings and subject evaluations (component "A" in Figure 6-1). A "Student" web agent with a web interface is responsible for collecting student information, transmitting the data for the updating module, determining what subjects are eligible to be taken from the subject offering list and finally displaying predicted ratings to the student (Component B and C in Figure 6-1). The model given in Chapter 5 corresponds to the "Prior Student Model". This model along with "Updating Module using Bayesian Procedure" corresponds to component "D" in Figure 6-1.



Student
Demographic Information
Collect Prior Ratings of Subjects already taken
Online Subjects Information
Subject Offering
Subject Evaluations
Potential Subjects
Prior Student Model
Updating Module using Bayesian Procedure
Personalized Student Model
Predicted Ratings of Potential Subjects
Collect rating at the end of semester
Chosen Subjects

Figure 6-2. Overview of the Online Academic Advisor

The dashed boxes and lines correspond to a "Bayesian updating module" that actively collects data from students and estimate individual level parameters. The updating procedure occurs at the end of the semester and at the start of a new recommendation cycle. The software agent is implemented using Java[15] and Java Servlet[16] technologies.

[15] A high-level programming language developed by Sun Microsystems (http://java.sun.com/).
[16] A small Java program that runs within a web server environment that delivers (*serves up*) Web pages.

The rest of this chapter will discuss in detail each of the components of the online academic advisor.

6.2. Defining the Choice Set

6.2.1. Online Subjects Information

The module acts as a web crawler that collects subject information from MIT's and Electrical Engineering and Computer Science's websites and constructs a library of subjects that are either offered in the upcoming academic year or that have already been given in previous terms. Future subjects are collected from MIT's course catalog for the current year. Previous subjects along with their evaluation data are collected from the EECS Underground Guide and are classified by semester. Figure 6-3 shows an overview of this process.

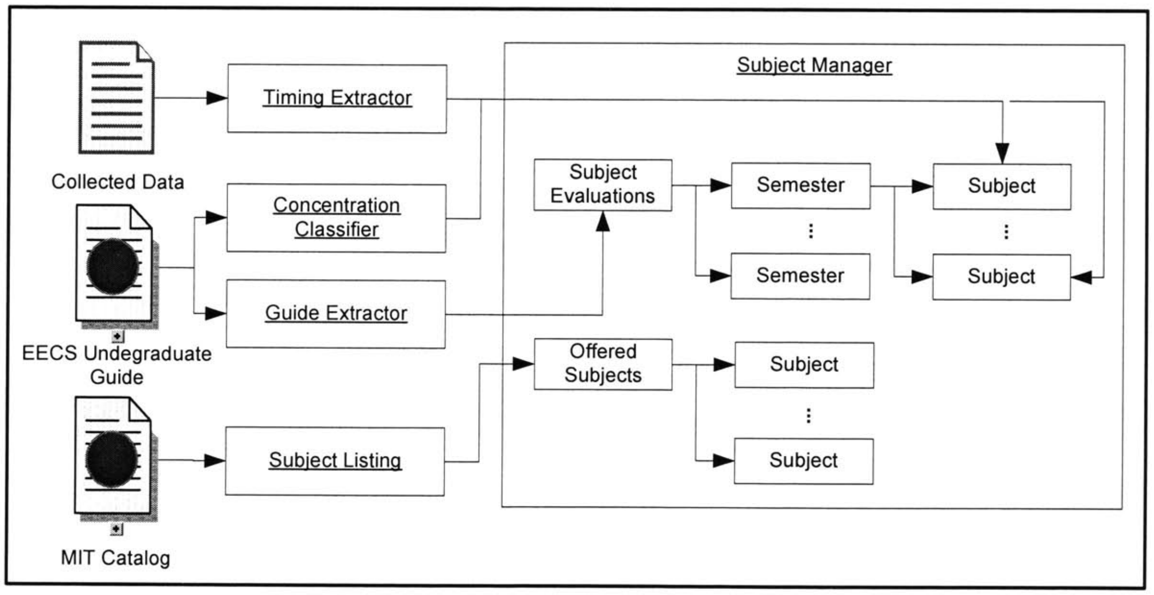


Figure 6-3. Collecting Data

Guide Extractor is a Java class that systematically go through all the terms of the 'Undergraduate Guide' that are available online. For each semester, a database of the subjects offered is constructed and is stored in the *Subject Manager* Java class. The following information is collected for each subject: Lecturers' Ratings, Overall Rating, Difficulty, Workload, Response Rate and the list of Prerequisites.

Subject Listing is a Java class that goes through the MIT catalog for the subjects that are offered for the current academic year and extract the list of subjects along with the term(s) that they are offered.

Concentration Classifier is a Java class that classifies a given subject under at least one of the seven engineering concentrations that the EECS department offers (see Chapter 1 for a description of those concentrations). To perform its task, this Java class uses the "Underground Guide" that gives lists of subjects that falls into each of the seven concentrations.

Timing Extractor is a Java class that uses the survey data (see Chapter 4) to calculate the average timing of a subject relative to the fall semester of a student's sophomore year (which is identified as Term 0). A subject taken in the spring of a student's sophomore year would therefore be identified as Term 1, and so on. The average timing that a given subject is taken is calculated by averaging the timings from all the students for this given subject (see "Chapter 4.2.3 Creating the Database" for more details on defining the timing variable).

The Subject Manager is a key Java class that stores the list of subjects given by the *Guide Extractor* and the *Subject Listing* components, and manages all subject related issues such as:

- Determining the list of potential subjects that may be taken by a given student depending whether the prerequisites for the class are satisfied.
- Calculating subjects attributes that aren't constant for all students. More specifically it determines:
    - Timing variables as given in "Chapter 4.2.3" (e.g. student is taking the subject in a later term than the average term when it's usually taken).

- Whether a given subject is a requirement for a given student (this would vary depending on the degree the student is pursuing[17]).
- The values of the concentration attribute. Students are asked to indicate on a scale of 1 to 4 the importance, as far as their interests are concerned, of each of the seven engineering concentrations that the EECS department offers. This indicator is used to quantify the relevancy of a given subject depending on the concentration it belongs to. The *Subject Manager* uses this information to associate a given subject with the relevancy indicator of the corresponding concentration for a given student. Take as an example of a student who rated the "Artificial Intelligence" concentration as 4 ("Essential") and the "Theoretical Computer Science" concentration as 2 ("Somewhat Important"). A course that falls under both of these concentrations would get a compounded concentration value of 6 for this student.

6.2.2. Potential Subjects

Defining the set of potential subjects to be taken for a given student is a task that heavily involves the *Subject Manager* Java class. Figure 6-4 depicts the process of defining the choice set (i.e. the list of subjects that a student can take in the upcoming semester) for a given student.

Given an instance of the *Student* Java class (i.e. one particular student), the first step in defining the list of potential subjects that this student can take is to go through all the subjects that are offered the next semester (*Subject Offered* in the *Subject Manager* Java class) and check whether the student has already taken the prerequisite subjects by comparing them with the list of subject that the student has already took (*Subjects Taken* in the *Student* Java class).

In case the prerequisites of a potential subject are satisfied, the next step is to describe the subject using the evaluation forms (setting values for the teaching, overall, difficulty, and workload ratings). This task is handled by the *Evaluation Data for Potential Subjects* method. As a reminder, the potential subject in question, although given in past

[17] The degree that a student is pursing is matched with the list of the required classes for that degree. See Chapter 4.2.3 for more details.

semesters, is a future subject that has not been given yet and hence doesn't have evaluation data from students taking it (e.g. 6.001 in Spring 2004 had an overall rating of 5.5; if the system is trying to recommend this subject for the upcoming Fall 2004, what would be the overall rating be?). The *Subject Manager* has a database of the evaluation data for all EECS subjects that have been given in the last 10 years. These historical data can be used to estimate the value of the evaluation data for upcoming potential subjects. There are many ways of using this historical data. One example would be to average the subject evaluation data for the given course over a certain number of recent semesters.

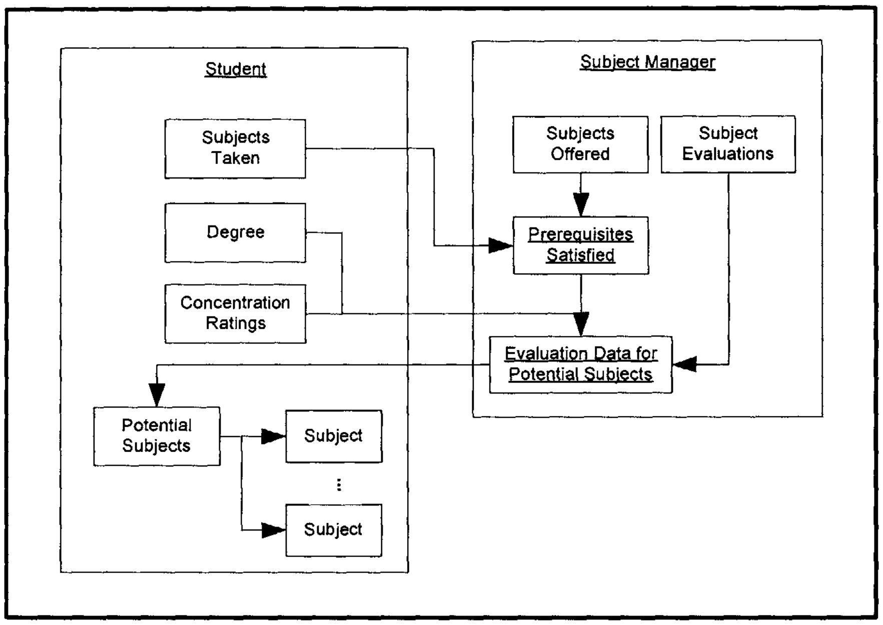


Figure 6-4. Potential Subjects

We used a much simpler technique where the evaluation data of the term that the subject was most recently given is used (in the case of our previous example, 6.001 for the Fall 2004 semester would have an overall rating of 5.5). We chose this technique based on the assumption that the core material of a subject is usually fairly stable over time; the way the material is delivered, however, significantly varies. In that case, the most recent version of the subject would be the most accurate one to describe the same subject that will be given in the upcoming semester.

6.3. Creating a Student Profile

How a system learns about a user's preference and what the system knows about the user's preferences is a key component to any personalization system (see Figure 6-1 components "B" and "C"). Our online academic advisor tackles this task by explicitly collecting student's information (demographics such as degree, concentration interest, etc.), collecting ratings of courses already taken, and using stated preference data to collect more observations. The task of creating a profile for a given students is divided into two parts:

- Instantiating a student object from the *Student* Java class
- Collecting data for that particular student using the *Student Controller* Java class that is responsible for displaying and collecting data.

Figure 6-5 shows an overview of the different Java classes involved in creating a student profile and collecting data.

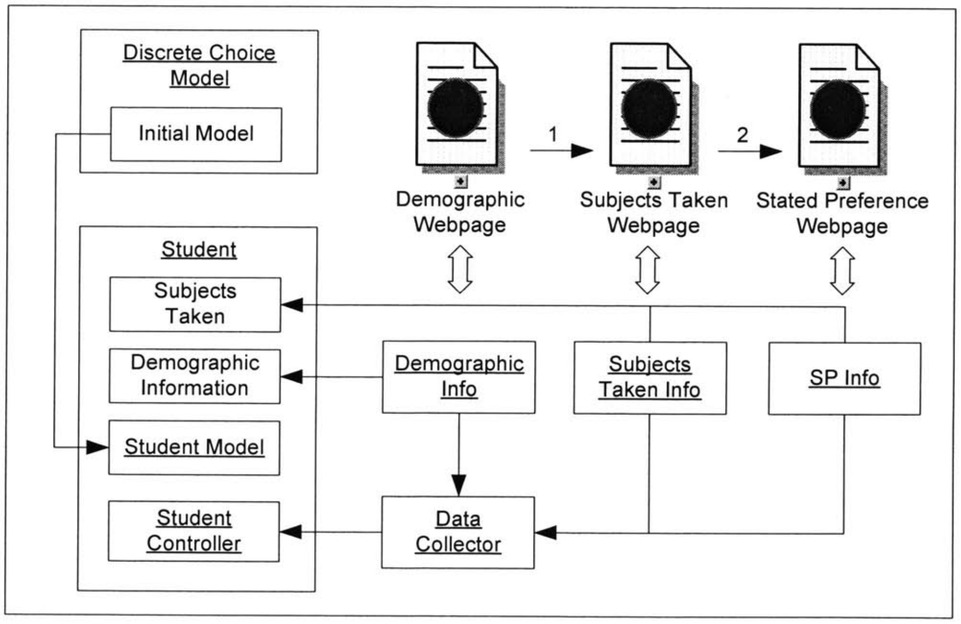
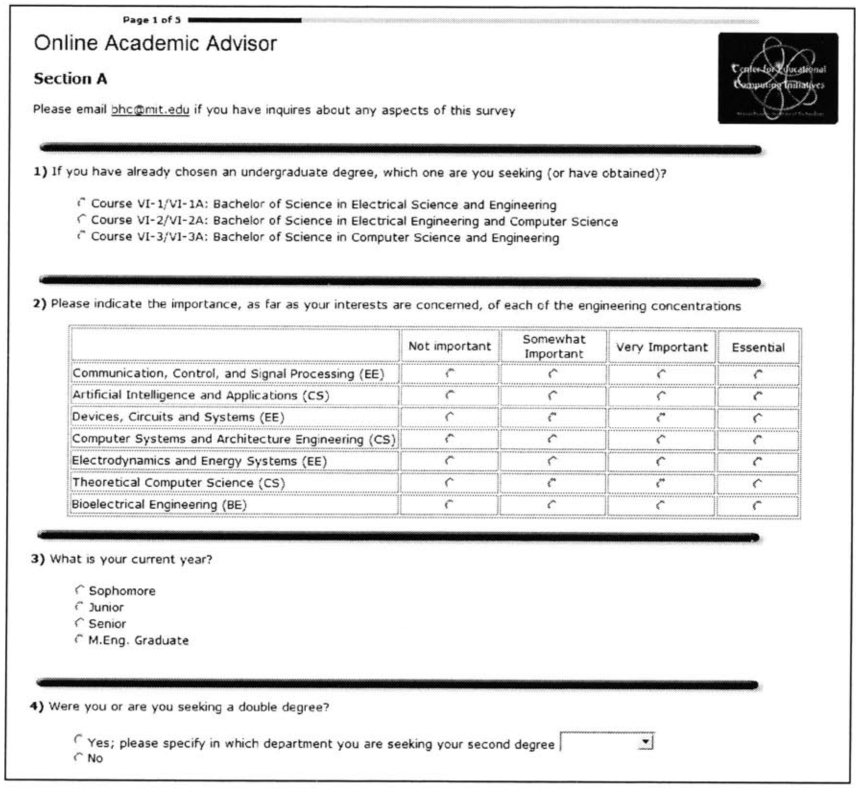


Figure 6-5. Creating a Student Profile

When a student log-in to the system for the first time, a student object is instantiated. It will hold the student's profile, information and model. The student model is the discrete choice model that was constructed and estimated in Chapter 5. A more detailed description of this model will follow in the next section, but for now it would useful to mention that the student model is initialized to the average model that was estimated in Chapter 5. The next step is to collect student information. The *Student Controller* Java Servlet class is responsible for collecting data through an abstract Java class called *Data Collector*. The *Data Collector* is either extended as a *Demographic info*, *Subjects Taken info* or *SP info* Java Servlets depending on the information the *Student Controller* is collecting. The *Data Collector* uses polymorphism[18] to display different webpages (using JSP[19] technology) and collect data depending on the class that is extending it. The first step, as shown in Figure 6-5, is to collect basic demographic information (a snapshot of the webpage is shown in Figure 6-6). The second step is to collect data on subjects that have already been taken by the student (as shown in Figure 6-7).


