

Recommendation of Items with Inter-Dependencies: A Course Plan Recommender System

Amjad Almahairi

Master of Science

School of Computer Science

McGill University

Montreal, Quebec

2011-07-15

A thesis submitted to McGill University in partial fulfilment of the requirements of
the degree of Masters of Science

©Amjad Almahairi, 2011

DEDICATION

To friends and all people of Syria who are struggling for their basic rights of dignity and freedom, my parents, and my dear wife.

ACKNOWLEDGEMENTS

First and foremost I would like to thank my advisor, Professor Gregory Dudek, for his direction, great ideas, support, encouragement, and above all his kindness and gentle demeanor. He has provided an excellent example of a successful professor and supervisor. No words that can express the gratitude I feel to my dear parents, who have sacrificed a lot to help their son be a successful person. Thanks to my beloved wife. I could not have accomplished so much without her support and patience. Thanks to my dear sister, who has been always there to support and encourage me. Finally, I would like to thank everyone in the Mobile Robotics Lab for the enjoyable time I spent with them, their suggestions especially on the web survey, and the nice discussions we had throughout our lab meetings.

ABSTRACT

In this thesis we address the problem of recommendation in domains where items have strong dependency constraints. We apply our work on the academic courses domain, where dependencies between items are present as explicit prerequisite constraints, implicit ordering patterns, and general course consumption restrictions. We propose a new recommendation approach that combines both ordering patterns of items learned from data and estimated user interests in providing a personalized *sequence* of recommendations that take into account item inter-dependencies. Our approach is based on modeling the recommendation problem as a Markov Decision Process (MDP), in which the goal is to find a plan with the maximum total reward. We have developed an implementation of our approach as a web-based course recommender system, which recommends academic course plans to students for a number of subsequent terms. Experiments on real data collected from students of McGill University demonstrate that our approach has better performance compared to other simpler models which rely exclusively on student interests or on course co-occurrence patterns.

ABRÉGÉ

Dans cette thèse, nous abordons le problème de recommandation dans les domaines où les éléments ont des fortes contraintes de dépendance. Nous appliquons notre travail sur le domaine des cours académiques où les dépendances entre les éléments sont présentes comme contraintes de condition préalable explicites, pattern de l'arrangement implicite, et restrictions générale de la consommation de cours. Nous proposons une nouvelle approche au problème de recommandation qui combine l'arrangement des articles appris à partir des données et les intérêts de l'utilisateur, estimés pour fournir une séquence personnalisée des recommandations, qui prend en compte les interdépendances des éléments. Notre approche est basée sur la modélisation du problème de recommandation comme processus de décision Markovien (MDP) dont le but est de trouver un plan à profit totale maximale. Nous avons implémenté notre approche sur le Web en tant qu'un système qui propose des plans de cours aux étudiants pour un certain nombre de périodes subséquentes. Nos expériences sur des données réelles recueillies auprès des élèves de l'Université McGill démontrent que notre approche est plus performante comparativement aux modèles qui ne considèrent que les intérêts d'étudiant ou les patterns de co-occurrence de cours.

TABLE OF CONTENTS

DEDICATION	ii
ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
ABRÉGÉ	v
LIST OF FIGURES	viii
1 Introduction	1
2 Related Work	7
2.1 Traditional Recommendation techniques	9
2.2 Sequence Recommendation	13
2.3 Course Recommender Systems	14
2.4 Recommendation with constraints	15
2.5 Temporal Modeling in Recommendations	17
2.6 MDPs in Recommendation	18
3 Course Plan Recommendation Problem	20
3.1 Notation	20
3.2 Problem Definition	22
3.3 Example	22
4 Course Reward Prediction	25
4.1 Rewards Based on Transcript Similarity	26
4.2 Rewards Based on Course Description Similarity	28
5 The MDP Course Planner	30
5.1 MDP Model Components	30

5.1.1	State space S	30
5.1.2	Action set A_s	32
5.1.3	Transition function $Tr(s, a)$	32
5.1.4	Reward function $R_T(s)$	33
5.1.5	Value function $V_T^\pi(s)$	33
5.2	Building the Model	34
5.3	Course Planning as Solving the MDP	34
6	Implementation Details	36
6.1	Course Plan Recommendation System	36
6.1.1	Recommendation Package	37
6.1.2	Database	39
6.1.3	Presentation	41
6.2	Transcripts Survey	42
7	Experimental Evaluation	45
7.1	McGill Transcripts Dataset	45
7.1.1	Data Sparsity and State Pruning	46
7.2	Course Reward Prediction Evaluation	49
7.2.1	Performance Measure	49
7.2.2	Experimental Procedure	50
7.2.3	Results and Discussion	51
7.3	Evaluation of Course Plan Recommendations	54
7.3.1	Performance Measures	54
7.3.2	Experimental Procedure	56
7.3.3	Results and Discussion	58
7.3.4	Experimental Evaluation of α and β Thresholds	64
8	Conclusions	69
8.1	Summary	69
8.2	Future Directions	70
	References	74

LIST OF FIGURES

<u>Figure</u>	<u>page</u>
3-1 Example of a student's transcript.	21
3-2 Example of the course plan recommendation problem.	23
6-1 Overview of the course plan recommendation package	37
6-2 Overview of the course plan recommendation package	40
6-3 System's presentation	43
6-4 Screenshot of a parsed transcript on the survey	44
7-1 Dataset sparsity illustrated by the number of courses or states that occur with certain frequencies in the dataset.	47
7-2 Reward prediction performance	52
7-3 Average values of known ratings for each of the nine values of grades in all transcript entries of the dataset.	53
7-4 Average <i>accuracy</i> and <i>coverage</i> vs. number of known terms when using state-based pruning (SBP) with $\alpha = 1$	60
7-5 Average <i>accuracy</i> and <i>coverage</i> vs. number of known terms when using course-based pruning (CBP) with $\beta = 3$	61
7-6 Average <i>accuracy</i> and <i>coverage</i> vs. length of plan in terms when using state-based pruning (SBP) with $\alpha = 1$	62
7-7 Average <i>accuracy</i> and <i>coverage</i> vs. length of plan in terms when using course-based pruning (CBP) with $\beta = 3$	63
7-8 Alpha values effect on average <i>accuracy</i> and <i>coverage</i> vs. number of known terms when using state-based pruning (SBP)	66

7–8	Beta values effect on average <i>accuracy</i> and <i>coverage</i> vs. number of known terms when using course-based pruning (CBP)	67
-----	--	----

CHAPTER 1

Introduction

In the presence of today's enormous volume of on-line content and media collections, using technologies that can process huge amounts of information and help us make right decisions has become inevitable. Researchers refer to the phenomenon of having an excessive amount of information that cannot be understood or taken advantage of by the *information overload* problem.

Recommender Systems tackle the information overload problem by providing individuals with personalized suggestions or *recommendations* to help them choose items that match their needs and interests. They draw an analogy from real life experience, where people ask their friends' opinion to decide what books to read or movies to watch.

Traditional recommendation techniques operate mainly by estimating a *rating* for each item, which is a numerical value that reflects how much the user likes the item. Item ratings are then used to select an item or a set of items with highest ratings as recommendations. These techniques have proven to work very well in recommending movies, news, music and other domains where satisfying user preferences is the main objective. In domains where items are subject to ordering constraints, however, recommending an item that violates the constraints is infeasible even if we expect the user to like it. For example, recommending the last episode of a drama

series will be useless to a person who has no idea about the the whole story or flow of events.

In this thesis we address the problem of generating recommendations for items with inter-dependencies in the sequence that they should be consumed. We focus on the domain of academic courses as our application. Our approach, however, applies to many other domains where items have sequential nature, such as novels, TV series, and website links. Academic courses enjoy strong dependencies that can take various forms. First, dependencies are present as “mandatory” course prerequisites which compel certain ordering in taking courses (e.g. the course “Programming I” is a prerequisite for the course “Programming II”). Another form is what we like to call *soft-prerequisites*, which are implicit course orderings that is proven to be advantageous to students (e.g. Discrete Structures course is highly recommended to be taken before Algorithms course, but not required). In addition, there could be other general constraints that are not included in any written format (e.g. taking four heavy programming courses is not a good idea). Therefore, recommending academic courses needs to take into account these dependencies in addition to the user’s interests in order to provide useful and feasible recommendations. Besides being an important instance of the class of items with inter-dependencies, academic course advising in general is a very complex problem which requires in-depth domain knowledge of course choices and their constraints, in addition to understanding the student’s interests and strengths. Hence, solving this problem has advantages from both theoretical and applied perspectives.

In domains where items have strong inter-dependencies it is not sufficient to provide the user with the top-N items even if they conform to dependency constraints. This applies to even the broader case of items that are consumed in a sequential manner, because the *order* in which items are consumed affects the degree of user satisfaction [29]. Therefore, we will try in this thesis to answer the question: “What is the sequence of courses that student x should take in the subsequent terms?”. This is a special and less studied class of problems in recommender systems literature called *sequence recommendation* [18].

The goal of this thesis is to recommend academic course plans to students based on their interests. We define a *course plan* formally as a t -length sequence of sets of k courses. Therefore, given the student’s course history, we recommend a plan of k courses per term for the following t terms. A student’s interest in an individual course is represented as a numerical value in the range of least interesting to most interesting, and the interest in a course plan is modeled as the sum of interests in courses that it is composed of. We provide formal definition of the course plan recommendation problem in Chapter 3.

The algorithm for recommending course plans balances two objectives: 1) the plan should fit the student’s interests; 2) it must respect ordering constraints between courses. In fact, these two objectives can conflict with each other. Choosing the best set of courses in terms of user preference for each successive academic session can violate long-term ordering constraints, because it might ignore uninteresting prerequisites although they are required for later interesting courses (e.g. the student might be very interested in a Robotics course but hates Linear Algebra, which is a

prerequisite). Therefore, greedily maximizing the interest value of current courses must be traded-off against respecting ordering constraints by recommending less desirable courses in earlier stages of the plan. We do *not*, however, adopt an approach that strictly satisfies prerequisites. We argue that holding tightly to these constraints is an impractical decision, since transcripts of real students often violate prerequisites (as we show in section 7.3.3). This is due to several factors: prerequisite constraints do not count for equivalence of courses in terms of satisfying prerequisites, they might be changed with different instructors, and they can be overlooked in certain circumstances. We rather adopt on a more practical solution to this problem that avails of course co-occurrence patterns, learned from real students’ transcripts, to enforce the choice of next courses that conform to a high degree to hard and soft prerequisites. Although co-occurrence patterns are not expected to fully capture real-world ordering of courses, we show in this thesis that they provide an efficient and powerful tool for capturing course dependencies.

As we show in Chapter 5, our overall approach is based on modeling the course plan recommendation problem as a Markov Decision Process (MDP), which is a standard model for probabilistic sequential optimization [38]. We define the states that our course planner can be in at any academic term (time step or decision epoch) by the set of the courses that the student has taken in that term. In each term, and based on the current state of the system, a set of courses (an action) is chosen based on maximizing the total reward for the whole plan. In contrast to the standard stochastic MDP model, our planner evolves to the next state deterministically after taking a certain action. This classifies our model as a Deterministic MDP (or

Deterministic Dynamic Program as in [38]). However, we use the notion of transition function between any two states to enforce the logical ordering between the two sets of courses. Therefore, by finding an optimal policy for this MDP model, we find the plan that has the maximum total reward, and respects logical ordering of courses. The main limitation that MDP approaches suffer from is the high dimensionality of solution space. We overcome this limitation by a careful choice of state-space and maintaining only the portion of it that we encounter in training data.

The recommended course plan is composed entirely of courses that the target student has not taken before. This raises the question on how to specify the rewards for these individual courses to the target student. We propose a solution to this problem in Chapter 4 by estimating the student’s interest in an individual course (course reward) based on reported knowledge of interest in previously taken courses. Our method is based on using a hybrid approach of traditional collaborative filtering, which matches the target student to other students with similar tastes, and content-based filtering, which is based on analyzing textual description provided with each course.

In Chapter 6 we describe the implementation of our approach as a course plan recommendation system. We have developed and deployed a system that recommends course plans to Computer Science and Software Engineering students at McGill University. Additionally, we discuss in this chapter the data collection of student transcripts through a web survey. We have used this transcript database for both the deployment of our recommender system and in the evaluation of our approach.

In Chapter 7 we discuss the experimental evaluation of our approach based on the transcripts collected from students of McGill University. The results demonstrate that our approach improves the quality of course plans that rely exclusively on course rewards or on course co-occurrences.

Finally, in Chapter 8 we conclude our work with a summary of the key points of this thesis and the important observations from the experimental results, and discuss directions for future work.

CHAPTER 2

Related Work

The general recommendation problem can be decomposed into two subtasks: rating prediction and item selection. *Rating prediction* assigns scores or ratings to potential items that reflect the target user’s preferences. *Item selection*, on the other hand, uses the estimated ratings, along with other criteria, for choosing items to be recommended to the target user. The majority of work in recommender systems literature deals with item domains that impose no restrictions on consuming items, such as movies, music tracks, and books [1]. In other words, items in such systems can be recommended independently, or without the need to check back with each other for satisfying any constraints. As a result, the primary focus in the literature is on offering various techniques for modeling user preferences and improving rating prediction, as item selection can be trivially accomplished in this case by sorting items according to their ratings.

In this thesis, we are interested in recommendation of items that are subject to ordering constraints in general, but we focus on academic courses as the application domain for our proposed approach. This leads to a special class of methods, where the goal is not only to recommend items that maximize user interests, but also respect these constraints. Two general approaches for recommendation in such domains were previously introduced in the literature:

- *Rating modification*: The score given to an item based on user preferences is increased or decreased according to how much it conforms to item dependencies. Hence, the final score given to an item represents its “utility” to the user rather than its mere “desirability”. For instance, an item given a high rating according to the preferences of the user could get a low utility score if it has unfulfilled dependencies. Consequently, items are selected based on a new score that takes into account item dependencies.
- *Constrained selection*: The item selection process takes other factors into account, aside from the estimated user ratings, in choosing items for presentation. Therefore, item ratings represent only the user’s preferences, and other constraints are taken into account to prune out choices that violate supplementary constraints. Although this problem seems easy for individual items, it is a hard problem when the goal is to recommend a “bundle” of items which have fulfilled dependencies and maximum sum of scores as shown in [34].

These two approaches are not necessarily mutually exclusive. The approach introduced in this thesis is based on assigning utility scores to items based on user preferences and item inter-dependency constraints. In addition, the user is provided with a *sequence* of items, which is ordered such that dependency constraints on items are respected.

We provide in Section 2.1 a brief description of traditional recommendation techniques which offer different approaches for estimating item ratings for a target user. This discussion is important in our work since we apply some of these techniques in assigning user desirability scores (rewards) to items regardless of their

inter-dependencies (see Chapter 4). In addition, we discuss some of the key relevant work from different perspectives to our problem and approach. The first is the problem of generating an *ordered* sequence of recommendations to the user (Section 2.2), as our goal is to recommend a *sequence* of courses. In Section 2.3 we discuss some key previous research on the academic course recommendation problem. In Section 2.4 we present related work discussing recommendation under constraints, as item inter-dependencies are a special class of constraints on the recommendation process. In Section 2.5 we present work done on using temporal information in making recommendations, and finally in Section 2.6 we present some key recommender systems that used MDPs in generating recommendations.

2.1 Traditional Recommendation techniques

Traditional approaches in recommender systems offer various techniques for rating prediction, but they fall mainly into three categories: content-based filtering, collaborative filtering and hybrid methods.

Content-based filtering relies on internal features of items in estimating their ratings. This approach is based on analyzing the content of items in order to capture only important features. Recommendation in content-based recommender systems is based on a simple heuristic that users will like items similar to what they liked in the past. For example, if Bob has expressed interest in action movies, we would expect him to like “The Dark Knight” movie. The first step of recommending items is to build an *item representation* that models items by a list of features. In addition, features of items favored by the user are used to learn a *user profile* is learned that models user preferences. In the movie recommendation example, a movie could be

represented by the list of genres it can be categorized to, its actors, director, ...etc. Additionally, the user profile models frequent genres or actors found in the user's favored movies. The second step is computing a feature-based distance metric that measures similarity between items and the user profile. Therefore, new items are recommended to the user if they are similar to the user's profile.

In order to build an item representation, content-based recommender systems assume that items are described by a rich information source from which features can be extracted. Most content-based recommender systems deal with items (or item descriptions) that are available in text format, such as news articles [27, 5] and web pages [36]. Extracted features in this case are important *keywords*. The simplest item representation is the Vector Space Model, which models each item as a point in an n -dimensional feature space. In this model, each keyword of the overall vocabulary of items represents a dimension of the space, and the weight represents how important is the keyword to the item. Keyword weights are often computed using the Term Frequency-Inverse Document Frequency (TF-IDF) metric [39], which measures the association between an item (document) and a keyword (term) by the number of times the keyword appears in the item (term frequency), but de-emphasising keywords that might appear frequently in all items (inverse document frequency).

Additionally, several approaches were introduced in the literature for modeling and learning user profiles [37]. One approach is based on modeling the user profile as an average weight vector of favored item vectors, which is computed using a variation of Rocchio algorithm method as in [3, 27]. Items are then matched with the user

profile using some similarity measure, such as cosine similarity. The weight vector representing user profile can be modeled also as a linear classifier, whose dot product with item vectors gives a predicted rating for corresponding items [47]. Another approach models the profile as a collection of past item vectors used as labeled samples for k -nearest neighbor method [6].

Many applications may have limited amount of information on the content of items, or it might be very hard to extract useful features from item content (e.g. images and music). Furthermore, content-based filtering has a major drawback of *over specialization*. Consistently recommending similar items to the current interests reduces the novelty of the recommendations, although they might be consistent with the user’s known interests. This specific issue is addressed with the concept of “serendipity” in recommendations, where the goal of serendipitous recommendations is to provide unexpected but interesting items.

The second class of methods, *collaborative filtering*, can potentially provide serendipitous recommendations because they are based on the experience of other users. Collaborative filtering makes use of the choices of other humans regarding items rather than item contents in rating prediction. In the movie example, ratings that users give to movies are used rather than movie descriptions in predicting a new movie rating. There are two primary approaches for collaborative filtering: *neighborhood-based* (memory-based) methods and *model-based* methods.

Neighborhood-based approaches find a neighborhood set from the entire collection of other users’ ratings by employing statistical techniques, and then use it in estimating ratings for the new user. The neighborhood set can be established

either in the user domain or in the item domain. *User-based collaborative filtering* (e.g. [16, 24, 19]) finds a neighborhood set of users to the target users. It is based on the intuitive idea that a user will like what other “similar” people have liked. Similarity between users is inferred based on the how much their ratings agree the same set of items. For example, if Bob and Alice have consistently expressed similar interests in movies -even if these movies have no common theme-, we induce that they have similar taste in movies and that Bob will most likely be interested in a new movie that Alice have liked. On the other hand, *item based collaborative filtering* (e.g. [41, 8]) finds a neighborhood set of items for a potential item. It is based on the same heuristic as content-based filtering that similar items will receive similar responses. Here, however, similarity between items is based on the degree to which user opinions agree about them, i.e. two items are considered similar if they have received similar ratings. Therefore, the two movies “The Dark Knight” and “Batman Begins” would be deemed similar just because they have received similar ratings.

Model-based approaches learn an off-line model that captures associations between users and items. Several machine learning techniques have been used in this context, such as Graphical Models [17], Linear Regression [42], and latent factor models such as: Latent Semantic Analysis (LSA) [21], Probabilistic LSA [20], and most recently an improved Matrix Factorization Model called SVD++ [26] which was introduced in the winning solution to the Netflix competition.¹

¹ A global competition for improving prediction accuracy of user ratings for movies based on previous ratings by 10%. <http://www.netflixprize.com/>

The third class of traditional recommender systems approaches is *hybrid* methods, which are based on the combination of the above two mentioned approaches. Burke [7] discuss several methods for combining recommendation techniques. The goal of hybrid techniques is mainly to overcome some limitations of one method by using the other method. For example, collaborative filtering suffers from the “new-item problem”, i.e. it cannot recommend items that have not been rated before. Content-based filtering can help in this case, as new items are recommended if they are similar to one previously rated item. Previous systems such as Fab [3] and Recommendz [14] provide good examples for combining content-based and collaborative filtering.

2.2 Sequence Recommendation

Most of the work in previous recommender systems focused on generating recommendations as single item or a set of items. The goal in sequence recommendation, however, is to recommend a list of items that is expected to be pleasing for the user if taken in order. Sequence recommendation has not been frequently addressed in the literature. Masthoff showed that the order of items of the recommended set in domains that have inherent ordering between items (e.g. TV series) affects user satisfaction [29], and thus recommending a sequence of items is more appropriate in such domains. The work of Masthoff [30] focuses on presenting a sequence of interactive TV items (e.g. news items), but its main concentration is on targeting a group of individuals, combining information from all group members, and selecting items that satisfy the targeted group as a whole. Methods used for ordering items in the recommended sequence are either based on heuristics such as having good narrative

flow, consistency in viewer mood, and strong ending, or based on interacting with viewers and reordering items according to their feedback.

When items are subject to ordering constraints, recommending a sequence of items becomes more evident as the sequence represents the order in which items *must* be taken rather than *should* be taken, and hence we adopt that in our approach. Likewise, Ge et. al. [15] recommend a sequence of pick-up locations for taxi drivers. Pick-up locations are subject to distance constraints, as the goal is to provide a sequence with minimized traveling distance. Selecting routes in their work, however, is based on optimizing business success rather than satisfying user opinion. An important commonality with our work is the combinatorial size of the solution space, which was addressed in their work by using several pruning strategies.

2.3 Course Recommender Systems

Recommender systems technology has been recently employed in the education domain, where the goal was primarily recommending academic courses, and more generally recommending educational resources. Most of this work, however, does not address item inter-dependencies. O'Mahony and Smyth [32] present an item-based collaborative filtering algorithm for recommending elective courses to students based on the core courses they have selected. In addition, CourseAgent system [12] adopts a social navigation approach for course recommendation. On the contrary to the general trend of recommender systems, recommendations in CourseAgent are provided as visual cues that help students in selecting courses rather than a set of recommended courses. Courses are tagged with community-based annotations in the form of expected workload of the course and expected relevance to the student's

career goals. AACORN system [40] employs a case-based reasoning approach for course recommendation. Recommendations are built from other student transcripts (cases), which are retrieved based on their similarity to the target student’s partial transcript. RARE system [4] adopts a data mining approach similar to the MDP model building step of our approach (see Section 5.2). RARE system learns from previous course histories a set of association rules, which represent courses that are often taken simultaneously. Recommendations are provided based on the similarity between courses taken by the students and antecedents of the rules. Similarity between courses in RARE, however, does not take into account student ratings or grades. The inferred rules are improved in time by modifying them according to student feedback on the provided recommendations.

Academic course recommendation has recently received special attention in the context CourseRank² project. The problem has been studied from various perspectives [34, 35, 33], which we will discuss in the following sections.

2.4 Recommendation with constraints

Recommendation with constraints differ from the typical recommendation problem in that it must satisfy multiple objectives rather than a single one related to the interests of the user. In the academic courses domain, the work of Parameswaran et. al. [34] addresses the same problem of recommending courses that take into account prerequisites. The authors, however, address the problem as an object selection task. The goal of their work is to recommend a set of k courses such that

² <http://www.courserank.com>

their prerequisites are already satisfied or otherwise included in the recommended set. The authors prove that this problem is NP-Hard via reduction from set cover problem, and provide polynomial time approximation algorithms for it. In contrast to their work, we do not strictly satisfy prerequisite constraints (as we mentioned in Chapter 1). We rather adopt an approach that relies on course co-occurrence patterns that can capture both hard-prerequisites and soft-prerequisites.

Felfernig and Burke [13] discuss a special type of recommender systems called *Constraint-based recommendation*. This technique deals with recommending products or services that are subject to complex constraints, but particularly ones that may neither have large user communities nor long purchase histories. Jannach et. al. [23] discuss applying this technique in recommending travel plans, where constraints are present in destination selection, transportation and accommodation arrangements, and price or time limits. In such domains, products that satisfy the set of constraints are considered good recommendations and the problem takes on aspects of traditional resource-constrained planning. Therefore, the recommendation task is modeled as a constraint satisfaction problem. This technique follows a knowledge-based approach, which exploits domain expertise rather than learning patterns of user preferences from data, and hence it suffers from the typical knowledge acquisition bottleneck. Our approach, in contrast, distinguishes between user interest in items and external constraints that should be satisfied to generate feasible recommendations.

Jambor and Wang [22] follow a rating modification approach when dealing with recommendation under constraints. They model the rating prediction problem as

a multiple objective optimization problem, where prediction accuracy is the main objective and other domain-specific objectives, such as recommending availability in stock and profitability, as the problem constraints. They apply this approach to address the “long-tail” problem [2] by adding constraints that degrade popular items.

2.5 Temporal Modeling in Recommendations

Leveraging sequential patterns has been studied in recommender systems, especially in the domain of Web personalization, as an extension to a large body of work on Web usage analysis [31]. In the context of “next page prediction” problem [9], Deshpande and Karypis propose using user web surfing behavior in making predictions on the next page to be accessed in web browsing. Their approach is based on modeling the user’s surfing path as a Markov chain, and then assigning probabilities for following pages. In recommender systems context, Zimdars et. al. [48] consider the order in which items were consumed in predicting user preferences. The temporal information in these applications is used as an alternative for explicit ratings. Other approaches apply time weighting schemes on rating data for computing similarity between users in a collaborative filtering [10]: as time difference increases between previously rated items, the contribution to similarity measure decreases. Another interesting research direction has considered improving the performance of collaborative filtering by studying temporal changes in user taste [25].

The precedence mining model [35] relies on precedence patterns that users tend to follow in consuming items (taking courses) to predict future items. More specifically, the probability of taking a course is estimated given previously taken courses, and items with the highest probabilities are recommended. The advantage of this

idea over traditional collaborative filtering is that it leverages precedence patterns over all users, not only users with similar course histories. Nevertheless, it only uses previous courses in identifying future courses, and ignores explicit information about the user’s preferences (e.g. course ratings). Our work, on the other hand, takes into account course ratings in identifying future courses the user might be interested in. Additionally, unlike the prior work, we provide the user with a full course plan for subsequent future terms, and use the precedence information in order to enforce the right order for courses in the plan.