Page 1 of 5

Online Academic Advisor

Section A

Please email bhc@mit.edu if you have inquires about any aspects of this survey

1) If you have already chosen an undergraduate degree, which one are you seeking (or have obtained)?

- Course VI-1/VI-1A: Bachelor of Science in Electrical Science and Engineering
- Course VI-2/VI-2A: Bachelor of Science in Electrical Engineering and Computer Science
- Course VI-3/VI-3A: Bachelor of Science in Computer Science and Engineering

2) Please indicate the importance, as far as your interests are concerned, of each of the engineering concentrations

	Not important	Somewhat Important	Very Important	Essential
Communication, Control, and Signal Processing (EE)	○	○	○	○
Artificial Intelligence and Applications (CS)	○	○	○	○
Devices, Circuits and Systems (EE)	○	○	○	○
Computer Systems and Architecture Engineering (CS)	○	○	○	○
Electrodynamics and Energy Systems (EE)	○	○	○	○
Theoretical Computer Science (CS)	○	○	○	○
Bioelectrical Engineering (BE)	○	○	○	○

3) What is your current year?

- Sophomore
- Junior
- Senior
- M.Eng. Graduate

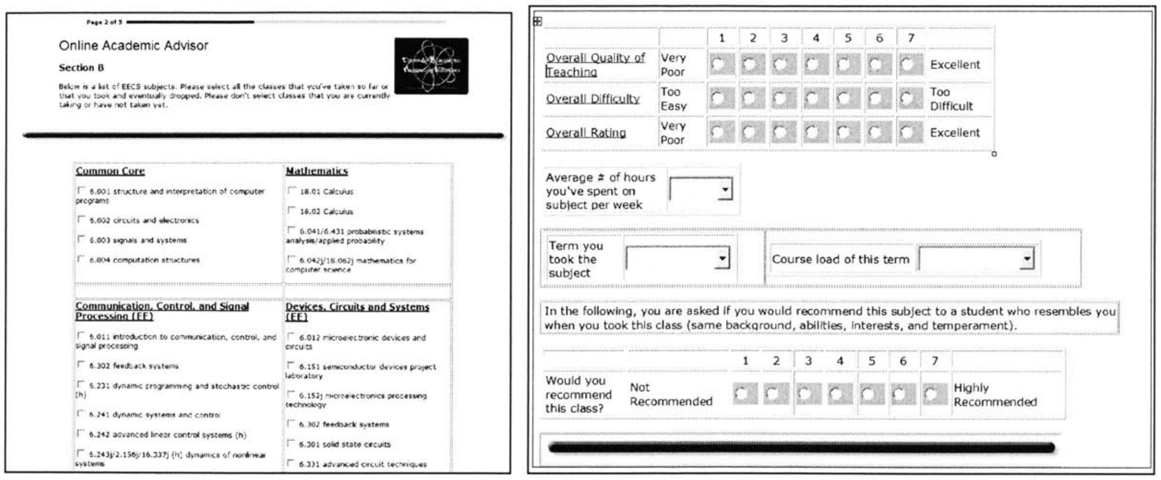
4) Were you or are you seeking a double degree?

- Yes; please specify in which department you are seeking your second degree
- No

Figure 6-6. Demographic Information

[18] In Java, *polymorphism* refers to the ability to process objects differently depending on their data type or class (http://www.webopedia.com/TERM/p/polymorphism.html).

[19] Short for ***Java Server Page***. A server-side technology, Java Server Pages are an extension to the Java servlet technology that was developed by Sun.



Page 2 of 5

Online Academic Advisor

Section B

Below is a list of EECS subjects. Please select all the classes that you've taken so far or that you took and eventually dropped. Please don't select classes that you are currently taking or have not taken yet.

Common Core	**Mathematics**
☐ 6.001 structure and interpretation of computer programs	☐ 18.01 Calculus
☐ 6.002 circuits and electronics	☐ 18.02 Calculus
☐ 6.003 signals and systems	☐ 6.041/6.431 probabilistic systems analysis/applied probability
☐ 6.004 computation structures	☐ 6.042j/18.062j mathematics for computer science
Communication, Control, and Signal Processing (EE)	**Devices, Circuits and Systems (EE)**
☐ 6.011 introduction to communication, control, and signal processing	☐ 6.012 microelectronic devices and circuits
☐ 6.302 feedback systems	☐ 6.151 semiconductor devices project laboratory
☐ 6.231 dynamic programming and stochastic control (h)	☐ 6.152j microelectronics processing technology
☐ 6.241 dynamic systems and control	☐ 6.302 feedback systems
☐ 6.242 advanced linear control systems (h)	☐ 6.301 solid state circuits
☐ 6.243j/2.156j/16.337j (h) dynamics of nonlinear systems	☐ 6.331 advanced circuit techniques

		1	2	3	4	5	6	7	
Overall Quality of Teaching	Very Poor	○	○	○	○	○	○	○	Excellent
Overall Difficulty	Too Easy	○	○	○	○	○	○	○	Too Difficult
Overall Rating	Very Poor	○	○	○	○	○	○	○	Excellent

Average # of hours you've spent on subject per week

Term you took the subject

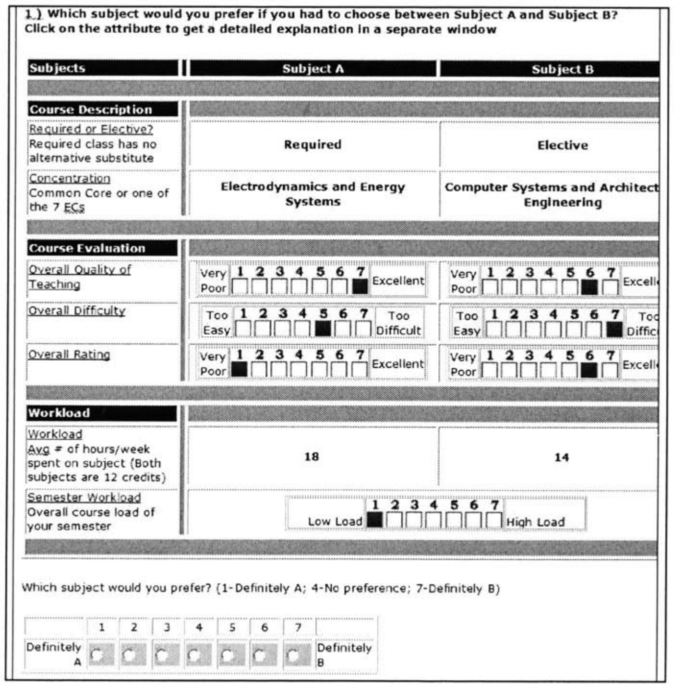
Course load of this term

In the following, you are asked if you would recommend this subject to a student who resembles you when you took this class (same background, abilities, interests, and temperament).

		1	2	3	4	5	6	7	
Would you recommend this class?	Not Recommended	○	○	○	○	○	○	○	Highly Recommended

Figure 6-7. (a) Select Subjects Taken; (b) Rate Selected Subjects

The final step is to complete the data collection by presenting the student with a set of stated preference questions (see Chapter 4 for more information on how the stated preference scenarios were designed).



1.) Which subject would you prefer if you had to choose between Subject A and Subject B?
Click on the attribute to get a detailed explanation in a separate window

Subjects	**Subject A**	**Subject B**
Course Description		
Required or Elective? Required class has no alternative substitute	**Required**	**Elective**
Concentration Common Core or one of the 7 ECs	**Electrodynamics and Energy Systems**	**Computer Systems and Architect Engineering**
Course Evaluation		
Overall Quality of Teaching	Very Poor 1 2 3 4 5 6 [■ 7] Excellent	Very Poor 1 2 3 4 5 [■ 6] 7 Excell
Overall Difficulty	Too Easy 1 2 3 4 [■ 5] 6 7 Too Difficult	Too Easy 1 2 3 4 5 6 [■ 7] Diffic
Overall Rating	Very Poor [■ 1] 2 3 4 5 6 7 Excellent	Very Poor 1 2 3 4 5 [■ 6] 7 Excell
Workload		
Workload Avg # of hours/week spent on subject (Both subjects are 12 credits)	**18**	**14**
Semester Workload Overall course load of your semester	Low Load [■ 1] 2 3 4 5 6 7 High Load	

Which subject would you prefer? (1-Definitely A; 4-No preference; 7-Definitely B)

	1	2	3	4	5	6	7	
Definitely A	○	○	○	○	○	○	○	Definitely B

Figure 6-8. Stated Preference Scenario

The more stated preference questions answered, the more data points the system has for the given student, the better an idea the system will have about the student's preferences.

There is a balance between asking too many questions to increase the accuracy of predictions, and too few questions to minimize the time spent answering the questionnaire. Finding the optimal number of stated preference questions (which will probably vary depending on the student) is a problem that will not be tackled by this dissertation. Based on findings in Chapter 4, five stated preference questions are used by the Online Academic Advisor.

6.4. Recommender Engine

One of the goals of this dissertation is to create a generic framework to use discrete choice models as the core engine for giving recommendations. In order to achieve that, a Java package that contains a set of Java classes that defines the behavior of a discrete choice model was created. The class diagram is shown in Figure 6-9.

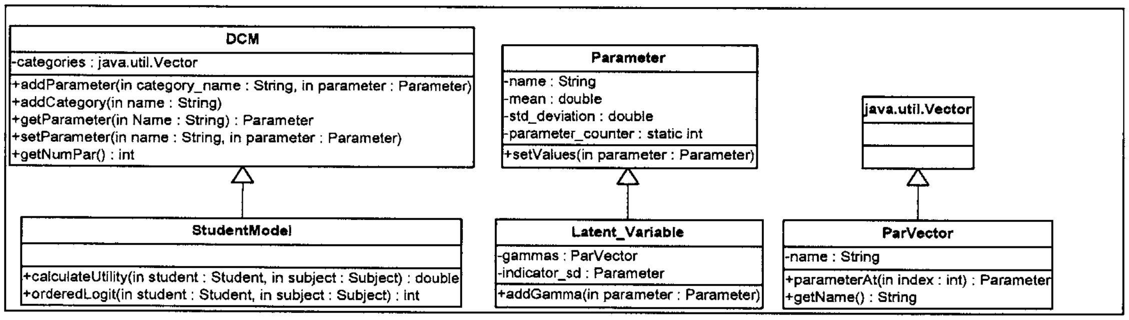

DCM
-categories : java.util.Vector
+addParameter(in category_name : String, in parameter : Parameter)
+addCategory(in name : String)
+getParameter(in Name : String) : Parameter
+setParameter(in name : String, in parameter : Parameter)
+getNumPar() : int
Parameter
-name : String
-mean : double
-std_deviation : double
-parameter_counter : static int
+setValues(in parameter : Parameter)
java.util.Vector
StudentModel
+calculateUtility(in student : Student, in subject : Subject) : double
+orderedLogit(in student : Student, in subject : Subject) : int
Latent_Variable
-gammas : ParVector
-indicator_sd : Parameter
+addGamma(in parameter : Parameter)
ParVector
-name : String
+parameterAt(in index : int) : Parameter
+getName() : String

Figure 6-9. Class Diagram of Discrete Choice Model

Parameter Java class is the building block of this package. This class defines a random coefficient parameter by giving it a name and a distribution (mean and standard deviation). A parameter with a fixed coefficient has its standard deviation set to 0.

ParVector Java class extends the *Vector* Java class and is used to store a collection of parameters. This is particularly useful when storing a set of parameters that act collectively. One such example is the set of explanatory variables for a given latent variable where these variables are used to give an estimate of the latent variable that will enter the utility function (see Chapter 5.1.5). A given parameter can actually belong to several *ParVector* Objects. Reusing our latent variable example, one explanatory variable can be part of the latent variable model as well as directly contributing to the utility function and being part of the choice model (See Figure 5.2).

Latent_Variable Java class extends the *Parameter* class. In addition to the distribution of the latent variable (mean and standard deviation) that are inherited from the *Parameter* Java class, this class contains a *ParVector* (called "gammas") that holds all the explanatory variables of this latent variable parameter.

DCM Java class defines any generalized discrete choice model as defined in Chapter 5. The *categories* is a vector of *ParVector*. Each *ParVector* defines the different parameters categories that might exist in a discrete choice model (e.g. Revealed Preference only variables, Stated Preference only variables, Combined variables, cutoffs for an ordered model, etc.). The *DCM* Java class, given the values of the explanatory variables, can calculate the overall utility of an alternative. By defining a link function (e.g. logit, probit, ordered logit, etc.), the *DCM* calculates the probability that a given alternative is chosen. The *DCM* uses an internal simulator to simulate the utility function when random coefficients are involved. The simulation goes as follow:

1. Take a draw β^d from each random coefficient in the model using its corresponding distribution.
2. Using β^d, calculate the probability P_i^d of choosing alternative *i*.
3. Repeat step 2 for all alternatives *i=1,...,J* where *J* is the total number of alternatives.
4. Repeat steps 1 to 3 many times, with the number of times labeled *D*. The resulting simulated probabilities are calculated as follows: $P_i = \frac{1}{D}\sum_{d=1}^{D} P_i^d$.

Student Model Java class extends *DCM* and is used to make the calculations easier for our Online Academic Advisor problem. It determines how the utility is calculated given a student and a subject. It also defines the link function (mixed ordered logit) that was defined in Chapter 5, and the output of the model. For an ordered logit model, there are several ways we can make a rating prediction. For every cutoff in the ordered model, we calculate the probability of the rating falling within that cutoff point. We can then either:

- Display the rating corresponding to the cutoff with the highest probability
- Display all cutoffs with their corresponding probabilities
- Calculate an expected rating by using: $E = \sum_i iP_i$, where i denotes a cutoff point.