2.6 MDPs in Recommendation

The MDP framework has been previously applied in recommender applications [44, 46]. These applications provide a different perspective to the general recommendation problem as a dynamic process, where the effect of each recommendation on the user’s choice is modeled, and both immediate and future returns are taken into consideration in predicting one next step. The model is also updated based on actual observations either periodically [44] or interactively using reinforcement learning [46]. Our work, on the other hand, uses the Markov Decision Process (MDP) model in constructing a whole plan rather than a one-step prediction. More importantly, we avail of the MDP model in designing course plans by making sequential decisions that maximize both the immediate rewards and long-term effects. The designed plans maximize an objective function that satisfies both user interests and course ordering constraints based on off-line experience rather than interacting with the user.

Conversational recommender systems establish dialogues between the user and the system in order to extract user preferences, but they usually apply fixed strategies for interaction. Mahmood and Ricci [28] use an MDP model and apply an adaptive Reinforcement Learning strategy for optimizing the dialogue and therefore building a better recommendation strategy.

CHAPTER 3

Course Plan Recommendation Problem

3.1 Notation

We start by defining the notation we use in the rest of this thesis. Let \mathcal{C} be the set of all courses that we have available, where $|\mathcal{C}| = N$. Let \mathcal{T} be the set of transcripts of previous students. A *transcript* T denotes the course history of a given student and is composed of a sequence of sets of (course, rating, grade) triples such that each set corresponds to courses the student has taken in a single term¹. A rating $r_c \in [r_{min}, r_{max}]$ that the target student gives for course $c \in \mathcal{C}$ is a value that expresses how much the student likes the course, where r_{min} represents the value of least interesting and r_{max} the most interesting. In addition, g_c is the grade that the student receives in course c . In fact, the only information about the student we are interested in is the transcript, so we will refer primarily to the transcript of the target student instead of the student in the rest of the thesis. Figure 3–1 shows an example of a transcript composed of three terms, where course objects are referred to by their codes and arrows denote the sequential relationship between terms.

A course plan P that we would like to recommend is a t -length sequence of sets of k courses. Figure 3–2a shows examples of course plans with $t = 2, k = 2$. Allowing

¹ We might abuse this notation and deal with the transcript as a set of its constituent courses rather than a sequence of sets.

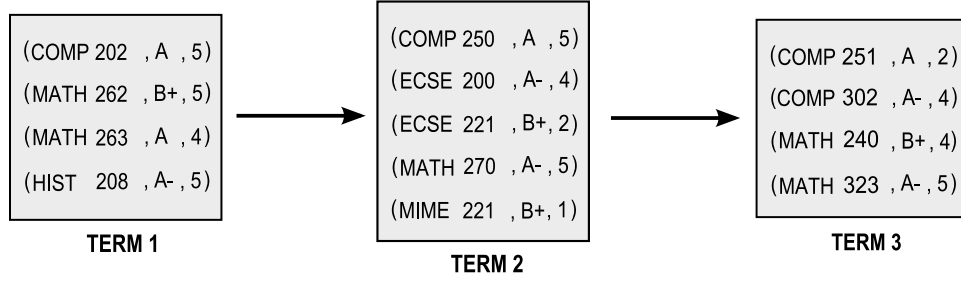


Figure 3-1: Example of a student’s transcript. Each box represents a set of courses taken in a single term, where the course is attributed with its code, received grade, and rating. Arrows denote the sequential relationship between terms.

a variable number of courses per term in the plan makes the problem much harder, as the number of possibilities increase exponentially. We discuss that in detail in Section 5.1.1.

A prerequisite of a course c which imposes specific constraints on taking the course is a logical expression composed of “AND” and “OR” operators and other courses $c' \in \mathcal{C} \setminus c$ that taking course c depends on as the expression’s operands. These course operands take the value *true* if they are taken by the student in a previous terms, and *false* if not, and the prerequisite expression evaluates to *true* or *false* accordingly. For example, the course “Artificial Intelligence” or COMP 424 has the prerequisite expression: “(COMP 206 or ECSE 321) and COMP 251”. If the student took the course COMP 206 in the term 2, then the prerequisite expression evaluates to *true* when COMP 424 is taken in term 3 or any *later* term. We say that a prerequisite for a given course is *satisfied* if its logical expression evaluates to *true*.

Finally, we say that a course plan P *fully* respects the ordering of courses or ordering constraints if *all* prerequisite expressions evaluate to *true*, i.e. they are all satisfied.

3.2 Problem Definition

We formally define in this section the problem of course plan recommendation. Given the set of transcripts \mathcal{T} , and the transcript T of a target student, we wish to devise a course plan P , consisting of k courses per term for the following t terms, such that P balances the following two objectives:

1. It has a maximum reward for the student.
2. It fully respects ordering constraints on courses.

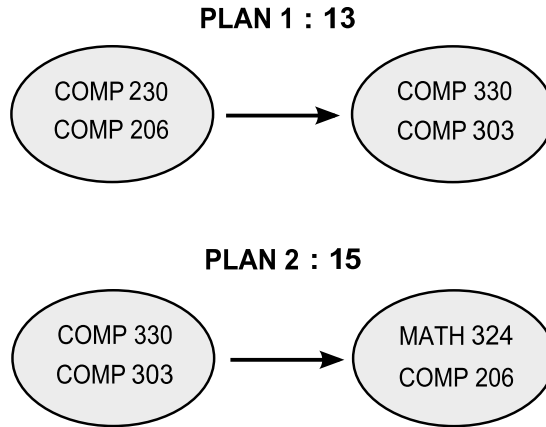
We define reward for plan P to a student with transcript T , or $R_T(P)$, by:

$$R_T(P) = \sum_{c \in P} R_T(c) \quad (3.1)$$

Where the reward $R_T(c)$ for an individual course c is a value that represents how “good” c is for the target student with transcript T . Since this value is *not* known *a priori* for every possible student-course pair, the first step in our approach to this problem is to estimate this value for each pair based on the information we already know about both the student and the course. Estimating rewards for a course c to a target user with transcript T includes two main factors: how much this course was liked by other similar users, and how much the user liked other courses of similar content. We discuss that in detail in Chapter 4. In addition, as mentioned earlier in Chapter 1, our goal is to balance these two objectives and *not* to fully satisfy them, as they can be conflicting with each other.

3.3 Example

Figure 3-2 illustrates a simple example of course plan recommendation to the transcript shown in Figure 3-1. The table shown in Figure 3-2b gives an example



(a) Example of two possible course plans with number of terms $t = 2$ and, courses per term $k = 2$ and their total rewards

COMP 206	3
COMP 303	4
COMP 330	5
COMP 230	1
MATH 324	3
MATH 235	2

(b) Predicted rewards for courses

Course	Prerequisites
COMP 303	COMP 206 AND COMP 250
COMP 206	COMP 202 OR COMP 250
MATH 324	MATH 323
COMP 330	COMP 251

(c) Hard prerequisites

Figure 3-2: Example of the course plan recommendation problem.

of a set of potential courses with their predicted rewards. The table in Figure 3-2c shows course prerequisites that are related to this example.

Reward prediction for potential courses is the first step performed to recommend course plans, because deciding on which plan to recommend for a specific user depends on estimated rewards for courses. Figure 3-2a shows two possible plans with number of terms $t = 2$ and, courses per term $k = 2$. The rewards for PLAN 1 is 13 and for PLAN 2 is 15 as the plan's reward is the sum of rewards for its constituent

courses. PLAN 2 is built by a greedy method that chooses the best set of k courses for the subsequent plans. Although it has a higher total reward, PLAN 2 does not take into account the ordering constraints on courses. The course “COMP 206: Introduction to Software Systems”, which is a prerequisite for the course “COMP 303: Software Development”, but it is recommended after it. PLAN 1, on the other hand, has a lower reward but it is more consistent with the ordering constraints between courses. Assuming that COMP 250, COMP 251, and MATH 323 are all taken by the student in previous terms, PLAN 1 does not violate any course prerequisites. Even if one of these prerequisites is violated, we can see that PLAN 1 is better than PLAN 2 since it violates fewer constraints. In addition, PLAN 1 recommends the course “COMP 230: Logic and Computability”, and it might be the case that it is highly recommended to be taken before the course “COMP 330: Theoretical Aspects of Computer Science”. As a result, PLAN 1 would be a better choice for the student. We do not show in this example a quantitative method for preferring PLAN 1 to PLAN 2, but we will discuss that in detail in Chapter 5.

CHAPTER 4

Course Reward Prediction

The approach presented in this thesis for recommending course plans relies on the estimated rewards for potential courses. Course rewards are key in devising personalized course plans that target the individual’s special interests. In this chapter we describe the approach we use to estimate course rewards for a given student.

Rewards for potential courses are student-dependent. More accurately, they depend on the specific course history of the student, along with course ratings given by the student or the received grade. For a target student with transcript T , the reward for a course c that the student have taken previously, i.e. $c \in T$, is based on rating or grade information explicitly given by the student. We will refer to these rewards as *known rewards*. In this thesis, we will use either the rating r_c the student gave to course c , or the grade g_c that the student scored in c as the known reward to a course. In the former case, the first objective of the course plan recommendation becomes finding courses that the student will like the best, while in the latter it becomes finding courses that the student will perform best at. In both cases, the reward prediction process and further the course plan recommendation procedure will be identical except that they will have different value ranges.

Based on the known rewards, the goal is to estimate the *unknown reward* $R_T(c)$ for any course $c \in \mathcal{C} \setminus T$. We adopt two approaches for estimating these rewards. The first is a user-based collaborative filtering approach, which is based on the idea that

students who have similar course histories and preferences (transcripts) will have similar responses on future courses. This approach is based on matching the target student transcript with other student transcripts in order to find a neighborhood set of transcripts which are the most “similar” to it. We then estimate the unknown reward for a course c based on a weighted average of the known rewards of the neighborhood members for the same course c . We will discuss this approach in detail in Section 4.1.

The second is a content-based filtering approach, which relies on the idea that courses with similar content will have similar responses by the same student. We analyze a textual course description provided for each course in our database and infer their similarity based on similarity of their descriptions. The reward for a future course is estimated as a weighted average of known rewards for previous courses taken by the target student. We will discuss this approach in detail in Section 4.2.

These two approaches give two separate predictions of rewards for potential courses. We propose taking a linear combination of both predictions to provide the final prediction, and will show in Section 7.2.3 that combining their results performs better than taking each one individually.

4.1 Rewards Based on Transcript Similarity

The collaborative filtering approach described in this section is accomplished by a neighborhood-based algorithm that computes the unknown reward $R_T(c)$ for the target transcript T , based on the known rewards for the same course c in similar transcripts. Therefore, it is a user-based neighborhood approach as explained in Section 2.1. The similarity between the two transcripts T, T' is measured with the

Pearson correlation coefficient using the set of courses they have in common, $S_{TT'}$:

$$sim(T, T') = \frac{\sum_{c \in S_{TT'}} (R_T(c) - \overline{R_T})(R_{T'}(c) - \overline{R_{T'}})}{\sqrt{\sum_{c \in S_{TT'}} (R_T(c) - \overline{R_T})^2 \sum_{c \in S_{TT'}} (R_{T'}(c) - \overline{R_{T'}})^2}} \quad (4.1)$$

Where $\overline{R_T}$ is the mean of known rewards for courses that appear in T . Based on this transcript similarity measure, we can define the set of most similar transcripts, \hat{S} , but only those who have a known reward for c . We can compute the value of $R_T(c)$ now based on the *known rewards* for c in transcripts of \hat{S} as follows:

$$R_T(c) = \overline{R_T} + k \sum_{T' \in \hat{S}} sim(T, T')(R_{T'}(c)) \quad (4.2)$$

Where $k = 1 / \sum_{T' \in \hat{S}} |sim(T, T')|$ is a normalizing factor. Hence, we have estimated the unknown reward for a course to a target transcript T as the weighted average of the known rewards for the same course but for other similar transcripts.

Neighborhood based collaborative filtering often performs poorly when there is a very small number of ratings, as it uses only ratings of in common items in making predictions and does not rely on prior assumptions [1]. This phenomenon occurs in our system when we have students taking diverse set of courses and have very few courses in common. Generally speaking, students of similar programs take similar courses at the first one or two terms, but tend to specialize and take diverse courses as they progress in their program and take advanced courses. This, in fact, suggests that collaborative filtering predictions of rewards for these advanced courses will not be as accurate as other courses as there will be less data samples to base the prediction on. Nevertheless, students take advanced courses that are related to interests. Thus, we can infer rewards for advanced courses based on known rewards of

similar courses taken by the student. In the following section, We introduce another reward prediction method based on this idea.

4.2 Rewards Based on Course Description Similarity

We describe in this section a content-based filtering method that exploits similarity between course contents in making new recommendations. We take advantage of the textual course description associated with each course in our database in applying content-based filtering. The descriptive textual data is written by course coordinators, so it explains, to a large extent, the content of each course sufficiently. We use this description to match between courses, as we expect that courses with related content have similar descriptions. Similarity between descriptions is based on in common words. For example, the description for Computational Biology (computer science course) has many words in common with the description of Molecular Biology (biology course), although the two are from two different programs. They have words like “molecule”, “gene”, “sequence”, and “structure” in common, although the words might have different inflected forms, such as having “molecular” in one description and “molecule” in the other. Therefore, we follow a standard natural language processing procedure of extracting keyword-based features from course descriptions. A vocabulary of terms is built from descriptions of all courses after stemming their words and thus representing each course description, d_i , as a vector in a vector space model. Then we use the term frequency-inverse document frequency (TF-IDF) weighting scheme to measure the weight of each keyword, kw_j , in the description. Consequently, each course description, d_i , is represented as a

vector $d_i = \{w_{i1}, w_{i2}, \dots\}$, where each w_{ij} is the weight of keyword kw_j in description d_i . The similarity between two course descriptions is computed using the cosine similarity measure as follows:

$$sim(d_i, d_j) = \frac{\sum_k w_{ik} w_{jk}}{\sqrt{\sum_k w_{ik}^2} \sqrt{\sum_k w_{jk}^2}} \quad (4.3)$$

The similarity between any pair of courses is computed only once and stored for later use in all future recommendations. The reward prediction of a potential course c in this method is computed using a nearest neighbor approach, where the predicted rating is computed as a weighted average of known rewards for courses in the target transcript T . The precomputed similarity value between a taken course, $c' \in T$, and the potential course c is used to weight the known reward $R_T(c')$, so courses with similar content will have a greater influence on the predicted rating for a new course. This method, on the other hand, has multiple limitations. Similarity based on textual course descriptions can suffer from the *synonymy* and *polysemy* phenomena that are often in Information Retrieval systems. This in fact can lead to inaccurate similarity values. In addition, this method relies on having similar courses in the student's course history. That is not always the case especially when there are few courses in the course history. We will see later that by combining the results of this method with the previous collaborative filtering method we obtain good prediction results.

CHAPTER 5

The MDP Course Planner

In this chapter, we present our proposed approach for the primary problem of this thesis: finding the “optimal” course plan. We assume that we are given the set of transcripts of an ensemble of students, \mathcal{T} , and we have computed the reward $R_T(c)$ for any potential course c for the target transcript T .

We model the course plan recommendation as a *finite horizon* decision problem for t decision epochs. Thus, we define the decision maker, or *course planner*, as an MDP whose goal is to determine an optimal sequence of actions that correspond to an optimal course plan. We begin with defining the components of the MDP model.

5.1 MDP Model Components

5.1.1 State space S

A state $s \in S$ in the MDP planner is a k -tuple of courses that a student takes in a single term. The choice of state space model fundamentally affects both the efficiency of the system and performance of the recommendations. For example, the most complex model of the state could be representing it as the *sequence* of all previous courses taken by the student. Although this model takes into account the order in which courses are taken in making future decisions, it leads to an intractable number of possible states that cannot be handled efficiently. In addition, it requires a larger number of training samples in order to be able to generalize to new samples,

which is often very hard to achieve. This model can be simplified by ignoring the *order* in which courses are taken, and considering the state as a set of previous courses.

We model the state of the planner as a k -tuple subset from courses of the current term only, as opposed to the more complex modeling of the state as a set of *all* courses taken by the student since the first term. This model reduces the size of the solution space from $O(2^N)$ to $O(\binom{N}{k})$ (N could be in hundreds). Although this approximation implies that previous history takes no part in deciding the next action to take, it does not harm significantly the performance of the planner as courses taken in a term usually depend on courses of the previous term. Besides, courses can implicitly indicate the presence of their prerequisites in previous terms, so they do need to be included in the current state.

We further reduce the effective size of the state space by ignoring unobserved states in the training set. Hence, we build the MDP planner states only from the courses that appear in transcripts of \mathcal{T} . Ignoring unobserved states reduces the novelty of recommended plans; however, it helps in making the problem computationally tractable and eliminates unlikely combinations of courses.

Finally, we reduce the complexity of the state space by *pruning* infrequent states. We eliminate states based on their occurrence in transcripts of \mathcal{T} using two simple schemes: *state-based pruning*, in which we prune states that have their k -tuples observed less than a certain threshold α , and *course-based pruning*, in which we prune states that have at least one of their constituent courses occurring less than a certain threshold β . In addition to reducing the number of possible states, pruning improves the prediction accuracy of the system because it disregards course

co-occurrence patterns that are not supported by enough evidence, as we will see in Section 7.1.1.

5.1.2 Action set A_s

The action set of state s is the set of all k -tuples of courses that can be recommended in s . We choose to consider actions as k -tuples, because the chosen set of “optimal” actions should correspond to the final goal of having a course plan with k courses per term. Likewise, we maintain only action sets observed in transcripts of \mathcal{T} to reduce the combinatorial possibilities of actions. Taking an action $a \in A_s$ in the state s will deterministically produce the new state $s' \in S$ of the planner in the following term, i.e. $Pr\{s_{t+1} = s' | s_t = s, a_t = a\} = 1$. State s' , therefore, is equivalent to action a .

5.1.3 Transition function $Tr(s, a)$

We define the one-step dynamics of our MDP planner as the probability of taking an action a in the state s :

$$Tr(s, a) = Pr\{a_t = a | s_t = s\} \quad (5.1)$$

This definition distinguishes our MDP planner model from the standard MDP stochastic model where the one-step dynamics is defined as the probability of the next state given the current state and chosen action. We adopt this definition of transition function because it models that the planner’s decision of taking an action a in state s based on how often this decision was made by real students. Since this probability encodes course ordering patterns, we use it to enforce taking decisions that respect ordering constraints on courses.

5.1.4 Reward function $R_T(s)$

This function assigns a scalar value to a state $s \in S$ to represent the immediate utility of being at state s in the current term to the target student with transcript T and hence we have it subscripted with T . We define $R_T(s)$ as the sum of rewards for the individual courses that compose state s to the target transcript T :

$$R_T(s) = \sum_{c \in s} R_T(c) \quad (5.2)$$

This raises an important consideration for the MDP model on which components are specific to a target user and which are general. Since rewards for courses are predicted based on the target student's interests and performance, the reward function for states is personalized (this applies to the value function introduced in the next paragraph as well). The state space, action set, and transition function, on the other hand, are not personalized so they are the same for all users.

5.1.5 Value function $V_T^\pi(s)$

The value of a state represents the expected return obtained when following a defined policy for a target student with transcript T . In other terms, it measures the long-term effects of following some defined policy starting from the given state. Since in our model the next state $s' \in S$ is deterministically decided given the chosen action, we define our variation of the *Bellman equation* [45] for the value of a state s under a given deterministic policy π as:

$$V_T^\pi(s) = R_T(s) + Tr(s, \pi(s))V_T^\pi(s') \quad (5.3)$$

5.2 Building the Model

Let us consider how the components of the MDP planner are built from the set of transcripts \mathcal{T} . The k -tuples of courses that represent states are created from courses that appear in the same term in each transcript of \mathcal{T} , and actions are instantiated by considering all k -tuples following these states in the next term. In addition, transition probabilities are estimated by counting co-occurrences of k -tuples in subsequent terms. We note here that this model is built once and used for multiple recommendations. It can be easily updated off-line after having significant quantity of new transcripts to add to the set \mathcal{T} . The reward and value functions, however, are built for each new recommendation because they depend on the target student's preferences.

5.3 Course Planning as Solving the MDP

Solving our MDP model corresponds to finding the policy that achieves the maximum *value* over the t decision steps for the target student. Since our model is deterministic, finding the optimal policy from a starting state s gives the best sequence of sets of courses that the target student should take on the following t terms. The optimal policy π^* has the maximum value function, V_T^* , over all possible policies. We write the *Bellman optimality equation* [45] for V_T^* as:

$$V_T^*(s) = R_T(s) + \max_{a \in A_s} \{Tr(s, a)V_T^*(s')\} \quad (5.4)$$

The system of equations consisting of equation (5.4) for each state s has a unique solution. We use value iteration [45] to find V_T^* .

We can now determine the optimal policy π^* by greedily choosing actions that give the maximum value. We formally write this as:

$$\pi^*(s) = \arg \max_{a \in A_s} \{Tr(s, a)[R_T(s) + V_T^*(s')]\} \quad (5.5)$$

Each application of equation (5.5) will give a set of k courses that the student should take in the corresponding term, and repeating this for t terms gives the required course plan.

The maximized expression in equation (5.5) gives a quantitative value for evaluating the choice of the next action a . This value depends on two factors: first, the transition value, $Tr(s, a)$, which represents how frequent this action is taken *after* the current state s , or in other words, the co-occurrence of the set of courses in action a after the courses of state s in the dataset. The second factor is the value of the next state s' , $V_T^*(s')$, which represents the long term rewards that can be gained after taking the action a . Therefore, both course co-occurrence patterns and user estimated preferences play a role in deciding the next step of the plan.

The final issue is the selection of the first state which the MDP planner starts searching for plans. Given the last term courses in target transcript T , we either try all states that are included in this set of courses, or when no included state can be found, we try all states that have non-zero intersection with this set and choose the one with the maximum total reward. When no state can be found matching the user's last term courses, the planner is not capable of recommending a course plan for this specific user.

CHAPTER 6

Implementation Details

One of the driving objectives of this thesis is to apply our approach for recommendation of items with inter-dependencies in a recommendation system that helps students in selecting academic course plans. Therefore, we have developed and deployed a web-based system course plan recommendation system that employs an implementation of our approach. The current system is tailored towards Computer Science and Software Engineering students at McGill University. Nevertheless, the system can be easily extended by including data from other departments or from other universities.

We present in this chapter our implementation of the course plan recommendation system. In addition, we discuss the data collection step of this project, which was a crucial step in building the transcripts database that the recommendation systems depends on in generating recommendations.

6.1 Course Plan Recommendation System

The goal of this system is to provide a web-based application that students can access through the Internet and obtain recommendations for courses to take in subsequent terms. We deployed this web-based system on the Z Object Publishing

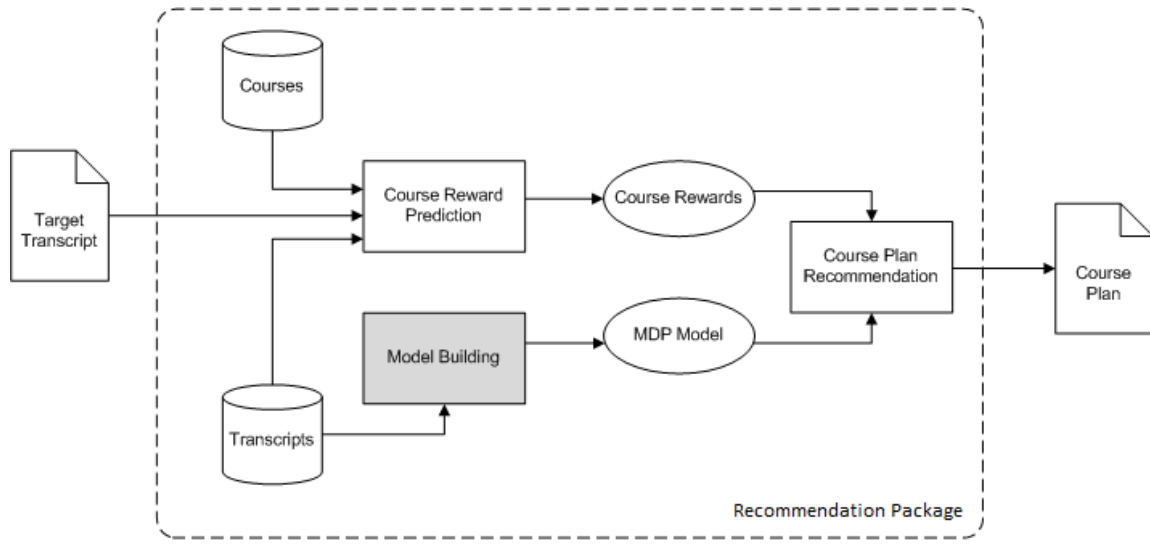


Figure 6–1: Overview of the course plan recommendation package, where *model building* is shaded to point that it is performed offline

Environment (Zope),¹ which is a free, well documented, and easy to use web application server with an embedded relational object database. The system interacts with the user through dynamic web pages to collect information about courses taken previously by the student, and then displays course plans generated by a back end recommendation package. Data used for recommendation and presentation is stored in a MySQL database. We describe in the following the components of the course plan recommendation system in more detail.

6.1.1 Recommendation Package

The recommendation package forms the computational core of the system. It uses data consisting of transcripts and courses in order to recommended a course

¹ <http://docs.zope.org/zope2/zope2book/>

plan for the target user. Figure 6–1 illustrates the main functionalities of the recommendation package. The package has three main procedures: model building, course reward prediction, and course plan recommendation. Model building differs from the other two procedures in that it is performed *offline* to generate the MDP model (states and transition probabilities) based on the set of transcripts \mathcal{T} stored in the database. The MDP model is kept in the memory of the server for later use in all recommendations in order to increase the efficiency of the recommendation process. The MDP model can be updated whenever there is additional “experience” or student transcripts in the database, and reloaded to the server’s memory. The *online* recommendation process starts with estimating the personalized rewards for potential courses based on the target transcript using the reward prediction method explained in Chapter 4. Reward prediction relies on potential course information and their textual descriptions in order to compute content-based rewards, and transcript data of other students in order to compute collaborative-filtering rewards, where both are stored in the database. Finally, the course plan recommendation process first computes values of states of the MDP model, and then it finds the best plan based on the MDP model and estimated rewards for courses. It generates as a result a course plan for the target student.