We decided to use the expected rating to give the rating prediction. The *orderedLogit* function in the *Student Model* Java class hence gives the expected rating for a given subject given a student profile. In the context of the student model and using the utility given in section 5.3.2:

$$U_{nt}^d = X_{nt}\beta + W_{nt}\delta + \sum_{l=1}^{4}\left(X_{\ln t}\lambda_l + F_l T\zeta_{nt}^d\right)\gamma_l + \sigma_\nu \tilde{\nu}_n^d + \varepsilon_{nt},$$

P_i^d is calculated as follows:

[6-1]
$$P_i^d\left(i \mid X_{nt}, W_{nt}, \zeta_{nt}^d, \tilde{\nu}_n^d; \beta, \tau, \delta, \gamma, \lambda, \Sigma_\omega, \sigma_\nu\right) = \left(\frac{1}{1+e^{(X_{nt}\beta + W_{nt}\delta + \sum_{l=1}^{4}\left(X_{\ln t}\lambda_l + F_l T\zeta_{nt}^d\right)\gamma_l + \sigma_\nu \tilde{\nu}_n^d - \tau_i)}} - \frac{1}{1+e^{(X_{nt}\beta + W_{nt}\delta + \sum_{l=1}^{4}\left(X_{\ln t}\lambda_l + F_l T\zeta_{nt}^d\right)\gamma_l + \sigma_\nu \tilde{\nu}_n^d - \tau_{i-1})}}\right)$$

This leads to:

[6-2] $$P_i = \frac{1}{D}\sum_{d=1}^{D} P_i^d\left(i \mid X_{nt}, W_{nt}, \zeta_{nt}^d, \tilde{\nu}_n^d; \beta, \tau, \delta, \gamma, \lambda, \Sigma_\omega, \sigma_\nu\right),$$

from which $E = \sum_i iP_i$ can be computed.

6.5. Updating Module: Individual-Level Parameters and Bayesian Procedures

The final piece of our recommender system is the updating module shown in Figure 6-2. To capture the differences in tastes among students, a mixed logit model presented in Chapter 5 was specified. However in order to give personalized recommendations, we need to determine where in the distribution of tastes does a particular student lie.

If we knew nothing about a student's past ratings, then the best we can do in describing his/her tastes is to say that his/her coefficients lie somewhere in $g(\beta|\theta)$ (see Figure 6-10). However, if we have observed that the student gave a rating y when facing situations described by X, then we can use $h(\beta\,|y,X,\theta)$. Since h is tighter than g, we have better information about the student's tastes by conditioning on his/her past choices.

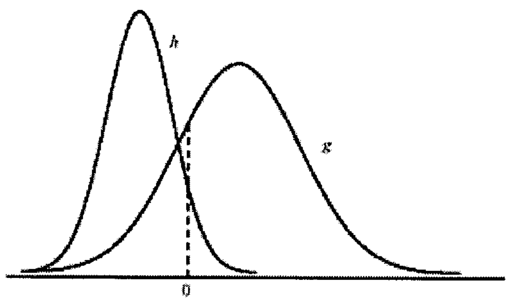


Figure 6-10. Taste variation [Train, 2003]

6.5.1. Conditional Distribution and Conditional Predictions

A student's choices reveal something about his/her tastes which we can discover. There is a precise way for performing this type of inference, given by [Revelt and Train, 2000] and [Train, 2003]. The main concept lies in using Bayesian procedures. Prior to collecting data from an individual student, the recommender's system estimates are based on past analyses given by a density of θ, called the prior distribution. The recommender system collects new data from a student in order to improve its ideas about the values of θ. In its most basic form, the utility that student n obtains in situation t is given by:

[6-3] $U_{nt} = X_{nt}\beta_n + \varepsilon_{nt}$, where $\beta_n \sim g(\beta\,|\,\theta)$,

and θ are the parameter of the distribution g. Let y_n=<y_{n1}, ... , y_{nT}> denote the student's t sequence of chosen ratings and X_n the collection of X_{nt} for those observations. Similar to section 5.1.2, we define the ordinal choice indicator as:

$$y_{nt} = i, if\ \tau_{i-1} < U_{nt} \le \tau_i, \qquad i = 1,...,7.$$

and

$$y_{\text{int}} = \begin{cases} 1, & if\ y_{nt} = i \\ 0, & otherwise \end{cases}.$$

If we knew β_n, then the probability of the person's sequence of ratings would be:

[6-4] $$P(y_n \mid X_n, \beta) = \prod_{t=1}^{T} \prod_{i=1}^{7} \left(\frac{1}{1+e^{\mu(X_{nt}\beta - \tau_i)}} - \frac{1}{1+e^{\mu(X_{nt}\beta - \tau_{i-1})}} \right)^{y_{int}}$$

Since we do not know β_n, we need to take the integral over the distribution of β:

[6-5] $$P(y_n \mid X_n, \theta) = \int P(y_n \mid X_n, \beta)\, g(\beta \mid \theta) d\beta;$$

The distribution of coefficient for the student giving a sequence of ratings y_n when facing a rating situation described by X_n is derived from Bayes' rule and is given by [Train, 2003]:

[6-6] $$h(\beta \mid y_n, X_n, \theta) = \frac{P(y_n \mid X_n, \beta)\, g(\beta \mid \theta)}{P(y_n \mid X_n, \theta)}$$

We know all the quantities on the right hand side and $g(\beta \mid \theta)$ is given from the full model (Table 5-2). The mean β for the new student model would be:

[6-7] $$\bar{\beta}_n = \int \beta \,.\, h(\beta \mid y_n, X_n, \theta) d\beta$$

This mean generally differs from the mean β in the entire population. Substituting the formula for *h*:

[6-8] $$\bar{\beta}_n = \frac{\int \beta . P(y_n \mid X_n, \beta)\, g(\beta \mid \theta)\, d\beta}{\int P(y_n \mid X_n, \beta)\, g(\beta \mid \theta)\, d\beta}$$

The integrals in this equation do not have a closed form; however, they can be readily simulated. Take draws of β from the population density $g(\beta \mid \theta)$. Calculate the weighted average of these draws, with the weight for draw β^d being proportional to $P(y_n \mid X_n, \beta^d)$. The simulated subpopulation mean is

[6-9] $$\hat{\beta}_n = \sum_d w^d \beta^d$$

where the weights are:

[6-10] $w^d = \frac{P(y_n \mid X_n, \beta^d)}{\sum_d P(y_n \mid X_n, \beta^d)}$

Suppose now that person n faces a new rating situation described by $X_{n(T+1)}$. If we had no information on the person's past choices, then we would assign the following probability to his/her rating i:

$$P(i \mid X_{n(T+1)}, \theta) = \int L_{n(T+1)}(i \mid X_{n(T+1)}, \beta)\, g(\beta \mid \theta) d\beta,$$

where:

$$L_{n(T+1)}(i \mid X_{n(T+1)}, \beta) = \frac{1}{1+e^{\mu(X_{n(T+1)}\beta-\tau_i)}} - \frac{1}{1+e^{\mu(X_{n(T+1)}\beta-\tau_{i-1})}}$$

If we observe the past ratings T of student n, then the probability can be conditioned on theses choices. The probability becomes:

$$P(i \mid X_{n(T+1)}, \theta) = \int L_{n(T+1)}(i \mid X_{n(T+1)}, \beta)\, h(\beta \mid y_n, X_n, \theta) d\beta$$

So instead of using $g(\beta \mid \theta)$, we mix over the conditional distribution $h(\beta \mid y_n, X_n, \theta)$. To calculate this probability, we substitute the formula for h from Equation [6-6] and simulate by taking draws of β from the population distribution $g(\beta \mid \theta)$. Calculating the logit formula for each draw and taking a weighted average of the results we obtain:

[6-11] $\hat{P}(i \mid X_{n(T+1)}, \theta) = \sum_d w^d L_{n(T+1)}(i \mid X_{n(T+1)}, \beta^d)$

where the weights w are given by Equation [6-10].

6.5.2. Student Model Application

The framework established in the previous section for calculating predictions for a new situation *(T+1)* conditioned on previous T observations can be readily applied to our student model and can be easily integrated in our Java framework using the simulator that was describe in the *DCM* Java class.

Using Equation [6-1] to get the likelihood function of a student faced by a new situation *(T+1)* for a given draw *d*, and using Equation [6-11] to calculate the conditional probability, we obtain:

$$\text{[6-12]}\quad \begin{aligned}&\hat{P}(i \mid X_{n(T+1)}, W_{n(T+1)})\\ &= \sum_d w^d P^d_{i(T+1)}\left(i \mid X_{n(T+1)}, W_{n(T+1)}, \zeta^d_{n(T+1)}, \tilde{v}^d_n; \beta, \tau, \delta, \gamma, \lambda, \Sigma_\omega, \sigma_v\right)\end{aligned}$$

where

$$w^d = \frac{P^d\left(y_n \mid X_n, W_n, \zeta^d, \tilde{v}^d; \beta, \tau, \delta, \gamma, \lambda, \Sigma_\omega, \sigma_v\right)}{\sum_d P^d\left(y_n \mid X_n, W_n, \zeta^d, \tilde{v}^d; \beta, \tau, \delta, \gamma, \lambda, \Sigma_\omega, \sigma_v\right)},$$

and

$$\begin{aligned}&P^d\left(y_n \mid X_n, W_n, \zeta^d, \tilde{v}^d; \beta, \tau, \delta, \gamma, \lambda, \Sigma_\omega, \sigma_v\right)\\ &= \prod_{t=1}^{T}\prod_{i=1}^{7}\left(\frac{1}{1+e^{\left(X_{nt}\beta + W_{nt}\delta + \sum_{l=1}^{4}\left(X_{\ln t}\lambda_l + F_l T \zeta^d_{nt}\right)\gamma_l + \sigma_v \tilde{v}^d_n - \tau_i\right)}} - \frac{1}{1+e^{\left(X_{nt}\beta + W_{nt}\delta + \sum_{l=1}^{4}\left(X_{\ln t}\lambda_l + F_l T \zeta^d_{nt}\right)\gamma_l + \sigma_v \tilde{v}^d_n - \tau_{i-1}\right)}}\right)^{y_{\text{int}}}\end{aligned}$$

Hence by using Equation [6-12], the predictions given by the Online Academic Advisor become personalized as they take into consideration previous ratings given by students.

6.6. Performance of the Online Academic Agent[20]

6.6.1. Data and Experimental Technique

In order to measure the performance of our recommender system, we divided the collected data that was used to estimate the model into two datasets; one was used to estimate the model and the other one was withheld to measure performance. The data consisted of 3286 ratings from 314 users having at least 5 ratings. 20% of the users (63 users) were randomly selected to be the test users. From the test sample, ratings for 5 items per user were withheld making up a total of 315 observations withheld from the data (approximately 10%). The quality of a given prediction can be measured by comparing the predicted values for the withheld ratings to the actual ratings.

[20] The notation in Section 6.6 is independent from the general notation of the thesis listed on page 9.

6.6.2. Metrics

The problem of developing good metrics to measure the effectiveness of recommendations has been extensively addressed in the recommender systems literature. Some examples of this work include [Mooney 1999], [Herlocker et al., 1999] and [Yand & Padmanabhan, 2001]. However, in most cases, the evaluation of a particular recommendation algorithm is usually limited only to testing its performance in terms of the coverage and accuracy metrics. Coverage is a measure of the percentage of items for which recommendation system can provide predictions [Herlocker et al., 1999]. This metric is more of a concern for collaborative filtering algorithms where recommending items is highly dependent on how often a particular item has been rated, or how well it can be associated with other items in the database. Coverage is not a concern in our algorithm as each class can be divided into attributes, and hence all classes can be potentially recommended regardless of whether they have been rated or not. Many metrics have been proposed for assessing the accuracy of a collaborative filtering system. In this research, we will be focus on statistical accuracy metrics that evaluate the accuracy of a filtering system by comparing the numerical prediction values against student ratings for the items that have both predictions and ratings [Herlocker et al., 1999].

Correlation between actual ratings and predictions has been commonly used to measure prediction engine performance [Hill, et al., 1995], [Kautz, et al., 1997], [Sarwar, et al., 1998]. The most common measure used is the Pearson correlation coefficient, this is the one reported in this dissertation. Pearson correlation measures the degree to which a linear relationship exists between two variables (see [Shardanand and Maes, 1995] and [Sawar et al., 1998] for more details). When computed in a sample, it is designated by the letter *r* and is sometimes called "Pearson's *r*". The value of *r* can vary from minus one to plus one. A minus one indicates a perfect negative correlation, while a plus one indicates a perfect positive correlation. A correlation of zero means there is no relationship between the two variables.

6.6.3. Performance with no Conditioning

In order to calculate the Pearson's correlation for the student model, the expected rating is computed for all the withheld ratings (5 ratings/user for 63 users). The expected rating is compared to the true ratings to compute *r* which is reported in Table 6-1 as "Model 1". Finally, we provide a naïve model ("Model 0") that assigns a correlation between the overall rating of a course and the true rating on how recommendable a course is. By calculating this correlation, the model actually measures how predictable recommendations are by simply taking into consideration the overall rating attribute. This naïve model allows us to benchmark the significantly more complex student model by computing the correlation improvement.

Table 6-1. Benchmarking The Student Model's Performance

Model	Description	# of observations/user used from withheld data	Pearson Correlation *r*	t-test
0	Naïve Model	5 (total # of obs = 315)	.31	6.18
1	Estimated Student Model without withheld data	5 (total # of obs = 315)	.67	15.54

The t-tests[21] indicate that the correlations are significantly different from zero at the 5% level (t-test > 1.96). The student model's performance shows a considerable improvement over the naïve model (a 116% increase in correlation). Table 6-2 is based on the methodology and notations described in Appendix G. A correlation coefficient *r* can be transformed into a z-score for purposes of hypothesis testing. This is done by using the following formula:

$$z = \frac{1}{2}\ln\left(\frac{1+r}{1-r}\right),$$

The end result is Fisher's *z*-score transformation of Pearson's *r*. Fisher's transformation reduces skew and makes the sampling distribution more normal as sample size *s* increases. Using Fisher's *z*-score, Table 6-2 shows that the difference in correlation is significant despite the relatively low number of observations.

[21] Refer to Appendix G for a description of statistical tests for Pearson correlations.

Table 6-2. Significance of Performance Improvement

Model	s	*r*	*z*	*Standard Error* of the difference	*t-test* of the difference
0	315	.33	.34	.67	4.24
1	315	.67	.81		

6.6.4. Performance with Conditioning

Recall that five randomly chosen observations were withheld from 63 randomly chosen users. In order to measure the performance of our updating system, we will divide those five observations per user into two parts:

- *u*: number of observations used to update the student model.
- *p*: number of observations used to measure performance ($p = 5 - u$).

We are also interested in observing the change in performance as *u* increases. For a given *u*, the procedure is as follows:

1. Randomly select *u* observations out of the 5 withheld ratings per user.
2. Using those *u* observations, calculate the expected rating for the *p* observations using the methodology describe in section 6.5.2.
3. Compute the Pearson's correlation *r* by comparing the expected ratings with the true ratings for those *p* observations.
4. The process is repeated *S* times and the average and standard deviation of the *S* calculated correlations are reported.

The results for u = 1,2,3 and 4 are presented in Table 6-3. The correlation improvement over the number of updates *u* is plotted on the chart in Figure 6-11.