The computational cost of the online course recommendation depends mainly on the efficiency of course reward prediction and MDP state values calculation (described in Section 5.3). The efficiency of our user-based collaborative filtering implementation depends on the number of available students, since we follow a neighborhood-based approach. On the other hand, finding similarity between *each*

pair of courses, which is required for the content-based filtering reward prediction, is $O(N^2)$, where N is the number courses, but we perform that *offline* and store similarity values in a look-up table. The performance bottleneck of our approach is, in fact, in calculating MDP state values. In our current implementation, the value iteration algorithm converges after 7 or 8 iterations. The average of iterations before convergence for a test of 200 runs of the algorithm is 7.6. This is mainly due to lack of loops between states, which is often called *directionality* in state space [44]. The main drawback of our approach is the potential of having very large number of state space, which we try to overcome by ignoring unobserved states and pruning the ones are observed infrequently.

6.1.2 Database

Providing useful and effective recommendations depends heavily on the richness of the dataset. Due to the lack of any publicly available course and transcript dataset, we had to build our own dataset. We collected our transcript data from Computer Science and Software Engineering students at McGill through a web survey which they were asked to complete voluntarily (we will discuss the survey in Section 6.2)². We collected total number of 65 transcripts from students. Although this size of the dataset is relatively small, it was enough to provide good validation results and prove the effectiveness of our approach.

² Subsequent to ethics approval




COMP 302		3 credits
Programming Languages and Paradigms		
Computer Science (Sci): Programming language design issues and programming paradigms. Binding and scoping, parameter passing, lambda abstraction, data abstraction, type checking. Functional and logic programming.		
Offered by: Computer Science		
<hr/>		
3 hours		
Prerequisite: <i>COMP 250 or COMP 203</i>		
<hr/>		
Terms		Instructors
 Fall 2011		 Brigitte Pientka
 Winter 2012		

Figure 6–2: Overview of the course plan recommendation package

Course data was extracted from online sources at the McGill University official website ³. This web page provides a synopsis for any course at McGill. Figure 6–2 shows an example of the synopsis for COMP 302 course. We initially extracted course data from the synopsis automatically for all Computer Science and Software Engineering courses and all related courses from other departments such as Mathematics, Physics, Chemistry, and other departments. In addition, courses that appeared in a new student’s transcript but were not present in the database were also extracted from the course web page and added to the database. As a result, we currently have 370 courses in the database.

Course data stored in the database includes: course code, title, textual description, and hard prerequisites. The textual description is used to compute similarity between courses as described in Section 4.2. This, however, is performed only once

³ e.g. <http://www.mcgill.ca/global/php/coursepopup.php?course=comp+302>

and the inter-course similarity is stored as a table for later use in course reward prediction for target transcripts. Hard prerequisites, on the other hand, require manual preprocessing before being stored in the database in a well-formed logical expressions. This is due to the fact that prerequisites written in in the available course synopses do not conform to a fixed format, as they are usually written by course coordinators in unstructured text and intended to be understood by humans rather than computer programs. This introduces ambiguity in parsing them automatically, and hence the need for manual preprocessing.

6.1.3 Presentation

The web interface of the system consists of a dynamic web page that users use to input their courses with their grades and ratings. We have used in designing the interface jQuery,⁴ which is a JavaScript library that facilitates client-side interactivity on web sites. Figure (6.1.3) shows an example of our system's web interface. The user can input his courses, as in Figure (6-3a), by typing in the course code or the course title, and this is searched through our course database. This is achieved through AJAX (i.e. without reloading the whole page) to provide a more efficient experience to users. If the course is found in the database it is added to the current term courses. Otherwise, it is looked up from the McGill course pop-up web application, and if it exists (a valid course at McGill), it is added to the current courses and also stored in the course database permanently. The user can add multiple courses per term and add multiple terms according to his/her current course history. Finally,

⁴ <http://jquery.com/>

when the user submits the transcript, it is sent to the recommendation application, which processes the transcript and returns a recommended course plan according to the number of courses per term, k , and number of terms, t , specified by the user. Figure (6–3b) shows the recommended plan for the input given in Figure (6–3a).

6.2 Transcripts Survey

Collecting transcript data from previous students is a critical step in both building a system that future students can use and also for evaluating the overall approach. In order to collect this data, we built a web survey for students to access online and provide their own experience. We targeted Computer Science and Software Engineering students with this survey who are either currently enrolled in the program or have graduated. Focusing on these two closely related majors leads to a small system, yet it retains important features of the problem that make it interesting and complex.

Having a large set of transcripts in the database helps in discovering more course co-occurrence patterns and covering more instances of student opinions. Therefore, the main design goals of this survey were to make it easy to use and can be finished quickly so that it does not take much the volunteering student’s time. Our first version of the survey required students filling all courses taken in each term and providing their grades and ratings. We found that we got low completion rates when it was too complex and took long time to finish. In order to minimize the time overhead, we improved the survey by adding the ability of parsing the electronic unofficial copy of transcripts, which is available to all McGill students from Minerva

Term 1

	Code	Title	Grade	Rating
✖	COMP 250	Introduction To Computer Science	A ▼	
✖	MATH 223	Linear Algebra	A ▼	
✖	MATH 222	Calculus 3	A- ▼	
✖	MATH 240	Discrete Structures 1	B+ ▼	

[Reset Onwards](#)

Term 2

	Code	Title	Grade	Rating
✖	COMP 206	Introduction To Software Systems	A- ▼	
✖	MATH 340	Discrete Structures 2	A ▼	
✖	COMP 273	Introduction To Computer Systems	A ▼	
✖	MATH 323	Probability	B+ ▼	

Type Course Code:

[Add Course](#)

[Add Another Term](#) [Submit Transcript](#)

(a) Example of input to the system

Term 1







Code	Title
COMP251	Data Structures And Algorithms
COMP302	Programming Languages And Paradigms
COMP310	Computer Systems And Organization

Term 2

Code	Title
COMP303	Software Development
COMP310	Computer Systems And Organization
COMP330	Theoretical Aspects: Computer Science

(b) Example of the system's output

Figure 6-3: System's presentation

Fall 2005					Winter 2006				
	Code	Title	Grade	Rating		Code	Title	Grade	Rating
✗	CHEM 212	Intro Organic Chemistry 1	A- ▼	    	✗	BIOL 112	Cell and Molecular Biology	A ▼	    
✗	CHEM 257D1	Intro Analytical Chemistry	A ▼	    	✗	COMP 250	Intro to Computer Science	A ▼	    
✗	COMP 202	Intro to Computing 1	A ▼	    	✗	LING 201	Introduction to Linguistics	A ▼	    
✗	MATH 133	Vectors, Matrices and Geometry	A ▼	    	✗	MATH 151	Calculus B	A ▼	    
✗	MATH 150	Calculus A	A ▼	    	✗	PSYC 211	Intro Behavioral Neuroscience	A ▼	    





Fall 2006					Winter 2007				
	Code	Title	Grade	Rating		Code	Title	Grade	Rating
✗	COMP 206	Intro to Software Systems	A- ▼	    	✗	COMP 252	Algorithms and Data Structures	B+ ▼	    
✗	COMP 302	Programming Lang & Paradigms	B ▼	    	✗	COMP 273	Intro to Computer Systems	B+ ▼	    
✗	MATH 235	Algebra 1	A- ▼	    	✗	MATH 251	Honours Algebra 2	B+ ▼	    
✗	MATH 242	Analysis 1	B ▼	    	✗	MATH 255	Honours Analysis 2	A ▼	    
✗	MATH 248	Honours Advanced Calculus	C ▼	    	✗	MATH 377	Honours Number Theory	A- ▼	    

Figure 6–4: Screenshot of a parsed transcript on the survey

information system.⁵ Students submit the HTML file of their unofficial transcript, and the parsed courses and grades are displayed to them as in Figure 6–4. The student has only to give a rating, which expresses their *personal opinion* on the course. The provide ratings for each course by selecting one of the five rating choices that appear for each course, and finally submit the transcript to complete the survey. The submitted transcript with course ratings is then stored in the database. This survey takes very little time to finish and has a better reliability for the provided transcripts, which has resulted in an increased user participation over earlier more tedious versions.

⁵ www.mcgill.ca/minerva

CHAPTER 7

Experimental Evaluation

In this chapter we provide an empirical validation of our approach. We divide our discussion into three sections. Section 7.1 provides a detailed description of the dataset we use in our experiments, and was collected from real students through a web survey. We present in Section 7.2 an evaluation of the hybrid reward prediction method described in Chapter 4 against baseline methods and both partial methods on our dataset. Evaluating the reward prediction method is key in assuring that our system can provide personalized course plans. We finally evaluate the MDP course plan recommender in Section 7.3, and show that our MDP approach generally outperforms methods that rely solely on course co-occurrence patterns or on course rewards in devising course plans.

7.1 McGill Transcripts Dataset

Our experiments are conducted on a set of transcripts that we collected from students studying Computer Science and Software Engineering at McGill University through the web survey discussed in Section 6.2.

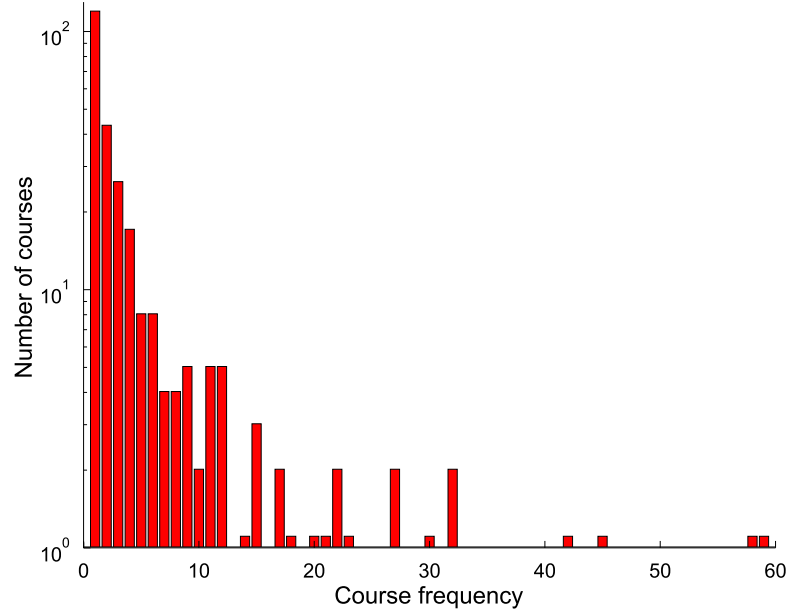
The dataset has 65 transcripts with various degrees of completion, but the average length of transcripts spans 4 terms (students typically take 3-5 courses per term). The dataset has 370 different courses, which contain all Computer Science and Software Engineering courses and their prerequisites from within the department and from other related departments such as Math and Physics. In addition to core

courses, the dataset contains a rich set of elective courses from various departments (e.g. Biology, Psychology, Economics ...etc), which were collected based on courses encountered in transcripts of student who participated in the survey. In the following section we discuss the sparsity phenomenon in our dataset, which is commonplace in recommender systems. In our dataset, however, sparsity is more severe as a result of the small number of transcripts that we have, and as a result of the set-based representation of our MDP states.

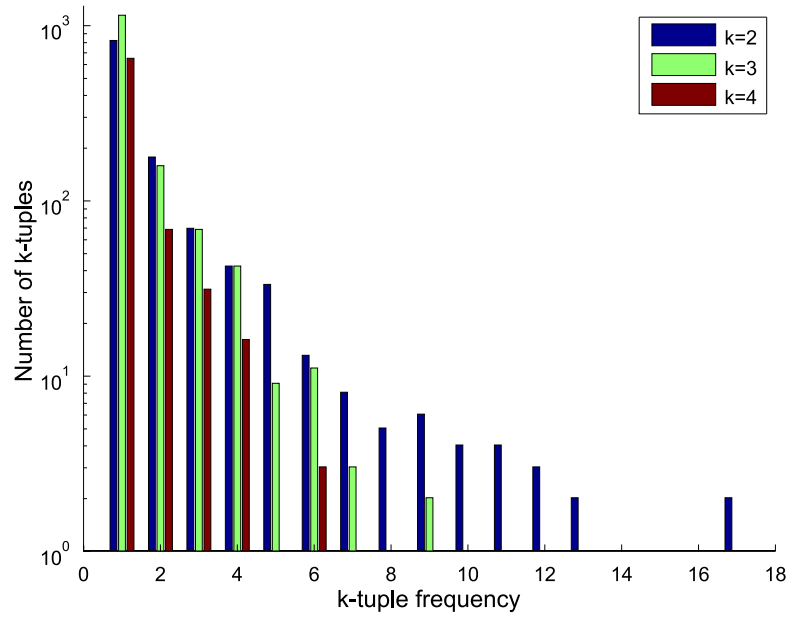
7.1.1 Data Sparsity and State Pruning

Although our dataset is collected from Computer Science and Software Engineering majored student only, the existing transcripts have few courses in common. This is due to the relatively small number of transcripts, which does not cover the vast diversity of possible course selections. We had many students taking the survey enrolled in joint major programs, or only have a minor in Computer Science, which lead to a diverse set of courses. In addition, the number of courses in Computer Science and Software Engineering is quite large, and there are multiple sub-interests withing these two fields. As a result, our dataset is highly sparse and this is manifested by having relatively few courses taken -and therefore rated- by a large number of students.