Table 6-3. Comparative Performance using Pearson Correlation

Model	*u*	*S*	Pearson Correlation $\bar{r}$	Std Deviation	Paired t-test	% Improvement
1	0	0	.672			
2	1	50	.737	.0131	35.3	9.7
3	2	50	.754	.0126	6.54	2.3
4	3	50	.762	.0128	3.15	1.1
5	4	50	.767	.0114	1.71	.66

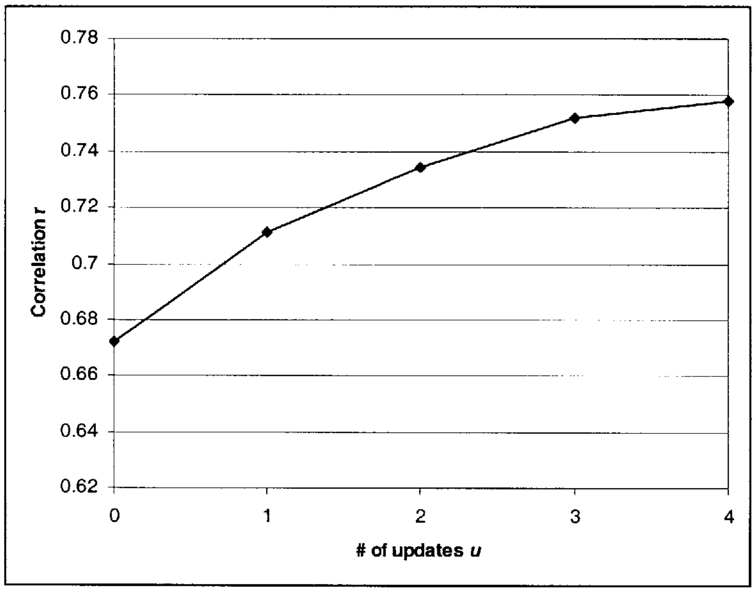


Figure 6-11. Performance vs. updates

The paired t-test was calculated by using the following equation:

$$t = \frac{\bar{r}_m - \bar{r}_{m-1}}{\sqrt{\frac{\sigma_m^2}{S_m} + \frac{\sigma_{m-1}^2}{S_{m-1}}}},$$

where:	m	is the model index,
	S_m	is the sample size (number of runs the Pearson Correlation was computed)
	$\bar{r}_m$	is the average Pearson Correlation over S_m
	σ_m	is the standard deviation of model m over S_m,

6.6.5. Performance Discussion

The average student model significantly showed an improvement over the naïve model. In Table 6-3, we see that conditioning on only one observation significantly improved the model by 9.7%. However, there are strongly decreasing returns to conditioning on more observations. Conditioning on two observations only improved the correlation by 2.3% over the correlation obtained from conditioning on one observation, and a mere 1.1% improvement for conditioning on 3 observations over 2. Finally conditioning on 4 observations didn't significantly improve performance over conditioning on 3 observations. Figure 6-11 shows how the performance graph flattens out as the number of updating observations increases. There are two reasons that might explain why the correlation gain from updating was not drastic and was mostly limited to the first observation.

First, the initial unconditioned average student model captured most of the variation in taste within the sampled student population. That leads to a stable model where the performance due to updating rapidly converges.

Second, we did not include in our average model all the attributes of the alternatives that were presented to students. In particular, we omitted attributes or variances (in case of random coefficients) that did not enter significantly in the estimation of the population parameters. Some students might respond to these omitted attributes and variances, even though they are insignificant for the population as a whole. Insofar as the updating observations involves trade-offs, the conditional distributions of tastes would be misleading, since the relevant tastes are excluded or the relevant tastes have been estimated as being fixed and hence cannot be updated. This explanation suggests that if a mixed logit is going to be used for obtaining conditional densities for each student, we might need to include attributes that could be important for some individuals even though they are insignificant for the population as a whole.

The updating module that was implemented as part of the Java package that was described in this chapter demonstrates how the distribution of coefficients for an individual are obtained from the distribution of coefficients in the population. While these conditional distributions can be useful in several ways, it is important to recognize the limitation of the concept as was demonstrated when the performance of the updating module was measured. While the conditional distribution of each student can be used, we would ideally relate preferences to observable demographics of the students. Those observable demographics (such as degree, concentration, requirements, gender, etc.) could be entered directly into the model itself, like we did in Chapter 5, so that the population parameters vary with the observed characteristics of the students in the population. In fact, entering demographics into the model is more direct and more accessible to hypothesis testing than estimating a model without these characteristics. By entering demographics into our student model however, the variability of the distributions becomes insignificant or limited to a few parameters. This leads to an initial model whose performance is high, but with limited room for improvement when it comes to updating with new observations. Although our specific application where a number of students' characteristics captured most of the variability in rating, this might not be true in other applications where there will always be great benefits from calculating user's conditional distributions even after including demographic data, or in applications where there is not enough information to identify people on the basis of demographics. The conditional densities would greatly facilitate analyses that have these characteristics.

Chapter 7: Conclusion

7.1. Summary and Contributions

Recommender systems represent user preferences for the purpose of suggesting items to purchase or examine. They have become fundamental applications in electronic commerce and information access, providing suggestions that effectively prune large information spaces so that users are directed toward those items that best meet their needs and preferences. A variety of techniques have been proposed for performing recommendation, including content-based, collaborative, knowledge-based and other techniques. One common thread in recommender systems research is the need to combine recommendation techniques to achieve peak performance. All of the known recommendation techniques have strengths and weaknesses, and many researchers have chosen to combine techniques in different ways.

In this dissertation, we investigated the use of discrete choice modeling as a new technique to constructing recommender systems. The approach was not specifically intended to overcome the weaknesses of any of the used techniques in the recommender systems literature, but was rather aimed at investigating and adapting a radically new model for giving personalized recommendations.

Discrete choice models had proven successful in many different areas and are still subject to continuous research to extend and enrich their capabilities. This research made use of the most advanced techniques in discrete choice modeling to develop statistical models that are behaviorally realistic. It also used advanced simulation methodologies that are becoming popular in the discrete choice literature to make the models scalable and estimable in a realistic timeframe. By using Bayesian updating techniques, this research incorporated a learning and personalization effect to the estimated models. Unlike the estimation of the initial models which is time consuming and is done offline, the Bayesian updating is fast and can be done online within a timeframe that online users would expect.

To test the framework of using discrete choice modeling with recommender systems, we tackled the problem of designing and developing an online academic advisor that recommends students academic courses they would enjoy. This application is innovative as no other system that we know of was developed to tackle this problem. The application also makes use of valuable course evaluation data that is available online for students, but is not efficiently used. The problem of advising courses is complex and should include assisting students in choosing which courses to take together and when to take them, what electives to choose, how to satisfy department requirements, and designing an extended list of courses that lead to graduation. The recommender system that this research developed did not aim at fully automating the problem of advising, but rather focused on developing an online software agent that assists students in assessing how recommendable a class would be for them to take and hence help them decide which term to take a required subject and which elective to take. For a given student, the online academic advisor produced a predicted likeness score for a list of recommended classes. The recommender system relied on a "student model" that was constructed and estimated based on a series of surveys and data collected from students. The model statistically inferred the importance of factors that influence student's overall impression of a class. Performance of the model was measured based on comparing withheld rating data with predicted data. The resulting high correlation between actual and predicted ratings showed that the recommender system was accurate in its predictions. By using the Bayesian updating techniques to personalize the student models, the correlation between actual and predicted ratings increased for the first two updating observations, indicating an increase in prediction accuracy and overall performance. However, the performance increase, although statistically significant, was modest. Updating with more than two observations did not improve the overall performance. Two reasons were identified that might explain why the correlation gain from updating was modest and was mostly limited to the first observation. First, the initial unconditioned average student model captured most of the variation in taste within the sampled student population. However capturing taste variations by including specific attributes might be more problematic in other applications than our academic advising. There will always be great benefits from calculating user's conditional distributions in these cases. Second, we omitted attributes or variances (in case of random coefficients) that did not enter significantly in the estimation of the population parameters. Some students might respond to these omitted attributes and variances, even though they are insignificant for the population as a whole. This explanation suggests that we might need to include attributes that could be important for some individuals even though they are insignificant for the population as a whole.

7.2. Research Direction

The methodology presented here and the empirical case study of recommending courses to students have brought to the surface the potential for using discrete choice models with recommender systems. Modeling individual choice behavior as part of an online system is a relatively new untested method. The design and implementation of an online academic advisor is innovative and no comparable application is found in the literature. Both areas require further investigation into numerous issues.

Bayesian Estimation: In this dissertation, the classical method of using maximum simulated likelihood has been used to estimate the parameter of the student model. A powerful set of procedures for estimating discrete choice models has been developed within the Bayesian tradition (details on Bayesian procedures can be found in [Train, 2001]). These procedures provide an alternative to the classical estimation method. The procedures can also be used to obtain information on individual-level parameters within a model with random taste variation. The Bayesian procedures avoid one of the most prominent difficulties associated with classical procedure. The Bayesian procedures do no require maximization of any function, avoiding the difficulty of solving simulated likelihood function that can be numerically difficult. Maximum simulated likelihoods sometimes fail to converge for various reasons. One reason is that the choice of starting values is often critical, with the algorithm converging from starting values that are close to the maximum but not from other starting values. The issue of local versus global maxima complicates the maximization further, since numerical convergence does not guarantee that the global maximum has been attained. For some behavioral models, Bayesian procedures are faster and more straightforward from a programming perspective than classical procedures. Developing a module that uses Bayesian procedures to estimate models would be an important extension to our current framework. It would avoid the use of computationally slow software tools, such as STATA, to maximize the likelihood functions. Having this module would hence complete our framework as a full Java package solution for constructing recommender systems.

Scalability: Scalability in recommender systems includes both very large problem sizes and real-time latency requirements. For instance, our recommender system connected to MIT's web site will get most of its recommendation requests around the same period of the academic year, probably peaking before the registration day of a given semester. Each recommendation must be produced within a few tens of milliseconds while serving hundreds or thousands of students simultaneously. The key performance measures are the maximum accepted latency for a recommendation, the number of simultaneous recommendation requests, the number of users, the number of courses available, and the number of ratings per student.

Effectiveness of recommendations: The evaluation of our recommender system was based on testing its performance in terms of accuracy metrics. Accuracy was measured statistically by comparing estimated ratings against actual ratings. Although crucial for measuring the correctness of recommendations, this technical measure does not capture adequately "usefulness" and "quality" of recommendations. For example, in our Online Academic Advisor application, predicting the rating of obvious courses, such as required courses, might not be helpful to the student. Therefore it is important to develop a usefulness measure that captures the "academic value" of the recommendations given by the system. Developing and studying such measures constitutes an interesting research topic.

User Interface: This research heavily focused on the statistical modeling and software architectural part of the recommender systems. Although effort has been made to make it architecturally easy to implement web-interfaces, work needs to be done on how best to present and collect information from students. Examples of important user interface extensions include:

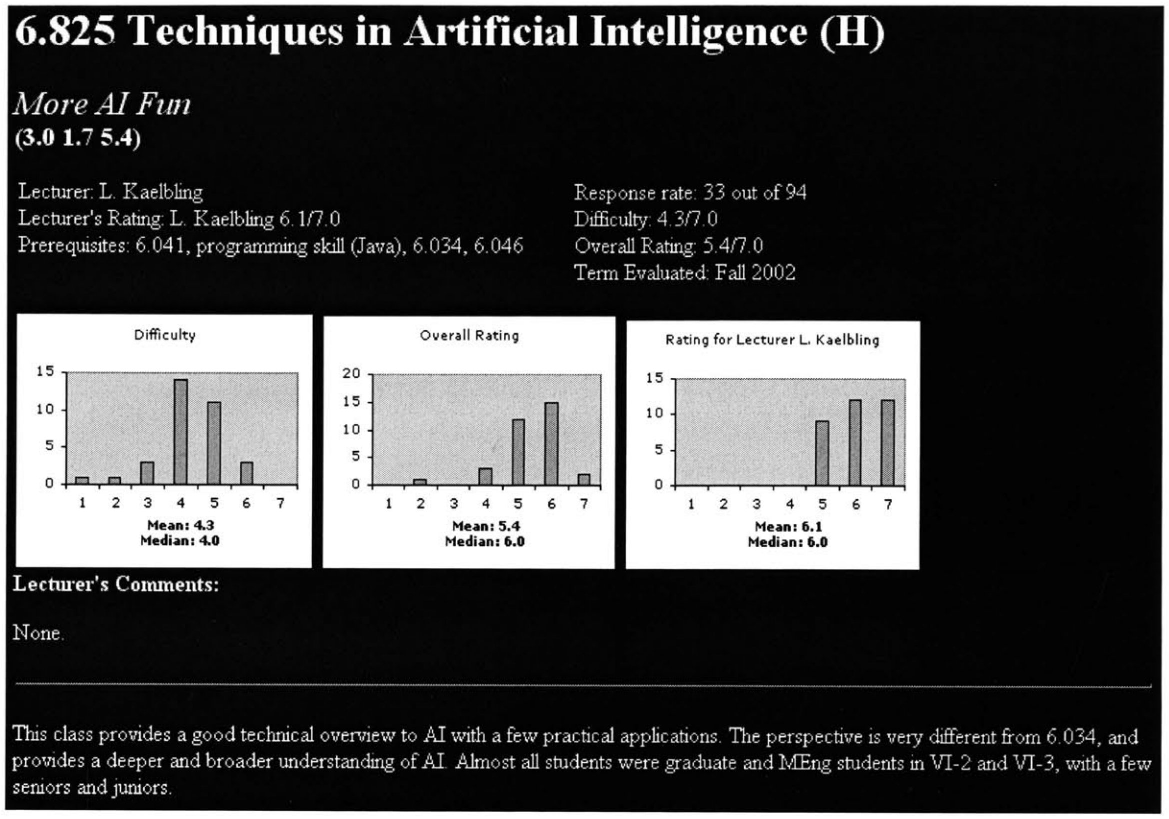
- Collecting data by designing intuitive and easy to access webpages to inform the system about one's preferences and personal information.
- Managing student accounts to give the ability to add ratings of courses taken, change demographic and login information, change some settings such as the maximum number of recommended courses to give, etc.
- Integrating a planner that would show the schedule of available courses along with their ratings on a calendar.
- Providing help and detailed explanations on how recommendation is performed and evaluated. This transparency would help increase the trustworthiness of the system.

Student Model: The Online Academic Advisor relied on many simplifying assumption to model students' ratings of subjects. The main assumption was to exclusively focus on recommending courses in the upcoming semester, ignoring an academic plan for the rest of the terms. Moreover, the modeling of the interaction between courses taken together or in a certain sequence over terms was limited. Finally, recommendations were only limited to courses given in the EECS department while ignoring humanities requirements and other potential courses given by the rest of MIT's departments. More research needs to be done in order to relax these assumptions by assisting students in choosing which courses to take together, what electives to choose, and designing an extended list of courses over many terms that lead to graduation.