We examine the data sparsity issue in our dataset in Figure (7-1). We define the term *course frequency* for a course c as the number of transcripts in the dataset that c appears in, or in other words, the number of students in our dataset took this course. For example, if we have a dataset of two transcripts: T_1 composed of courses $\{c_1, c_2, c_3, c_4\}$, and T_2 composed of $\{c_1, c_3, c_5\}$, then c_1 and c_3 have frequency



(a) Frequency of course occurrences in dataset.



(b) Frequency of k -tuple occurrences in dataset.

Figure 7–1: Dataset sparsity illustrated by the number of courses or states that occur with certain frequencies in the dataset.

of 2, and all other courses have frequency of 1. We are interested in the number of courses that occur in our dataset with a certain frequency, because it gives us an idea of how many courses the transcripts in our dataset have in common. Figure (7-1a) illustrates the number of courses that occur with a certain course frequency in our dataset. We can see that the number of courses taken by very few students is large compared to courses taken by many students (119 course taken by 1 student as opposed to only 1 course taken by 59 students). This, in fact, makes the course plan recommendation problem harder because, on one hand, it becomes very difficult to match student preferences, and on the other hand, it introduces many course co-occurrence patterns that are not supported by enough evidence and can be therefore considered as outliers.

Data sparsity is more severe when we consider frequencies of tuples of courses, which will be eventually the states of our MDP course planner. Similar to the individual course case, we define *k-tuple frequency* as the number of transcripts in the dataset that a *k*-tuple of courses appears in (tuples are composed of courses that occur in the same term). Figure (7-1b) shows the number of *k*-tuples (with $k = 2, 3, 4$) that have certain frequencies in our dataset. We can observe that *k*-tuple frequency values are less than course frequency values, but their counts are larger as a natural result of taking combinations of courses. We are interested in *k*-tuples that have high frequencies, because they are part of course co-occurrence patterns that we are after.

In order to alleviate the effect on data sparsity on our MDP course planner prediction accuracy, we eliminate infrequent states by performing either state-based

pruning or course-based pruning, depending the frequency of k -tuples or individual courses, respectively, in the dataset.

7.2 Course Reward Prediction Evaluation

Understanding preferences of individuals is a very important step in designing personalized course plans that match their interests and needs. Since user preferences in individual courses are modeled with reward values, we must ascertain that our reward prediction method provides good estimates for them. In this section we describe the performance evaluation conducted on the hybrid course reward prediction method explained in Section 4 on our transcripts dataset, and show that it can provide very good validation results.

7.2.1 Performance Measure

The ultimate goal of course reward prediction is to use the predicted rewards in selecting the most interesting set of courses to the user. The specific values of course rewards are therefore not significant as long as the reward prediction method gets the right *ranking* for courses. Consequently, we use in evaluating reward predictions the Kendall- τ statistic, which is a standard measure for estimating ranking correlation, i.e. the degree of agreement in ordering pairs of items between two rankings. In our case, the first ranking is the true ranking of courses that the student gave, which is the reference ranking, and the second is the ranking accomplished with the estimated rewards of our system.

We adopt the implementation of Kendall- τ explained by Shani and Gunawardana [43]. Given the rating value r_{ui} of the user and \hat{r}_{ui} of the system that is given

to an item i for a user u , we define:

$$C^+ = \sum_{ij} \text{sgn}(r_{ui} - r_{uj}) \text{sgn}(\hat{r}_{ui} - \hat{r}_{uj}) \quad (7.1)$$

$$C^- = \sum_{ij} \text{sgn}(r_{ui} - r_{uj}) \text{sgn}(\hat{r}_{uj} - \hat{r}_{ui}) \quad (7.2)$$

$$C^u = \sum_{ij} \text{sgn}^2(r_{ui} - r_{uj}) \quad (7.3)$$

$$C^s = \sum_{ij} \text{sgn}^2(\hat{r}_{ui} - \hat{r}_{uj}) \quad (7.4)$$

The values C^+ and C^- represent the number of pairs that the two rankings agree and disagree on their ordering respectively, and C^u is the number of pairs of items for which the user ranking gives an ordering (are not tied), while C^s is the same but for system ranking. Given these values, Kendall- τ metric is calculated as follows:

$$\tau = \frac{C^+ - C^-}{\sqrt{C^u} \sqrt{C^s}} \quad (7.5)$$

When there is a perfect agreement between the two rankings $\tau = 1$, and in complete disagreement $\tau = -1$, while $\tau = 0$ represents that the two rankings are independent.

7.2.2 Experimental Procedure

Evaluation is based on the leave-one-out cross validation paradigm, where all transcripts in the dataset, except one test transcript, are used as the set \mathcal{T} , and the reported value is the average for all possible test transcripts. In each transcript, we assume courses are ordered according to their terms, but within the same term they

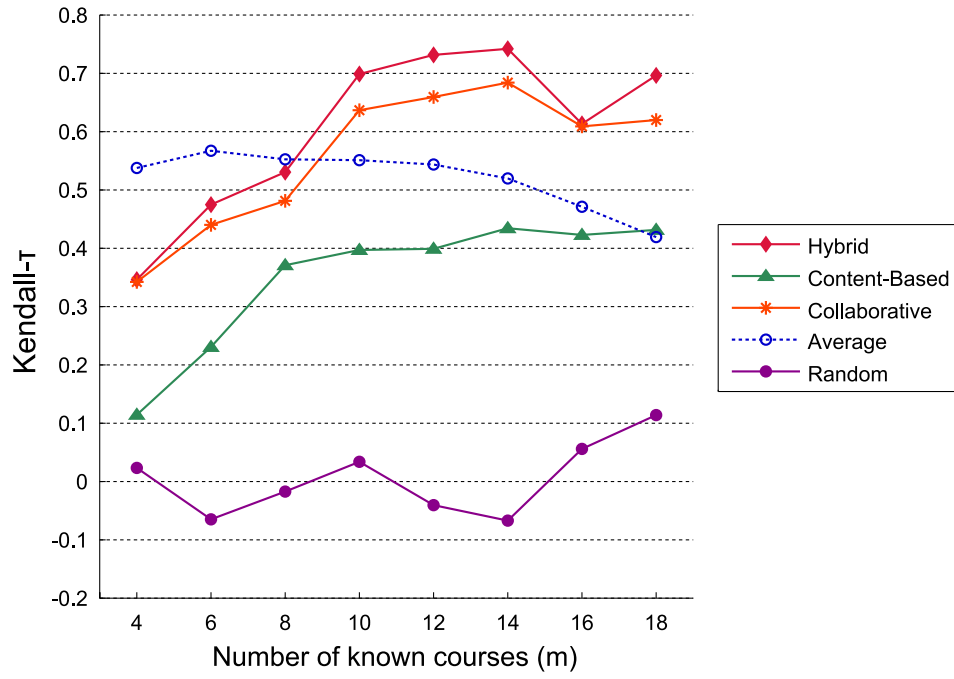
are ordered arbitrarily. We consider the first m courses as “known” courses, so we know their rewards *a priori* (i.e. the known courses compose a target transcript T that we wish to recommend courses to), and we wish to predict the rewards for the rest of the courses. In Figure (7–2) we study the prediction performance against the number of known courses, m , in each test transcript. Since transcripts in our dataset vary in their size, we choose only transcripts that have at least 2 more than the maximum number of known courses we take (i.e. $m = 18$) in order to do the evaluation on the same set of test cases and make results comparable.

We compare the hybrid method with two non-personalized baseline methods: “Random”, which predicts a random reward for a known course, and “Average”, which assigns an average of other students’ known rewards for the course as the estimated reward. We show the results of the pure collaborative and content-based filtering methods as well. We perform the experiment on course ratings in the range $[1, 5]$ as rewards (Figure 7–2b), and on course grades in the range $[1, 9]$ ¹ as rewards (Figure 7–2a). We refer to the collaborative filtering method (Section 4.1) by “Collaborative”, the content-based method (Section 4.2) by “Content-based”, and finally the hybrid of the two by “Hybrid”.

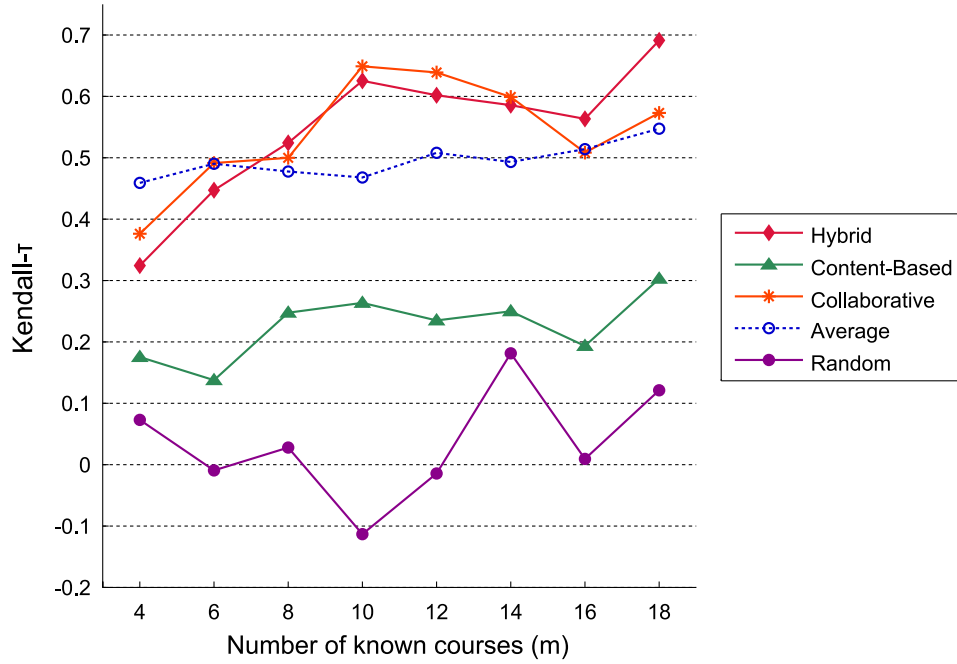
7.2.3 Results and Discussion

We can observe that “Hybrid” generally does the best, and has a comparable performance with “Collaborative” in Figure (7–2b). It has large Kendall- τ values (over 0.7) after it has sufficient data about the user ($m \geq 10$). This means that over

¹ This range corresponds to McGill University grade scale



(a) Rewards based on course ratings



(b) Rewards based on course grades

Figure 7-2: Reward prediction performance

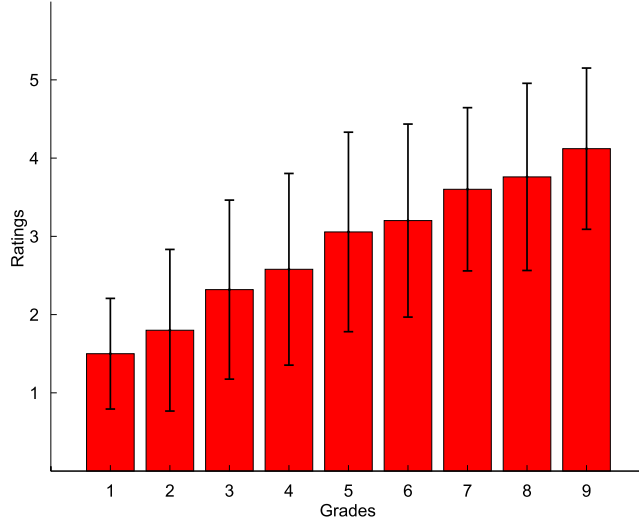


Figure 7–3: Average values of known ratings for each of the nine values of grades in all transcript entries of the dataset.

70% of the predicted ranking of pairs of courses agree with the true ranking. The three personalized models: “Hybrid”, “Collaborative”, and “Content-based” start with low ranking correlation values, and increase as m increases. This is a natural behavior because increasing the knowledge about the student’s previous preferences or performance should lead to better reward prediction performance. We note that the “Content-based” model has relatively low performance, which points out that using subjective opinions of users in our problem is a more reliable indicator of their preferences. Although the “Content-based” model does not perform well, combining it with the “Collaborative” model helps most of the time in increasing the performance. This is mainly due to the fact that our dataset is highly sparse, and a pure neighborhood-based approach needs to be supported with additional information. Using course similarity, especially when having enough information about the user,

can help in making better predictions. We notice that the “Average” model has quite stable values around 0.5 performance, which means that general opinion on courses achieves correct ranking about 50% of the time.