7.3. Conclusion

All existing recommender systems employ one or more of a handful of basic techniques that have complementary advantages and disadvantages. Modeling what users might like or dislike is clearly complex, and those methods are a simplistic representation of this behavior. Advancements in discrete choice models offer the tools to improve the behavioral representation of users' preferences by integrating methods that exploit the use of different types of data, capture unobserved heterogeneity for all aspects of the rating process, and explicitly model behavioral constructs such as attitudes and perceptions. With increasing computational power and increasingly rich datasets, the framework described in this dissertation can be practically applied to any application and improve forecasts and predictions of individual ratings and choices. The online academic advisor that this research developed served as a proof of the flexibility and practicality of this approach.

Appendix A



6.825 Techniques in Artificial Intelligence (H)

More AI Fun
(3.0 1.7 5.4)

Lecturer: L. Kaelbling
Lecturer's Rating: L. Kaelbling 6.1/7.0
Prerequisites: 6.041, programming skill (Java), 6.034, 6.046

Response rate: 33 out of 94
Difficulty: 4.3/7.0
Overall Rating: 5.4/7.0
Term Evaluated: Fall 2002

Lecturer's Comments:

None.

This class provides a good technical overview to AI with a few practical applications. The perspective is very different from 6.034, and provides a deeper and broader understanding of AI. Almost all students were graduate and MEng students in VI-2 and VI-3, with a few seniors and juniors.

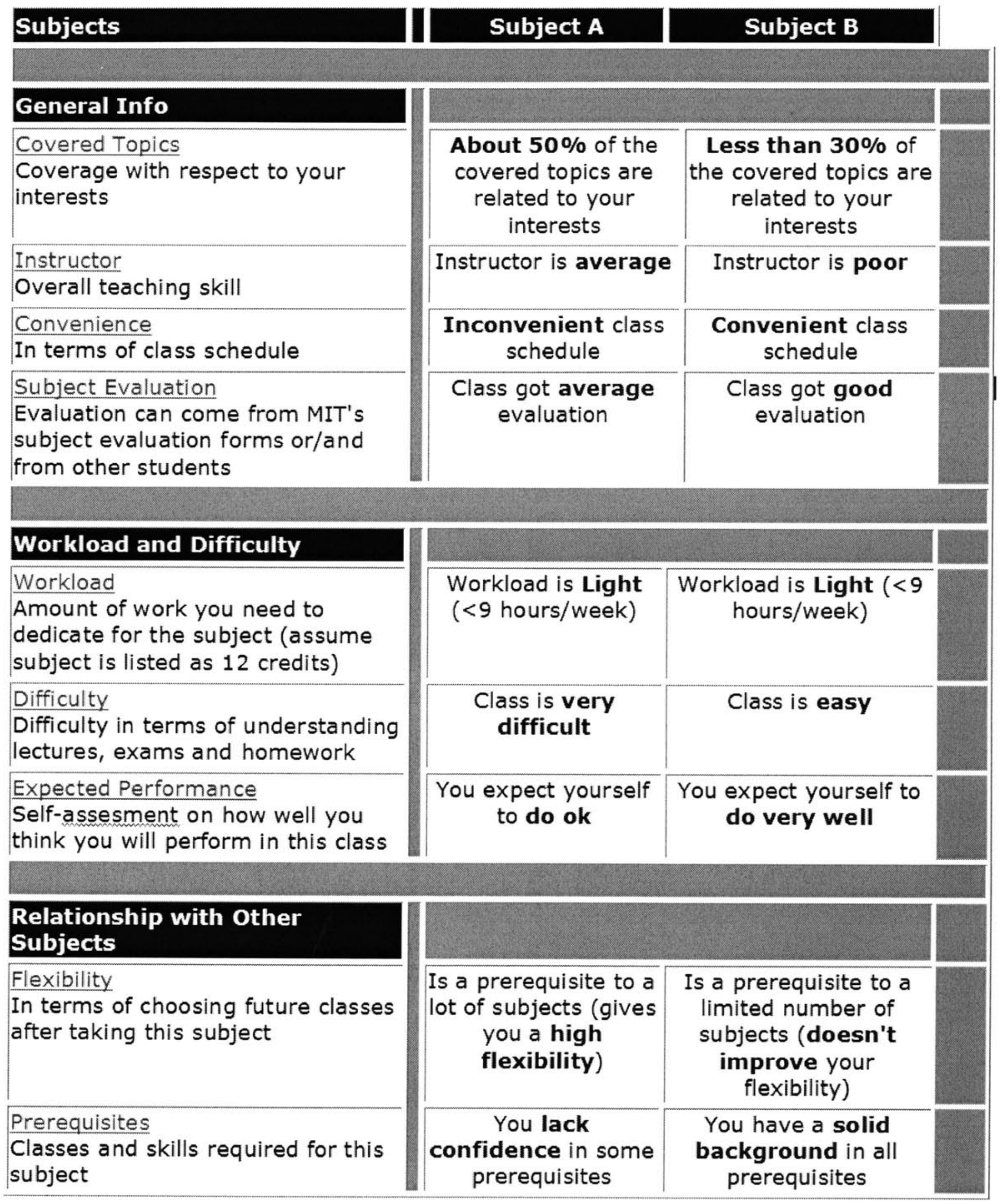
For a given class, the names, prerequisites, and hours (class, lab, preparation) given are *not* the official ones. Instead, what the students surveyed thought was appropriate is presented. The lecturer listed is the lecturer for the term the subject was evaluated and is not necessarily the lecturer in the following term. Response rate is the number of evaluation forms received, and total enrollment for the class is based on post-add date statistics. "Difficulty" is valued on a scale of 1 (Trivial) to 7 (Impossible), with 4.0 as the optimal response. "Subject Overall" and "Lecturer Overall" use the same scale. For these two quantities, a low number indicates a negative response, while a higher one demonstrates a favorable one.

Appendix B

The conjoint analysis part presented students with 15 hypothetical courses with different levels of the five considered attributes. The respondents had to rank those 15 courses according to the course they would most likely select. The five attributes were the following:

- Teaching Style: The style of delivering the content of the course. In particular I was interested in three lecture styles:
  - Lecture (more than 60% of the class consists of lecture)
  - Mix (case studies and team projects 40- 60%, lecture 40-60%)
  - Case (more than 60% case studies and team projects)
- Instructor Rating: This is an overall rating of the professor giving the course on a three level scale. Fair, Good, and Excellent. Examples of factors that influence this attribute are the instructor's teaching skills and experience in the field.
- Workload: The amount of work a student needs to dedicate for the course and the difficulty of the material.
  - Light (requiring less than 4 hours of study per week)
  - Medium (4 to 8 hours)
  - Heavy (more than 8 hours)
- Content: the level of relevancy of the course content with the student's interests on a three level scale; high, medium and low.
- Convenience: The convenience of the course from a time schedule and location point of view. In this questionnaire, a course is rated as either Convenient or Inconvenient.

Appendix C



Subjects	Subject A	Subject B
General Info		
Covered Topics Coverage with respect to your interests	**About 50%** of the covered topics are related to your interests	**Less than 30%** of the covered topics are related to your interests
Instructor Overall teaching skill	Instructor is **average**	Instructor is **poor**
Convenience In terms of class schedule	**Inconvenient** class schedule	**Convenient** class schedule
Subject Evaluation Evaluation can come from MIT's subject evaluation forms or/and from other students	Class got **average** evaluation	Class got **good** evaluation
Workload and Difficulty		
Workload Amount of work you need to dedicate for the subject (assume subject is listed as 12 credits)	Workload is **Light** (<9 hours/week)	Workload is **Light** (<9 hours/week)
Difficulty Difficulty in terms of understanding lectures, exams and homework	Class is **very difficult**	Class is **easy**
Expected Performance Self-assesment on how well you think you will perform in this class	You expect yourself to **do ok**	You expect yourself to **do very well**
Relationship with Other Subjects		
Flexibility In terms of choosing future classes after taking this subject	Is a prerequisite to a lot of subjects (gives you a **high flexibility**)	Is a prerequisite to a limited number of subjects (**doesn't improve** your flexibility)
Prerequisites Classes and skills required for this subject	You **lack confidence** in some prerequisites	You have a **solid background** in all prerequisites

• Definitely A	• Definitely B
• Probably A	• Probably B
• No Preference	

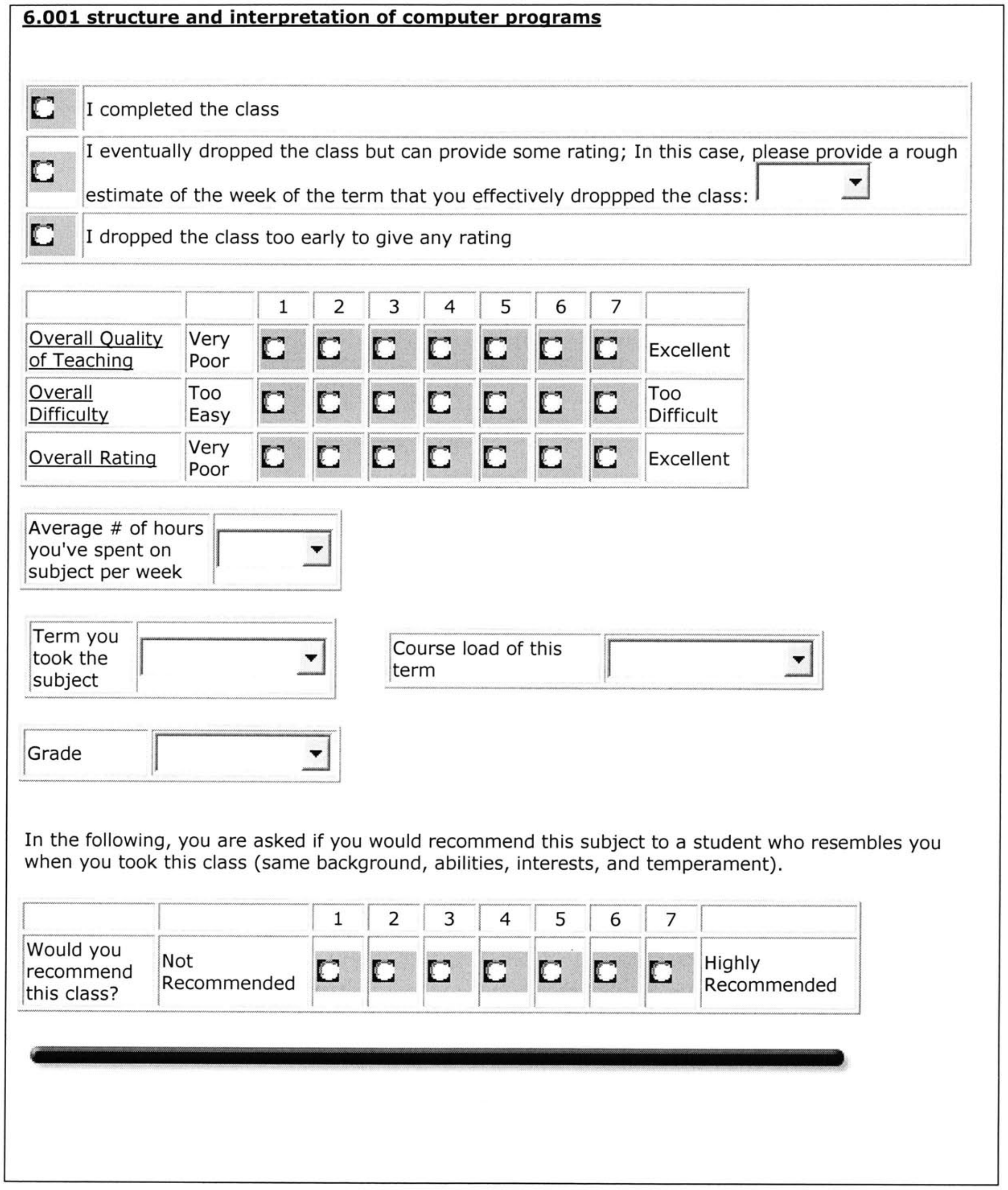
Table C-1. Description of the Attributes

Attribute	**Description**
Covered Topics	A class usually covers several topics. Assume that: - **More than 70%** of the covered topics are related to your interests - **About 50%** of the covered topics are related to your interests - **Less than 30%** of the covered topics are related to your interests
Instructor	The overall instructor's teaching skill is rated on a three level scale: **Excellent, Average** and **Poor.**
Convenience	In terms or class schedule; an example of an inconvenient course might be: classes on Friday or meeting at 8:30 am. In this survey, a subject is rated as either **Convenient** or **Inconvenient.**
Subject Evaluation	Is what you heard about the class from your fellow students or from the course evaluation forms; in this survey a course can either have: - a **Good evaluation** - an **Average evaluation** - a **Bad evaluation**
Workload	The amount of work you need to dedicate for the subject (assuming the subject is listed as 12 credit subject). Three levels are presented: - **Light** (requiring less than 9 hours of study per week for a 12 unit course) - **Medium** (9 to 15 hours/week) - **Heavy** (more than 15 hours/week)
Difficulty	Is a measure on how difficult the material is in terms of exams and homework; the exams or homework in a class can either be: - **Very Difficult** - of **Medium Difficulty** - **Easy**
Expected Performance	Is a measure on how well you think you will perform in a class before actually taking it; in this survey, you either expect yourself to: **- Do very well** **- Do ok** **- Do poorly**
Flexibility	In terms of future classes you might take. A class can be a prerequisite to: - A lot of subjects giving you a **high flexibility** in choosing your future subjects - A limited number of classes and hence **doesn't improve your flexibility** in future subjects
Prerequisites	Given a subject with prerequisites, you can consider yourself as either: - Having a **solid background** in all the prerequisites **- Lacking confidence** in some perquisites

Appendix D

Table D-1. Description the attributes in the revealed preference section

Overall Quality of Teaching	In rating this attribute, please consider the followings: - well prepared lectures - instructor(s) explained clearly and stimulated interest in the subject - instructor(s) encouraged questions and class participation - help was available outside of class for questions If more than one instructor, please use this rating as an average of the quality of teaching of all the instructors
Overall Difficulty	This rating should be a conglomeration of the level of difficulty of the text and readings, exams and problem sets. The amount of time spent on this subject should not be considered a factor in this rating
Overall Rating	This attribute can be seen as a conglomeration of: - the quality of teaching, - the difficulty and usefulness of the text and readings - the difficulty and the quantity of exams and problem sets - the pace, workload and organization of the subject - overall value of what is taught or learned


6.001 structure and interpretation of computer programs

○	I completed the class
○	I eventually dropped the class but can provide some rating; In this case, please provide a rough estimate of the week of the term that you effectively droppped the class:
○	I dropped the class too early to give any rating

		1	2	3	4	5	6	7	
Overall Quality of Teaching	Very Poor	○	○	○	○	○	○	○	Excellent
Overall Difficulty	Too Easy	○	○	○	○	○	○	○	Too Difficult
Overall Rating	Very Poor	○	○	○	○	○	○	○	Excellent

Average # of hours you've spent on subject per week	

Term you took the subject		Course load of this term	

Grade	

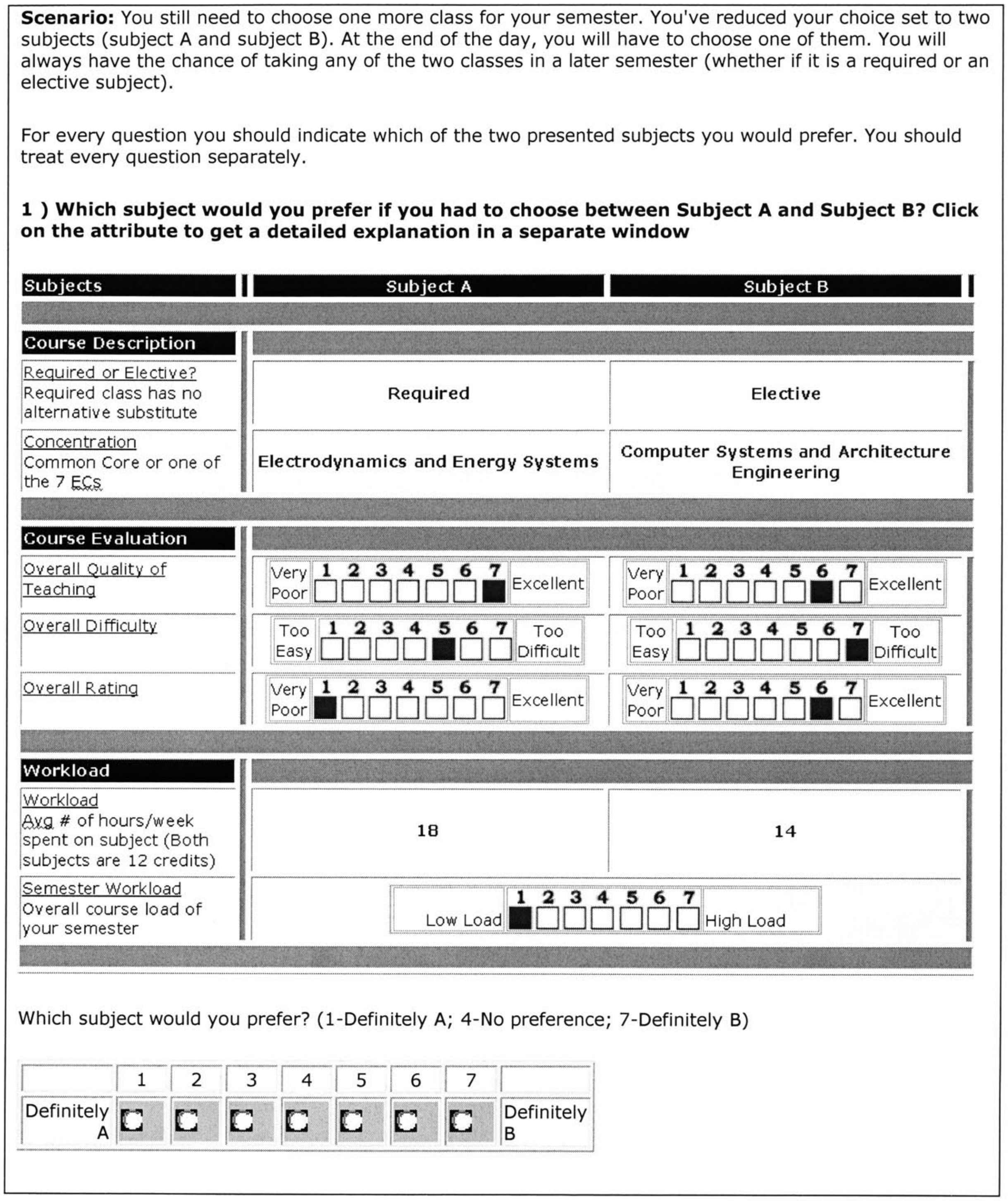
In the following, you are asked if you would recommend this subject to a student who resembles you when you took this class (same background, abilities, interests, and temperament).

		1	2	3	4	5	6	7	
Would you recommend this class?	Not Recommended	○	○	○	○	○	○	○	Highly Recommended

Figure D-7-1. Sample revealed preference question

Table D-2. Description the attributes in the stated preference section

Required or Elective?
A required class is a class with no alternative substitute for it; e.g. if you're a 6.1, then the headers of the EE concentrations are requirements.
Engineering Concentration
Technical area that the subject covers. This can be either a **Common Core** subject or one of the **7 Engineering Concentrations**.
Overall Quality of Teaching
This rating takes into account the followings: - well prepared lectures - instructor(s) explained clearly and stimulated interest in the subject - instructor(s) encouraged questions and class participation - help was available outside of class for questions If more than one instructor, please use this rating as an average of the quality of teaching of all the instructors
Overall Difficulty
This rating is an conglomeration of the level of difficulty of the text and readings, exams and problem sets. The amount of time spent on this subject should not be considered a factor in this rating.
Overall Rating
This attribute can be seen as a conglomeration of: - the quality of teaching, - the difficulty and usefulness of the text and readings - the difficulty and the quantity of exams and problem sets - the pace, workload and organization of the subject - overall value of what is taught or learned
Workload
The amount of work you need to dedicate for the subject (assuming the subject is listed as 12 credit subject).
Semester Workload
Overall course load of the semester you're considering taking this subject with



Scenario: You still need to choose one more class for your semester. You've reduced your choice set to two subjects (subject A and subject B). At the end of the day, you will have to choose one of them. You will always have the chance of taking any of the two classes in a later semester (whether if it is a required or an elective subject).

For every question you should indicate which of the two presented subjects you would prefer. You should treat every question separately.

1) Which subject would you prefer if you had to choose between Subject A and Subject B? Click on the attribute to get a detailed explanation in a separate window

Subjects	Subject A	Subject B
Course Description		
Required or Elective? Required class has no alternative substitute	**Required**	**Elective**
Concentration Common Core or one of the 7 ECs	**Electrodynamics and Energy Systems**	**Computer Systems and Architecture Engineering**
Course Evaluation		
Overall Quality of Teaching	Very Poor 1 □ 2 □ 3 □ 4 □ 5 □ 6 □ 7 ■ Excellent	Very Poor 1 □ 2 □ 3 □ 4 □ 5 □ 6 ■ 7 □ Excellent
Overall Difficulty	Too Easy 1 □ 2 □ 3 □ 4 □ 5 ■ 6 □ 7 □ Too Difficult	Too Easy 1 □ 2 □ 3 □ 4 □ 5 □ 6 □ 7 ■ Too Difficult
Overall Rating	Very Poor 1 ■ 2 □ 3 □ 4 □ 5 □ 6 □ 7 □ Excellent	Very Poor 1 □ 2 □ 3 □ 4 □ 5 □ 6 ■ 7 □ Excellent
Workload		
Workload Avg # of hours/week spent on subject (Both subjects are 12 credits)	18	14
Semester Workload Overall course load of your semester	Low Load 1 ■ 2 □ 3 □ 4 □ 5 □ 6 □ 7 □ High Load	

Which subject would you prefer? (1-Definitely A; 4-No preference; 7-Definitely B)

	1	2	3	4	5	6	7	
Definitely A	○	○	○	○	○	○	○	Definitely B

Figure D-7-2. Sample stated preference question

Appendix E

The base model presented in Table E.1 was reached after estimating similar models that combined different parameters and eliminating all insignificant factors. In this model, no latent variables were incorporated in the model. Instead, the "Teaching", "Overall", "Workload" and "Difficulty" attributes are the values from the evaluation data.

Table E-1. Base Model
Choice Model

Course Utility	Est. β, δ, λ	t-stat
Teaching	0.19	5.23
Overall	0.66	7.76
Workload	0.192	2.48
Difficulty	-0.139	-1.94
Concentration	0.12	7.19
Required	0.43	6.36
Term Late	1.01	3.71
How Late	-0.21	-2.21
How Early	0.17	2.02
Required x Term Early	-0.60	-3.71
Required x Term Late	-1.06	-4.89
τ^{SP} (SP constant)	-0.42	-3.02
μ^{SP} (SP constant)	0.51	19.25
ν_n (Panel Data)	0.09	2.75
τ_1	1.66	7.94
τ_2	3.00	11.32
τ_3	3.94	16.79
τ_4	5.04	18.74
τ_5	6.30	21.12
τ_6	7.64	24.94
Rho-bar Squared		0.316

The model presented in Table E.2 incorporates the four latent variables described in section 5.2.1. The incorporation of those four latent variables in the student model significantly improved the goodness of fit of the choice model. Note that some of this improvement in fit would probably be captured in the choice model by including in the base choice model the additional variables that are included in the latent variable structural model. The rho-bar-squared for the model with latent variables uses the same degrees of freedom correction as the base model (the 4 attributes were replaced by the 4 latent variables), and thus this degrees of freedom adjustment only accounts for the estimated parameters of the choice model.

Table E-2. Model with Latent Variables
Choice Model

Course Utility	**Est. β, δ, λ**	**t-stat**
T^*: Teaching (Latent)	0.38	8.37
O^*: Overall	0.54	9.95
W^*: Workload	0.38	2.74
D^*: Difficulty	-0.11	-2.33
Concentration	0.16	6.90
Required	0.34	5.14
Term Late	0.93	3.78
How Late	-0.19	-1.97
How Early	0.15	1.98
Required x Term Early	-0.57	-3.62
Required x Term Late	-1.04	-4.74
τ^{SP} (SP constant)	-0.49	-3.55
μ^{SP} (SP constant)	0.62	25.31
ν_n (Panel Data)	0.12	2.20
τ_1	2.19	5.69
τ_2	3.48	9.00
τ_3	4.35	10.93
τ_4	5.22	12.93
τ_5	6.34	15.30
τ_6	7.70	18.04
Rho-bar Squared		0.421

Latent Variable Model

Structural Models		**Est. γ, σ**	**t-stat**
Teaching			
	Lecturer	0.94	118.45
	Variance ω_T	0.31	13.64
Overall			
	Overall	0.95	121.75
	Variance ω_O	0.33	10.97
Covariance ω_{OT}		0.32	15.27
Workload			
	Workload	0.45	31.87
	Female	0.17	4.90
	Degree EE	0.33	5.82
	Degree EECS	0.45	13.79
	Variance ω_W	0.15	27.80
Difficulty			
	Difficulty	0.86	56.95
	Degree EE	0.13	1.25
	Degree EECS	0.59	5.52
	Variance ω_D	0.36	11.19
Covariance ω_{OT}		0.16	6.96

Measurement Model		**Est. σ**	**t-stat**
$I_{Lecturer}$	Variance υ_T	1.37	59.85
$I_{Overall}$	Variance υ_O	1.29	59.92
$I_{Workload}$	Variance υ_W	0.26	55.27
$I_{Difficulty}$	Variance υ_D	1.12	56.41

The final model that is presented in Table 5-2 incorporates 4 additional factors to the model presented in Table 5-2. Adding those 4 variables not only improved on the goodness of fit, but also helped explaining the reason behind students rating SP questions about 0.5 points lower than RP questions (τ^{SP} = -0.49 in Table E.2). More Results interpretations are provided in section 5.4.

Appendix F

combined_model.do

```
quietly{
clear

set more off
global Path "c:\documents and settings\bhc\my documents\research\phd\New
Questionnaire\Analysis\STATA\"

global D_Draws = 5000
global S_seed = 123456789
global N_Var = 6

insheet using "${Path}Data\halton_${D_Draws}.txt", tab clear
sort id
save halton, replace

insheet using "${Path}Data\Combined_Data.txt", comma clear
//insheet using "${Path}Data\synthetic.txt", tab clear

sort id
merge id using halton
drop if _merge == 2

run "${Path}SML\Do Files\data_corrections.do"
run "${Path}SML\Do Files\rp_data.do"
run "${Path}SML\Do Files\demographics.do"
run "${Path}SML\Do Files\sp_data.do"
run "${Path}SML\Do Files\interactions.do"
run "${Path}SML\Do Files\combined.do"
run "${Path}SML\ologit_halton.do"
}

global SP_DATA sp_data
global Panel id


//inverse cumulative normal for all the halton draws and assign them to global variables
local var = 1
while `var'<=$N_Var{
        local r=1
        while `r'<=$D_Draws{
                qui replace halton_var`var'_iter`r' = invnorm(halton_var`var'_iter`r')
                global HALTON_var`var'_iter`r' halton_var`var'_iter`r'
                local r = `r' + 1
        }
        local var = `var' + 1
}


replace workload = workload/12
replace workload_c = workload_c/12

global Teaching teaching
global Overall overall
global Workload workload
global Difficulty difficulty

replace how_late = 0 if how_late == .

global Cholesky_C
constraint drop _all

#delimit ;
ml model d0 ologit_halton
(Combined:  rating = concentration_c required_c, nocons)
(RP_Only: term_late how_early how_late r_term_early r_term_late, nocons)
(SP_Only:)
(T_Gamma: teaching_c, nocons)(T_Sigma:)(T_SV:)(T_Beta:)
(O_Gamma: overall_c, nocons)(O_Sigma:)(O_SV:)(O_Beta:)
(W_Gamma: workload_c female degree_ee degree_eecs, nocons)(W_Sigma:)(W_SV:)(W_Beta:)
(D_Gamma: difficulty_c female degree_ee degree_eecs, nocons)(D_Sigma:)(D_SV:)(D_Beta:)
```

```
(T_O_Covariance:)
(W_D_Covariance:)
(SP_mu:)
(Rho:)
(cut_1:)(cut_2:)(cut_3:)(cut_4:)(cut_5:)(cut_6:)
;
#delimit cr


di c(current_date) ": " c(current_time)
di "Number of Halton Draws: " $D_Draws
ml max
di c(current_date) ": " c(current_time)
```

combined.do

```
//Combine data
gen rating = sp
replace rating = rp if rating == .


gen required_c = required                      if sp_data == 0
replace required_c = sp_required               if sp_data == 1
//replace required_c = 0                              if required_c == .

gen concentration_c = concentration            if sp_data == 0
replace concentration_c = sp_concentration     if sp_data == 1
//replace concentration_c = 0                         if required_c == .

gen commoncore_c = commoncore                         if sp_data == 0
replace commoncore_c = sp_commoncore           if sp_data == 1


gen teaching_c = lecturer                      if sp_data == 0
replace teaching_c = sp_teaching               if sp_data == 1
//replace teaching_c = 0                              if teaching_c == .

gen overall_c = overallrating                         if sp_data == 0
replace overall_c = sp_overall                        if sp_data == 1
//replace overall_c = 0                               if overall_c == .

gen useful_c = useful                          if sp_data == 0
replace useful_c = sp_useful                   if sp_data == 1
//replace useful_c = 0                         if useful_c == .

gen difficulty_c = course_evaluation_difficulty       if sp_data == 0
replace difficulty_c = sp_difficulty            if sp_data == 1
//replace difficulty_c = 0                      if difficulty_c == .

gen workload_c = workload_eval                        if sp_data == 0
replace workload_c = sp_workload               if sp_data == 1
//replace workload_c = 0                              if workload_c == .

gen termworkload_c = termworkload              if sp_data == 0
replace termworkload_c = sp_termworkload       if sp_data == 1
//replace termworkload_c = 0                   if workload_c == .
```