Another interesting observation is that although the personalized methods have slightly better predictions in the case of ratings as rewards, we can notice that all methods have very similar relative performance in both cases of ratings and grades as rewards. This indicates that ratings and grades are both correlated, such that when a student gets a high grade in some course, they tend to give a high rating to it, and vice-versa. We investigate this phenomenon more closely in Figure (7-3), where we take the average of student ratings for each of the nine grade values. We can see that rating values are linearly correlated with grade values, which is not surprising as the grade that students receive usually affects their final opinion on the course. The Pearson correlation coefficient of given ratings and received grades for courses in all transcripts of our dataset is 0.47.

7.3 Evaluation of Course Plan Recommendations

In this section we evaluate the effectiveness of our approach in designing course plans on our dataset of real student transcripts. We discuss in the following sections the used performance measures, the experimental methodology, and finally results of experiments.

7.3.1 Performance Measures

We consider three measures to evaluate the performance of the MDP course plan recommender and compare it with other methods.

Accuracy

This measure evaluates how well the recommender predicts “relevant” courses the student will take in the future. The relevance of a course is decided based on the user’s true opinion on the course. Evaluating recommended course plans on how much they agree with true transcripts is motivated by the fact that students generally try their best in choosing courses they will like, and it would be nice to see how much a recommender system can match their performance.

In contrast to the other definition of accuracy, where the goal is to evaluate predicting *any* course the student will take (*predictability* as in [35]), this measure assesses the ability of the algorithm of predicting the items that the user likes. In our case, a course is considered relevant to the student if its reward is greater than the average of rewards for all courses previously taken by the student. For example, if the student in reality took two courses c_1 and c_2 , and gave them a rating of 1 and 3 respectively, we consider c_2 to be a relevant course because it has a reward above the average 2, while c_1 is not relevant. We prefer this definition of relevance to having a fixed threshold (e.g. ≥ 4 is relevant in $[1, 5]$ scale) because it helps avoiding bias in cases where students give high ratings or low ratings for all courses (and likewise for grades). Therefore, we define accuracy as the percentage of relevant courses that were recommended to the user of all recommended courses.

Coverage

This measure evaluates the user coverage of the system. We define the coverage measure as user-based coverage, i.e. as the percentage of recommendations that the system can provide to users of the ones it is asked for. A similar definition for

coverage has been used in previous works [9, 35]. In our case, when, for example, the system is asked to provide a course plan of $t = 2$ terms for 2 students, and it could only provide one of the with a full plan and the other with only one term plan the coverage will be 75%.

We are interested in coverage because we expect to get higher prediction accuracy as we prune more infrequent states, but this will not make the system useful as it will cover fewer test samples (i.e. pruning will leave only mainstream courses).

Prerequisite concordance

On the contrary to the first two metrics which are used to evaluate how much the recommended plan is truly relevant or interesting to the user, this performance metric is designed to evaluate how much the recommended course plans conform to course prerequisites. We define the prerequisite concordance as the percentage of courses that satisfy prerequisites from the total number of recommended courses.

As we have discussed earlier in Section 3.1, course prerequisites are modeled as a logical expression of AND-OR operations and prerequisite courses as variables. We check that prerequisites are satisfied by giving a value *true* for each prerequisite course that appears in previous terms, and *false* otherwise. If the expression evaluates to *true* then the course prerequisites are satisfied.

7.3.2 Experimental Procedure

We adopt the same leave-one-out cross validation methodology we applied in Section 7.2.2 in this set of experiments. We assume here, however, that we know *a priori* a first part of the test transcript (not included in \mathcal{T}), and hide all courses in the left part of it, as opposed to hiding only their rewards. The “known” part of the

test transcript in this set of experiments is in terms rather than individual courses, because it reflects students with different degrees of program completion that we wish to recommend courses to.

The MDP planner model is evaluated against two methods. The first is *Markov Chain* (MC) model that has the same state space and transition function of the MDP model, and is built in a similar way as described in Section 5.2. The transition function in the MC model, however, defines transition between two states, and the MC model does not have the notions of state rewards or values. Therefore, this model recommends plans based on course co-occurrences only. We would like to compare the quality of its recommended plans with the ones that our MDP approach recommends by incorporating values of courses. The second method we compare against is a *Reward-based* course planner, which relies exclusively on course rewards in designing plans. This method chooses greedily for each term courses that have the highest rewards, but only from the courses that have satisfied prerequisites. Comparing our model to this method shows how exploiting course co-occurrences can affect the quality of plans.

We evaluate the MDP and MC models with state-based pruning (SBP) and course-based pruning (CBP) strategies introduced in Section 5.1.1. We, therefore, have two versions of our experiments for the SBP and CBP, as we are interested in comparing the performance of MDP and MC models in a similar setting, along with the Reward-based model. In order to use these two pruning strategies, however, we must set a value to the thresholds α and β for the SBP and CBP strategies respectively. We set in the following experiments the value $\alpha = 1$ for SBP that

correspond to pruning states occurring only once in the dataset, and the value $\beta = 3$ for CBP that correspond to pruning states occurring less than 3 times in the dataset. We choose these values empirically because they give a good trade-off between accuracy and coverage values as we will show in Section 7.3.4.

We perform two types of experiments: the first studies the average performance of the three methods as we vary the knowledge of the target student (i.e. the size of the “known” part of the target transcript). The second studies the effect of increasing the length of the recommended plan. We consider, in addition, the both cases of ratings and grades as rewards for MDP and Reward-based models. We discuss the results of these experiments in detail in the following section.

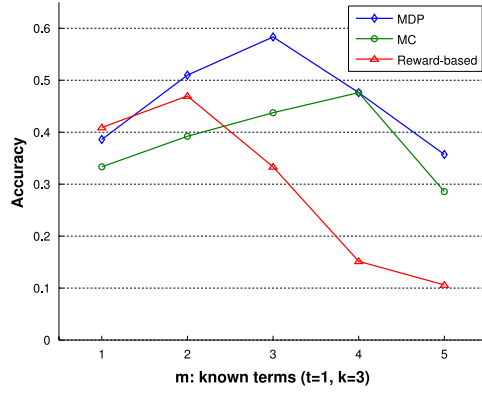
7.3.3 Results and Discussion

Accuracy and coverage vs. size of *a priori* knowledge of user

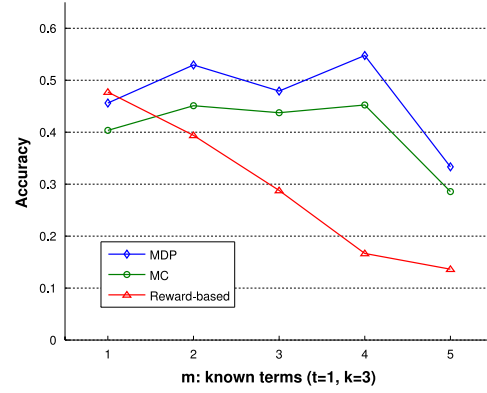
Figures (7-4) and (7-5) show the average accuracy and coverage results of the three methods versus different values of m , i.e. the number of terms known from the target transcript, when recommending a plan of 3 courses per term for a single term (i.e. $t = 1, k = 3$). Figure (7-4) shows the results of the MDP and MC models when state-based pruning is performed with $\alpha = 1$, while Figure (7-5) shows their results when course-based pruning is performed with $\beta = 3$. We can observe that in both experiments that the MDP model generally does better than the MC and Reward-based models. We believe this is due to the fact that the MDP model enjoy the benefits of both exploiting user interests and course co-occurrence patterns in choosing courses. The Reward-based model performs relatively well for $m \leq 2$, but does worse afterwards mainly because estimated rewards fail to provide a better

advantage to advanced courses over introductory courses in last terms - when both are available as their prerequisites are satisfied. For example, it might recommend in the last term an introductory Biology course for a student who expressed interest in Bioinformatics courses before, while it might be appropriate to recommend an advanced Computational Biology course. This is due to ignoring the temporality information in this method, while methods that exploit co-occurrence patterns take this information into account by assigning high probability to advanced courses to follow after introductory courses. All methods perform similarly when rewards are based on ratings or on grades, which is due to the correlation between ratings and grades as we discussed earlier. The accuracy of all methods increases (except the Reward-based model) as a result of having more *a priori* knowledge about the student, but it then deteriorates because the prediction problem becomes harder as the courses at the end of the program become more specialized and harder to predict.

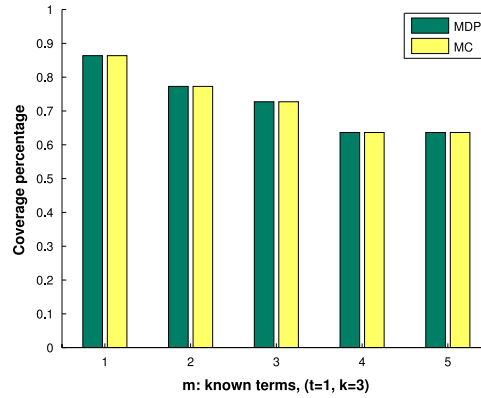
The coverage of the MDP and MC models is shown in Figure (7-4c) and (7-5c). We show one coverage result for both ratings and grades as rewards, since it does not have any effect on the coverage. We observe that the coverage of both methods are equal because we are evaluating a plan of only one step (i.e. $k = 1$) in these experiments (we will see later that they can differ when plans have more than one step). We do not show the coverage of the Reward-based model, since it can always find some courses with satisfied prerequisites to recommend and therefore covers all test samples. In the MDP and MC models, however, the student's last term might match only to states that have no further transitions, so it cannot provide any recommendation, and since the last term either matches to some state or not,



(a) Accuracy with ratings as rewards



(b) Accuracy with grades as rewards



(c) Coverage percentage

Figure 7–4: Average *accuracy* and *coverage* vs. number of known terms when using state-based pruning (SBP) with $\alpha = 1$

both methods have same coverage values. This phenomenon, in fact, happens more frequently in last terms, and we therefore see coverage decreasing as we increase m .

Finally, we observe that although using SBP with the MDP and MC models gives better accuracy results than CBP (with $\beta = 3$), it has lower coverage - even when we use the lowest possible α threshold.

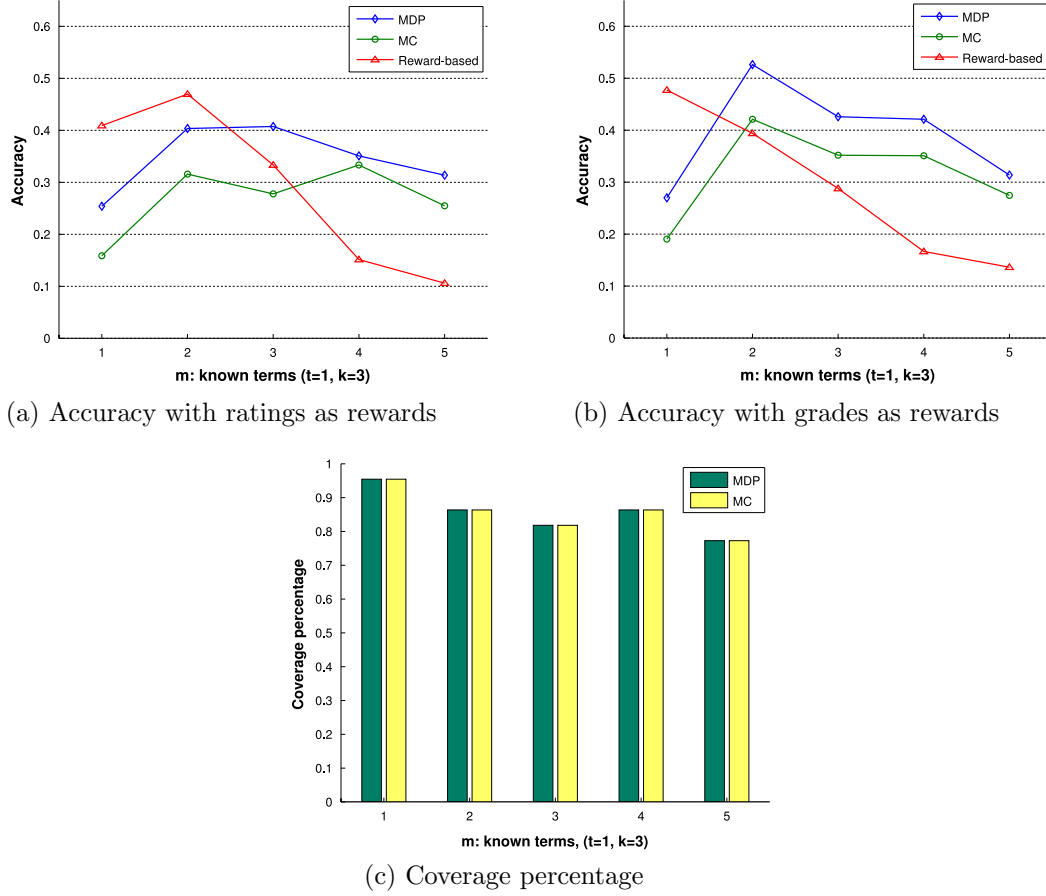
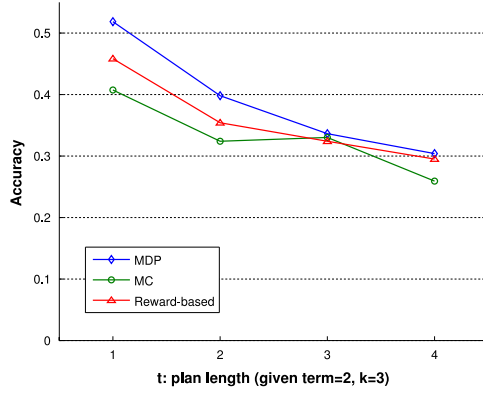