data_corrections.do

```
//data corrections
replace lecturer1rating = 5.1 if coursenumber == 6034 & term == "Fall 1998"

replace lecturer1rating = 5.5 if coursenumber == 6033 & term == "Fall 2001"
replace lecturer2rating = 3.8 if coursenumber == 6033 & term == "Fall 2001"
replace lecturer1rating = 5.3 if coursenumber == 6033 & term == "Fall 2003"
replace lecturer2rating = 3.9 if coursenumber == 6033 & term == "Fall 2003"
replace lecturer1rating = 5.5 if coursenumber == 6033 & term == "Fall 2002"
replace lecturer2rating = 3.8 if coursenumber == 6033 & term == "Fall 2002"

//Drop students with no rp data

drop if id == 24
drop if id == 36
drop if id == 59
drop if id == 73
drop if id == 94
drop if id == 105
drop if id == 117
drop if id == 185
drop if id == 301
drop if id == 313
```

rp_data.do

```
//Required:
replace required = -1 if required == .
replace commoncore = 0 if commoncore == .
replace mathematics = 0 if mathematics == .

//Difficulty
generate difficulty = rp_courses_difficulty

//Workload
gen workload_eval = hoursclass + hourslab + hoursprep

//Evaluation forms lecturer
gen lecturer = (lecturer1rating + lecturer2rating)/2 if lecturer2rating != 0
replace lecturer = lecturer1rating if lecturer2rating == 0

run "${Path}SML\Do Files\rp_cleaning.do"

//Concentration: replace concentration by ranking importance
replace ec1 = 0 if ec1 == .
replace ec2 = 0 if ec2 == .
replace ec3 = 0 if ec3 == .
replace ec4 = 0 if ec4 == .
replace ec5 = 0 if ec5 == .
replace ec6 = 0 if ec6 == .
replace ec7 = 0 if ec7 == .
replace ccsp = 0 if ccsp == .
replace aia = 0 if aia == .
replace dcs = 0 if dcs == .
replace csae = 0 if csae == .
replace ees = 0 if ees == .
replace tcs = 0 if tcs == .
replace be = 0 if be == .

//Department lab
gen departmentLab = 0
replace departmentLab = 1 if coursenumber<=6182 & coursenumber>=6100

//Grades
gen grade_scale = 8           if grade == "A+"
replace grade_scale = 7       if grade == "A"
replace grade_scale = 6       if grade == "A-"
replace grade_scale = 5       if grade == "B+"
replace grade_scale = 4       if grade == "B"
replace grade_scale = 3       if grade == "B-"
replace grade_scale = 2       if grade == "C+"
replace grade_scale = 1       if grade == "C or below"
replace grade_scale = 0       if grade_scale == .

gen grade_given = 1 if grade_scale != 0
replace grade_given = 0 if grade_scale == 0

//Terms
gen term_num = -10            if term== "Fall 1998"
replace term_num = -8         if term== "Fall 1999"
replace term_num = -6         if term== "Fall 2000"
replace term_num = -4         if term== "Fall 2001"
replace term_num = -2         if term== "Fall 2002"
replace term_num = 0          if term== "Fall 2003"
replace term_num = -9         if term== "Spring 1999"
replace term_num = -7         if term== "Spring 2000"
replace term_num = -5         if term== "Spring 2001"
replace term_num = -3         if term== "Spring 2002"
replace term_num = -1         if term== "Spring 2003"

gen term_start = 0            if year == 1
replace term_start = -2               if year == 2
replace term_start = -4               if year == 3
replace term_start = -6               if year == 4

gen term_taken = term_num - term_start

//Concentration
gen concentration  = 0
```

```
replace concentration  = ec1 if ccsp == 1
replace concentration  = concentration  + ec2 if aia == 1
replace concentration  = concentration  + ec3 if dcs == 1
replace concentration  = concentration  + ec4 if csae == 1
replace concentration  = concentration  + ec5 if ees == 1
replace concentration  = concentration  + ec6 if tcs == 1
replace concentration  = concentration  + ec7 if be == 1

//Concentration
gen ccsp_conc = 0
replace ccsp_conc = 1 if ec1 >= 3
gen aia_conc = 0
replace aia_conc  = 1 if ec2 >= 3
gen dcs_conc = 0
replace dcs_conc  = 1 if ec3 >= 3
gen csae_conc = 0
replace csae_conc = 1 if ec4 >= 3
gen ees_conc = 0
replace ees_conc  = 1 if ec5 >= 3
gen tcs_conc = 0
replace tcs_conc  = 1 if ec6 >= 3
gen be_conc = 0
replace be_conc   = 1 if ec7 >= 3


//Timing
sort coursenumber
by coursenumber: egen term_avg = mean(term_taken)
gen term_timing = term_taken - term_avg
gen term_timing_int = round(term_timing)

gen term_abs = abs(term_timing_int)

gen term_early = 0
        replace term_early = 1 if term_timing_int < 0
gen term_late = 0
        replace term_late = 1 if term_timing_int > 0

gen how_early = term_abs * term_early
gen how_late  = term_abs * term_late

replace how_early = 0 if how_early == .
replace how_late = 0 if how_late == .

gen required_temp = 0
replace required_temp = 1 if required == 1

gen r_term_early = term_early * required_temp
gen r_term_late = term_late * required_temp
gen r_how_early = how_early * required_temp
gen r_how_late = how_late * required_temp

drop required_temp


//Prerequisites
sort id
gen temp = .
gen avgx =.
gen quantity = 0
local num = 1
while `num'<=3{
        qui gen preU_`num' = .

        local pre_num = 6001
        qui replace prerequisite_`num' = . if prerequisite_`num'<6000 |
prerequisite_`num'>7000

        while `pre_num'<=6400{
                quietly{
                        replace temp =.
                        replace avgx=.
                        by id: replace temp = rp if coursenumber == `pre_num'
                        by id: replace avgx = sum(temp)/sum(temp !=.)
                        by id: replace temp = avgx[_N]
                        by id: replace preU_`num' = temp if prerequisite_`num' == `pre_num'
                        local pre_num = `pre_num' +1
                }
        }
        quietly{
                replace preU_`num' = 0 if preU_`num' == .
                by id: replace quantity = quantity + 1 if preU_`num' !=0
        }
        local num = `num' + 1
}


gen pre_avg = (preU_1 + preU_2 + preU_3)/quantity
replace pre_avg = 0 if pre_avg == .
drop temp preU_1 preU_2 preU_3 avgx quantity
```

sp_data.do

```
//SP data
gen con_1 = 0
replace con_1 = ec1 if concentration_1 == 1
replace con_1 = ec2 if concentration_1 == 2
replace con_1 = ec3 if concentration_1 == 3
replace con_1 = ec4 if concentration_1 == 4
replace con_1 = ec5 if concentration_1 == 5
replace con_1 = ec6 if concentration_1 == 6
replace con_1 = ec7 if concentration_1 == 7

gen con_2 = 0
replace con_2 = ec1 if concentration_2 == 1
replace con_2 = ec2 if concentration_2 == 2
replace con_2 = ec3 if concentration_2 == 3
replace con_2 = ec4 if concentration_2 == 4
replace con_2 = ec5 if concentration_2 == 5
replace con_2 = ec6 if concentration_2 == 6
replace con_2 = ec7 if concentration_2 == 7


gen sp_required = (required_1 - required_2)/2 // Reverse scale to have 1 is required and
-1 is elective
gen sp_concentration = (con_2 - con_1) +4
gen sp_teaching = (teaching_2 - teaching_1 + 6)/2+1
gen sp_difficulty = (difficulty_1 - difficulty_2 + 6)/2+1 //Reversed scale
gen sp_overall = (overall_2 - overall_1 + 6)/2+1
gen sp_workload = ((workload_2 - workload_1 +6)/2+1)*2+6
gen sp_termworkload = termworkload_1+1   // Only one termworkload presented
gen sp_heavycourse = (heavycourse_1 - heavycourse_2+6)/2 + 1 // Reverse the scale:
consistent with workload
gen sp_useful = (useful_2 - useful_1+6)/2+1


gen sp_commoncore = 0
replace sp_commoncore = 1 if concentration_1 == 0 & sp<4
replace sp_commoncore = 1 if concentration_2 == 0 & sp>4
```

interactions.do

```
gen workload_term = termworkload * workload // workload nomramlized by load of current
term

gen sp_workload_term = sp_termworkload * sp_workload // workload nomramlized by load of
current term
gen sp_heavycourse_term = sp_termworkload * sp_heavycourse
gen sp_work_inter = sp_workload_term * sp_heavycourse_term
gen sp_workload_avg = sp_workload*avg_units/12 //workload normalized by average units per
semester

gen overall_teaching = overall * teaching
gen overall_useful = overall * useful
gen teaching_useful = teaching * useful
gen difficulty_workload = difficulty * workload


gen ccsp_match = 0
replace ccsp_match = 1 if ccsp== 1 & ccsp_conc == 1

gen aia_match = 0
replace aia_match = 1 if aia == 1 & aia_conc == 1

gen dcs_match = 0
replace dcs_match = 1 if dcs == 1 & dcs_conc == 1

gen csae_match = 0
```

```
replace csae_match = 1 if csae == 1 & csae_conc == 1

gen ees_match = 0
replace ees_match = 1 if ees == 1 & ees_conc == 1

gen tcs_match = 0
replace tcs_match = 1 if tcs == 1 & tcs_conc == 1

gen be_match = 0
replace be_match = 1 if be == 1 & be_conc == 1
```

demographics.do

```
replace doublemajor = 1 if doublemajor == .
replace doublemajor = doublemajor -1   //make it a "0" or "1" variable

//UROP
replace urop = 2 if urop == .
replace urop = 0 if urop == 2 //make it a "0" or "1" variable
gen uropreason = 0 if match(urop_reason, "for credit")
replace uropreason = 1 if match(urop_reason, "for credit and pay")
replace uropreason = 2 if match(urop_reason, "for pay")
replace uropreason = 3 if match(urop_reason, "")
gen urop_credit = 0
replace urop_credit = 1 if uropreason == 0

//Employed
replace employed = 1 if employed == .
replace employed = employed-1
gen employed_offcampus = 0
replace employed_offcampus = 1 if employed == 2
gen employed_fulltime = 0
replace employed_fulltime = 1 if employed == 3

//Sleep
replace hourssleep = 6 if hourssleep == .

//Age
replace age = 0 if age == .
replace age = age + 14 if age !=0

//Female
gen female = sex-1
replace female = 0 if female == .
drop sex

//international
gen international= 0
replace international = 1 if citizenship == 3

//futureplan
gen future= shorttermplan
replace future = 8 if future == .


gen sophomore = 0 if year != 1
replace sophomore = 1 if year == 1
gen junior = 0 if year != 2
replace junior = 1 if year == 2
gen senior = 0 if year != 3
replace senior = 1 if year == 3
gen meng = 0 if year != 4
replace meng = 1 if year == 4

gen degree_ee = 0 if degree != 1
replace degree_ee = 1 if degree == 1

gen degree_eecs = 0 if degree != 2
replace degree_eecs = 1 if degree == 2

gen work = 0
replace work = 1 if future == 6 | future == 2 | future == 3
```

rp_cleaning.do

```
//Drop incomplete observations or term is stated as other
drop if difficulty == . & sp_data == 0
drop if teaching == . & sp_data == 0
drop if overall == . & sp_data == 0
drop if workload == . & sp_data == 0
drop if term == "Other" & sp_data == 0
drop if term == "" & sp_data == 0

//Drop observations that have a wrong term assigned to them
drop if lecturer == . & sp_data == 0
drop if overallrating == . & sp_data == 0

//Drop observations that have lecturer == 0
drop if lecturer == 0 & sp_data == 0

drop if rp == . & sp_data == 0
drop if sp == . & sp_data == 1
```

ologit_halton.do

```
capture program drop ologit_halton
program ologit_halton
version 8.1
args todo b lnf
tempvar xb_c xb_rp xb_sp lnfj
tempname cut1 cut2 cut3 cut4 cut5 cut6 rho rho2 mu_sp cons_sp
tempname cov_1_2 cov_3_4 cov2_1_2 cov2_3_4
//Latent_1: Teaching Latent_2: Overall Latent_3: Workload Latent_4: Difficulty Latent_5:
Timing

mleval `xb_c' = `b',   eq(1)
mleval `xb_rp' = `b',   eq(2)
mleval `xb_sp' = `b',   eq(3) scalar

local counter 4

local i 1
while `i'<= 4{
        tempvar xg`i'
        tempname xb`i' xs2_`i' sv2_`i'  xs`i' sv`i'
        mleval `xg`i'' = `b',   eq(`counter')
                local counter = `counter' + 1
        mleval `xs2_`i'' = `b',   eq(`counter') scalar
                local counter = `counter' + 1
        mleval `sv2_`i'' = `b',   eq(`counter') scalar
                local counter = `counter' + 1
        mleval `xb`i'' = `b',   eq(`counter') scalar
                local counter = `counter' + 1

        local i = `i'+ 1
}


local cut_index `counter'

mleval `cov2_1_2' = `b',   eq(`cut_index') scalar
        local cut_index = `cut_index' + 1

mleval `cov2_3_4' = `b',   eq(`cut_index') scalar
        local cut_index = `cut_index' + 1

mleval `mu_sp' = `b',   eq(`cut_index') scalar
        local cut_index = `cut_index' + 1
mleval `rho2' = `b', eq(`cut_index') scalar
        local cut_index = `cut_index' + 1

local j 1
while `j' <= 6{
        mleval `cut`j'' = `b', eq(`cut_index') scalar
        local j = `j' + 1
        local cut_index = `cut_index' + 1
}

tempname A
mat `A' = I(9)
mat `A'[1,1] = `xs2_1'
mat `A'[2,2] = `xs2_2'
mat `A'[3,3] = `xs2_3'
mat `A'[4,4] = `xs2_4'
mat `A'[5,5] = `sv2_1'
mat `A'[6,6] = `sv2_2'
mat `A'[7,7] = `sv2_3'
mat `A'[8,8] = `sv2_4'
mat `A'[9,9] = `rho2'

mat `A'[1,2] = `cov2_1_2'
mat `A'[2,1] = `cov2_1_2'
mat `A'[3,4] = `cov2_3_4'
mat `A'[4,3] = `cov2_3_4'


capture mat `Cholesky_C' = cholesky(`A')

if _rc != 0 {
        //di "Warning: cannot do Cholesky factorization"
}

scalar `xs1' = $Cholesky_C[1,1]
scalar `xs2' = $Cholesky_C[2,2]
scalar `xs3' = $Cholesky_C[3,3]
scalar `xs4' = $Cholesky_C[4,4]
scalar `sv1' = $Cholesky_C[5,5]
scalar `sv2' = $Cholesky_C[6,6]
scalar `sv3' = $Cholesky_C[7,7]
scalar `sv4' = $Cholesky_C[8,8]
scalar `rho' = $Cholesky_C[9,9]
//scalar `xs_rp' = $Cholesky_C[10,10]

scalar `cov_1_2' = $Cholesky_C[2,1]
scalar `cov_3_4' = $Cholesky_C[4,3]


tempvar last
tempvar level1 level2 level3 level4 level5 level6 level7 v_rp v_sp v_common lnfj_S
lnfj_Avg

qui{

        gen double `lnfj_S' = 0
        gen double `lnfj' = 0
        gen double `v_rp' = 0
        gen double `v_sp' = 0
        gen double `v_common' = 0


        gen double `level1'=0
        gen double `level2'=0
        gen double `level3'=0
        gen double `level4'=0
        gen double `level5'=0
        gen double `level6'=0
        gen double `level7'=0

        set seed $S_seed
        local repl=$D_Draws

        local id $Panel
        local sp_data $SP_DATA
        sort `id'
        by `id': gen `last' = _n == _N

        local r=1
        while `r'<=`repl'{

                //Halton sequence are already normally inversed before calling
"ologit_halton"
                //Halton sequence are generated by observation for stdN_1 to stdN_4
                //Halton sequence are generated by id for stdN_5 (one random variable by
id for a given iteration)
                tempvar latent_1 latent_1_dist stdN_1
                tempvar latent_2 latent_2_dist stdN_2
                tempvar latent_3 latent_3_dist stdN_3
                tempvar latent_4 latent_4_dist stdN_4
                tempvar stdN_5


                gen double `stdN_1' = ${HALTON_var1_iter`r'}
                gen double `stdN_2' = ${HALTON_var2_iter`r'}
                gen double `stdN_3' = ${HALTON_var3_iter`r'}
                gen double `stdN_4' = ${HALTON_var4_iter`r'}
                gen double `stdN_5' = ${HALTON_var5_iter`r'}
```

```
            //Calculate latent variables
            gen double `latent_1' = `xg1' + `xs1' * `stdN_1'
            gen double `latent_1_dist' = normden($Teaching,`latent_1',`sv1')


            gen double `latent_2' = `xg2' + `cov_1_2' * `stdN_1' + `xs2' * `stdN_2'
            gen double `latent_2_dist' = normden($Overall,`latent_2',`sv2')


            gen double `latent_3' = `xg3' + `xs3' * `stdN_3'
            gen double `latent_3_dist' = normden($Workload,`latent_3',`sv3')


            gen double `latent_4' = `xg4' + `cov_3_4' * `stdN_3' + `xs4' * `stdN_4'
            gen double `latent_4_dist' = normden($Difficulty,`latent_4',`sv4')



            replace `v_common' =  `xb_c' + `xb1' * `latent_1' + `xb2' * `latent_2' +
`xb3' * `latent_3' + `xb4' * `latent_4'+ `rho' * `stdN_5'
            replace `v_rp' = `v_common' + `xb_rp'
            replace `v_sp' = `v_common' + `xb_sp'

            local l = 1
            while `l'<=6{
                  replace `level`l'' = 1 / (1+exp(-(`cut`l'' - `v_rp')))
      if `sp_data' == 0
                  replace `level`l'' = 1 / (1+exp(-`mu_sp'*(`cut`l'' - `v_sp')))
      if `sp_data' == 1
                  local l = `l' + 1
            }
            replace `level7' = 1 / (1+exp(-(`v_rp'-`cut6')))     if `sp_data' == 0
            replace `level7' = 1 / (1+exp(-`mu_sp'*(`v_sp'-`cut6'))) if `sp_data' == 1

            replace `lnfj' = `level1'        if $ML_y1 == 1
            replace `lnfj' = `level2'-`level1' if $ML_y1 == 2
            replace `lnfj' = `level3'-`level2' if $ML_y1 == 3
            replace `lnfj' = `level4'-`level3' if $ML_y1 == 4
            replace `lnfj' = `level5'-`level4' if $ML_y1 == 5
            replace `lnfj' = `level6'-`level5' if $ML_y1 == 6
            replace `lnfj' = `level7' if $ML_y1 == 7

            tempvar ll_prod       temp_prod measurement_1_prod measurement_2_prod
measurement_3_prod measurement_4_prod
            egen double `ll_prod' = prod(`lnfj'), by(`id')
            egen double `measurement_1_prod' = prod(`latent_1_dist'), by(`id')
            egen double `measurement_2_prod' = prod(`latent_2_dist'), by(`id')
            egen double `measurement_3_prod' = prod(`latent_3_dist'), by(`id')
            egen double `measurement_4_prod' = prod(`latent_4_dist'), by(`id')

            gen double `temp_prod' =
`ll_prod'*`measurement_1_prod'*`measurement_2_prod'*`measurement_3_prod'*`measurement_4_p
rod'

            replace `lnfj_S' = `lnfj_S' + `temp_prod'
            local r = `r' + 1

      }

      gen double `lnfj_Avg' = `lnfj_S'/`repl'

}

mlsum `lnf' = ln(`lnfj_Avg') if `last' == 1
end
```

Appendix G

This appendix[22] is based on [Chen and Popovich, (2002)] and is dedicated for describing statistical tests performed on Pearson correlation r.

Significance of r

One tests the hypothesis that the correlation is zero ($r = 0$) using this formula:

$$t = \frac{r}{\sqrt{\frac{1-r^2}{s-2}}},$$

where s is sample size. The computed t value is compared to the critical t value that can be found in a table of the distribution of t, for $(s - 2)$ degrees of freedom. For large values of s (> 100), if the computed t value is 1.96 or higher, the difference in the correlations is significant at the .05 level.

Z-Score Conversions of Pearson's r

A correlation coefficient r can be transformed into a z-score for purposes of hypothesis testing. This is done by using the following formula:

$$z = \frac{1}{2}\ln\left(\frac{1+r}{1-r}\right),$$

The end result is Fisher's z-score transformation of Pearson's r. Fisher's transformation reduces skew and makes the sampling distribution more normal as sample size increases. z's standard error is given by:

$$\sigma_z = \frac{1}{\sqrt{s-3}}$$

Significance of the difference between two correlations from two independent samples

[22] The notation in this appendix is independent from the general notation listed on page 9.

To compute the significance of the difference between two correlations from independent samples, follow these steps:

1. Convert the two correlations to z-scores as outlined above.
2. Estimate the standard error of difference between the two correlations as:

$$SE = \sqrt{\frac{1}{s_1 - 3} + \frac{1}{s_2 - 3}},$$

   where n_1 and n_2 are the sample sizes of the two independent samples
3. Divide the difference between the two z-scores by the standard error.

$$t = \frac{z_1 - z_2}{SE}$$

If the t value computed in step 3 is 1.96 or higher (if s_1>100 and s_2 >100), the difference in the correlations is significant at the .05 level.

Bibliography

Adomavicius, G., R. Sankaranarayanan, S. Sen, and A. Tuzhilin. Incorporating Contextual Information in Recommender Systems Using a Multidimensional Approach, submitted for publication, 2003.

Alspector, J., Kolcz, A., and Karunanithi,N. Comparing Feature-Based and Clique-Based User Models for Movie Selection. *In Proceedings for the Third ACM Conference on Digital Libraries*, pages 11 –18, 1998

Balabanovic, M. and Shoham, Y. FAB: Content-based collaborative recommendation. *Communications of the ACM*, 40(3), March 1997.

Basu, C., Hirsh, H., and Cohen, W. (1998). Recommendation as Classification: Using Social and Content-based Information in Recommendation. In *Recommender System Workshop '98*. pp. 11-15.

Ben-Akiva, M. and S. Lerman (1985) "Discrete Choice Analysis: Theory and Application to Travel Demand", The MIT Press, Cambridge, MA.

Ben-Akiva, M. and T. Morikawa (1990) "Estimation of switching models from revealed models with randomly distributed values of time", *Transportation Research Record 1413*, 88-97.

Ben-Akiva, M., J. Walker, A.T. Bernardino, D.A. Gopinath, T. Morikawa, and A. Polydoropoulou (2002) "Integration of Choice and Latent Variables Models", *In Perpetual Motion: Travel Behaviour Research Opportunities and Application Challenges* (Elsevier Science, Mahmassani, Ed.), Chapter 21, 431 – 470.

Bentler, P.M. (1980) "Multivariate Analysis with Latent Variables", *Annual Review of Psychology* **31**, 419-456.

Bhat, C. (1999) "An analysis of evening commute stop-making behavior using repeated choice observation from a multi-day survey", *Transportation Research B33*, 495 – 510.

Bhat, C.R. (2000) "Quasi-Random Maximum Simulated Likelihood Estimation of the Mixed Multinomial Logit Model", forthcoming, Transportation Research.

Bhat, C.R. "Quasi-Random Maximum Simulated Likelihood Estimation of the Mixed Multinomial Logit Model", forthcoming, *Transportation Research.*

Billsus, D. and M. Pazzani. User modeling for adaptive news access. *User Modeling and User-Adapted Interaction*, 10(2-3):147-180, 2000.

Bolduc, D. (2003) "Simulation-Based Econometric Estimation of Discrete Choice Models", University Laval.

Breese, J. S., Heckerman, D., and Kadie, C. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence,* pp. 43-52, 1998.

Burke, R. Knowledge-based recommender systems. In A. Kent (ed.), *Encyclopedia of Library and Information Systems.* Volume 69, Supplement 32. Marcel Dekker, 2000.

Chen, P. Y. and P. M. Popovich (2002). *Correlation: Parametric and nnparametric measures.* Thousand Oaks, CA: Sage Publications.

Claypool, M., A. Gokhale, T. Miranda, P. Murnikov, D. Netes, and M. Sartin. Combining content-based and collaborative filters in an online newspaper. In *ACM SIGIR'99. Workshopon Recommender Systems: Algorithms and Evaluation*, August 1999.

Claypool, M., Gokhale, A., Miranda, T. Murnikov, P., Netes, D. and Sartin, M. Combining content-based and collaborative filters in an online newspaper. *ACM SIGIR Workshop on Recommender Systems – Implementation and Evaluation*, August 19, 1999.

ChoiceStream. Comparing Collaborative Filtering and Attributized Bayesian Choice Modeling. *Technology Brief.* *http://www.choicestream.com/pdf/ChoiceStream_TechBrief.pdf*

Dahan E., John R. Hauser, Duncan Simester, and Olivier Toubia, Application and Test of Web-based Adaptive Polyhedral Conjoint Analysis, *Unpublished, May 28 2002.*

Dahan E. and John R. Hauser, The Virtual Customer, *Journal of Product Innovation Management, 2001.*

Dekhytar, A. and Goldsmith, J. The Bayesian Advisor Project. *Unpublished*, March 15 2002.

Gould, W., J. Pitblado, and W. Sribney (2003). *Maximum Likelihood Estimation with Stata*, Second edition, Stata Press Publication.

Gourieroux, C. and A. Monfort (1996) *Simulation-Based Econometric Methods*, Oxford University Press.

Hensher, D. and M. Bradley (1993), “Using stated response data to enrich revealed preference discrete choice models”, *Marketing Letters 4*, 39 – 152.

Hensher, D., J. Louviere, and J. Swait (1999), “Combining sources of preference data”, *Journal of Econometrics 89*, 197 – 221.

Herlocker, J., Konstan, J., Borchers, A., and Riedl, J. (1999). An Algorithmic Framework for Performing Collaborative Filtering. In *Proceedings of ACM SIGIR'99*. ACM press.

Huber, J. Train, K. On the Similarity of Classical and Bayesian Estimates of Individual Mean Partworths. Marketing Letters 12:3, 259-269, 2001.

Keesling, J.W. (1972) *Maximum Likelihood Approaches to Causal Analysis*, Ph.D. Dissertation, University of Chicago.

Louviere, J.J., D.A. Hensher and J.D. Swait (2000) *Stated Choice Methods: Analysis and Application*, Cambridge University Press.

McFadden, D. (1974) “Conditional Logit Analysis of Qualitative Choice Behavior”, *Frontiers of Econometrics*, P. Zarembka, Ed., Academic Press.

McFadden, D. (1989) “A Method of Simulated Moments for Estimation of Discrete Response Models without Numerical Integration”, *Econometrica 57(5)*, 995 - 1026.

McFadden, D. and K. Train (2000) “Mixed MNL models of discrete response”, *Journal of Applied Econometrics* 15, 447 - 470

Miller, B. N., J. A. Konstan and J. Riedel (2004) “PocketLens: Toward a Personal Recommender System”, *ACM Transactions on Information Systems, Vol. 22, No. 3, July 2004,* 437 – 476.

Mooney, R. J. Content-based book recommending using learning for text categorization. In *ACM SIGIR'99. Workshop on Recommender Systems: Algorithms and Evaluation*, 1999.

Olivier Toubia, Duncan I. Simester and John R. Hauser "Fast Polyhedral Adaptive Conjoint Estimation". Unpublished, February 2002.

Personalization Consortium. http://www.personalization.org/personalization.html

Pazzani, M. A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review*, pages 393-408, December 1999.

Resnick, P. and Varian, H. Recommender systems. Communications of the ACM, 40(3):56-58, 1997.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J. (1994). GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proceedings of CSCW '94*, Chapel Hill, NC.

Revelt, D. and K. Train (2000), 'Specific taste parameters and mixed logit', *Working Paper No. E00-274*, Department of Economics, University of California, Berkeley.

Russel, S. Norvig, P. Artificial Intelligence: A Modern Approach. Prentice Hall series in Artificial Intelligence, 1995.

Sarwar, B. M., Karypis, G., Konstan, J. A., and Riedl, J. (2000). Analysis of Recommendation Algorithms for E-Commerce. In *Proceedings of the ACM EC'00 Conference*. Minneapolis, MN. pp. 158-167

Schafer, J. B., J. A. Konstan, and J. Riedl. E-commerce recommendation applications. *Data Mining and Knowledge Discovery*, 5(1/2):115-153, 2001.

Shardanand, U., and Maes, P. (1995). Social Information Filtering: Algorithms for Automating 'Word of Mouth'. In *Proceedings of CHI '95*. Denver, CO.

Skrondal, A. and S. Rabe-Hesketh (2004) "Generalized Latent Variable Modeling", Chapman & Hall/CRC.

Ungar, L. H., and Foster, D. P. Clustering Methods for Collaborative Filtering. In *Workshop on Recommender Systems at the 15th National Conference on Artificial Intelligence, 1998.*

Train, K. (2003) *Discrete Choice Methods with Simulation*, Cambridge University Press.

Tran, T. and R. Cohen. Hybrid Recommender Systems for Electronic Commerce. In *Knowledge-Based Electronic Markets, Papers from the AAAI Workshop, Technical Report WS-00-04*, AAAI Press, 2000.

Walker, J. (2001) "Extended Discrete Choice Models: Integrated Framework, Flexible Error Structures, and Latent Variables", Ph.D. Dissertation, Department of Civil and Environmental Engineering, Massachusetts Institute of Technology.

Wiley, D.E. (1973) "The Identification Problem for Structural Equation Models with Unmeasured Variables", *Structural Models in the Social Sciences*, A.S. Goldberger and O.D. Duncan, Eds., Academic Press, New York.