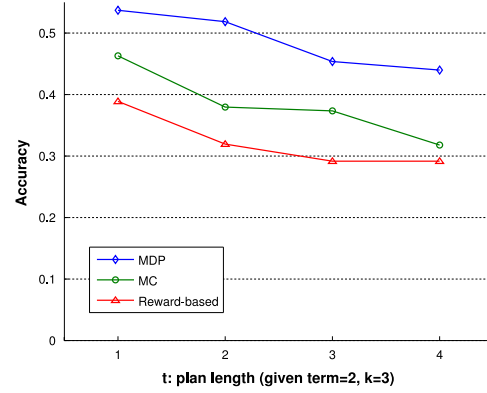
Figure 7-5: Average *accuracy* and *coverage* vs. number of known terms when using course-based pruning (CBP) with $\beta = 3$

Accuracy and coverage vs. length of course plan

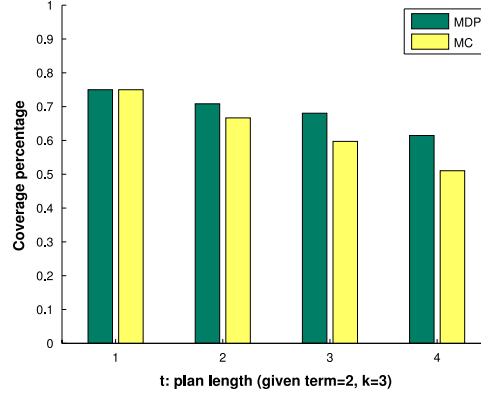
In Figures (7-6) and (7-7) we study the effect of increasing the length of the recommended plan t , which is composed of also 3 courses per term ($k = 3$). We assume here a fixed “known” part of 2 terms. Similar the previous experiment, we observe that the MDP model (with SBP and CBP) generally has better accuracy than other methods. Accuracy, in all methods, gradually decreases as we increase the



(a) Accuracy with ratings as rewards



(b) Accuracy with grades as rewards

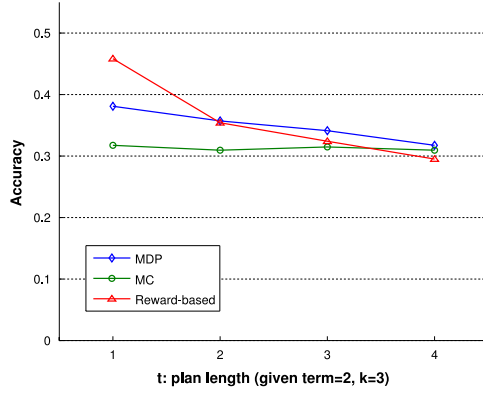


(c) Coverage percentage

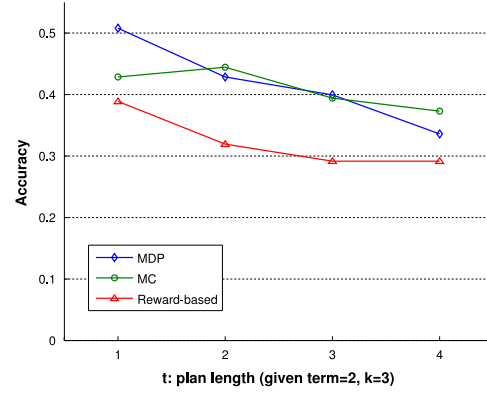
Figure 7-6: Average *accuracy* and *coverage* vs. length of plan in terms when using state-based pruning (SBP) with $\alpha = 1$

size of the recommended plan. This is mainly due to the fact that as we increase the size of the required recommendation, the numerator of the accuracy ratio becomes bigger and the predictability problem becomes harder, which will make the accuracy naturally drop.

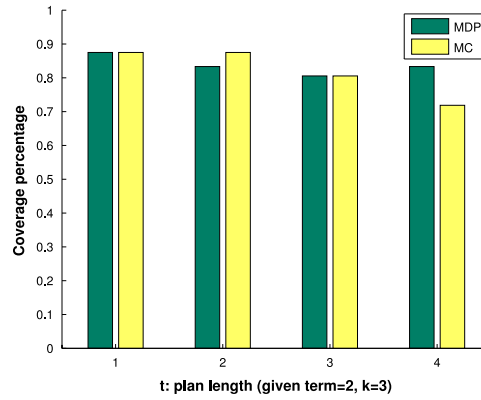
The coverage of the MDP and the MC models differs slightly in these experiments. This is because the two algorithms take different approaches in choosing



(a) Accuracy with ratings as rewards



(b) Accuracy with grades as rewards



(c) Coverage percentage

Figure 7-7: Average *accuracy* and *coverage* vs. length of plan in terms when using course-based pruning (CBP) with $\beta = 3$

states of the model. We observe that the MDP model has slightly higher coverage than the MC model. This is mainly because action selection in the MDP model is based on state values, which generally increase when states are followed by future states for multiple time steps, especially when there is no discount factor. This can be observed in Figure (7-6c), but it is less obvious in Figure (7-7c).

Table 7–1: Prerequisite Concordance Results

Real Transcripts	MDP	MC
72.5%	68.6%	63.3%

Prerequisite concordance results

In order to evaluate our MDP approach in producing plans that conform to course dependencies (exemplified in course prerequisites), we evaluate the ratio of the recommended plan courses that have satisfied prerequisites, and compare it to the same ratio but for real student transcripts. Table (7–1) shows the prerequisite concordance values for real transcripts, MDP, and MC approaches. Real transcripts in our dataset have the value 72.5% compared to 68.6% and 63.3% of plans recommended with the MDP and MC approaches respectively. The recommended plans have $t = 3$, $k = 3$, and are based on 2 “known” terms. This means that 68.6% of the recommended courses by our MDP model satisfy prerequisites, and this not very far from real student transcripts. In practice, transcripts of real students do not always satisfy all prerequisites for several reasons as discussed in Chapter 1. The MDP and MC models, however, match the performance of real transcripts, because they rely on course co-occurrence patterns learned from the dataset.

7.3.4 Experimental Evaluation of α and β Thresholds

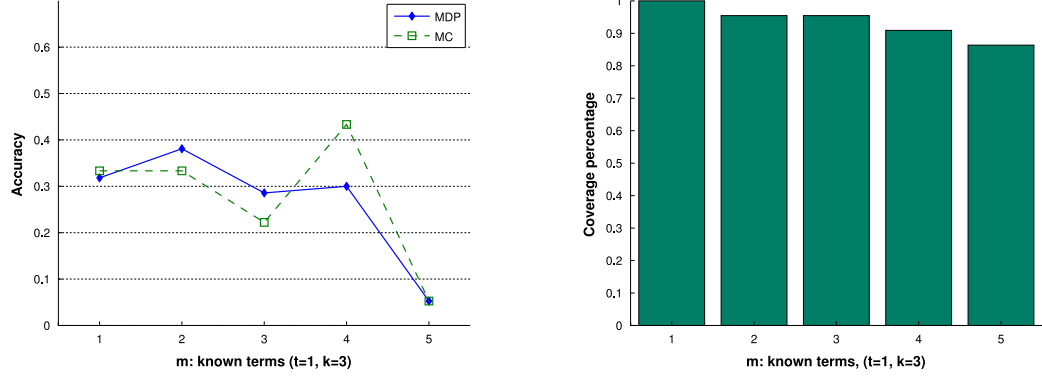
In this section we present an experimental evaluation of different values for pruning thresholds α in state-based pruning (SBP), and β in course-based pruning (CBP), and their effect on the accuracy and coverage of our MDP model. We perform a similar experiment to the one introduced in Section (7.3.3), where we report average values of accuracy and coverage of recommending a plan of a single term with 3

courses per term (i.e. $t = 1, k = 3$), versus a different values of “known” terms of the target transcript m .

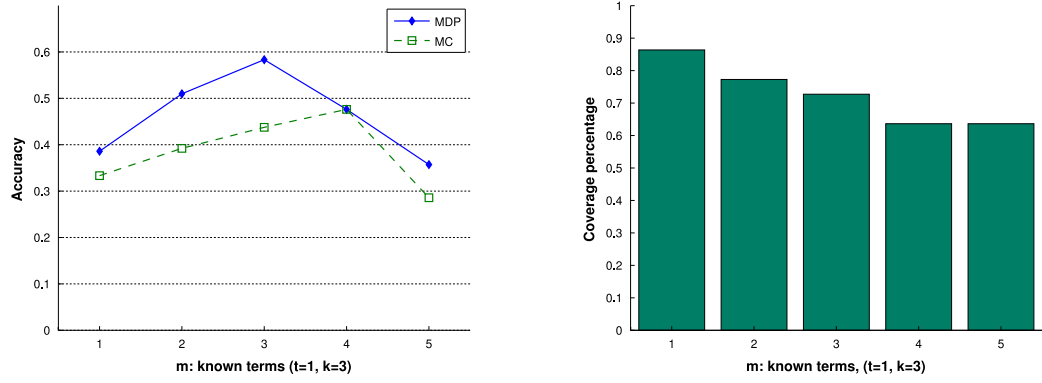
Figure (7–8) shows accuracy and coverage results for pruning threshold $\alpha = 0, 1, 2, 3$ when using SBP. The value $\alpha = 0$ implies that the algorithm is not performing any pruning on the allowed states. We can see in Figure (7–9a) that accuracy values are generally lower than those obtained by performing pruning. We conjecture that is because states may contain some outlier states that do not correspond to general patterns of student behavior, and therefore might introduce low accuracy performance. The coverage in this un-pruned case, however, is the highest since the state space is the largest among other α values. Increasing α values increases accuracy generally for all m “known” terms, and decreases the coverage as a result of a trimmed state space. It is shown in Figure (7–9a) that accuracy has quite good averages with acceptable average coverage ratios. In higher values of $\alpha = 2, 3$, we notice an interesting phenomenon of a deteriorating accuracy when $m \geq 3$. This is mainly caused of having very sparse states space, which cannot provide good recommendations when the problem gets harder, which is also illustrated in low coverage ratios.

Figure (7–8) shows accuracy and coverage results for the pruning threshold $\beta = 1, 2, 3, 4$ when using CBP. We notice that since pruning based on course frequencies is a more fine-grained approach, values differ less significantly than the previous case. We can observe that with $\beta = 3$, we get a good trade-off between accuracy and coverage.

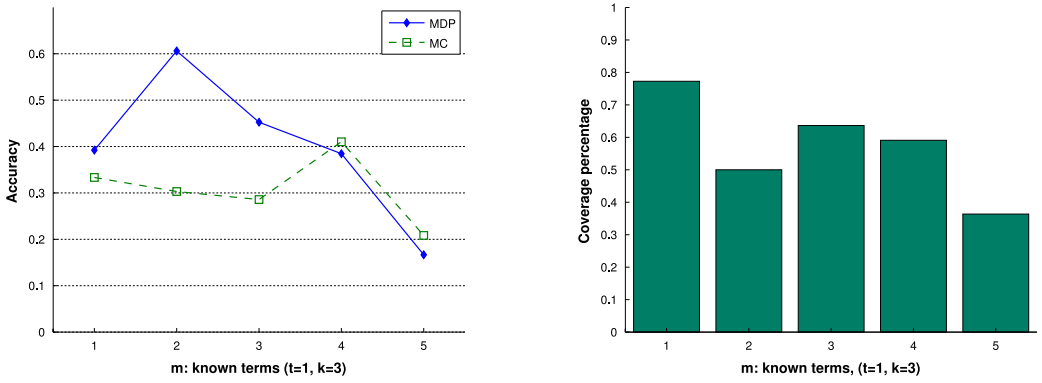
Figure 7–8: Alpha values effect on average *accuracy* and *coverage* vs. number of known terms when using state-based pruning (SBP)



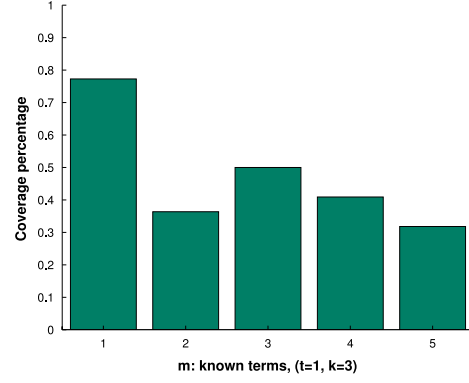
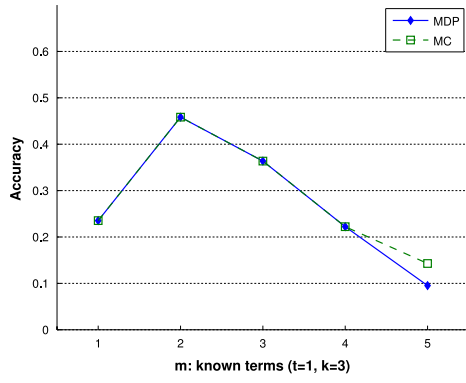
(a) $\alpha = 0$



(b) $\alpha = 1$

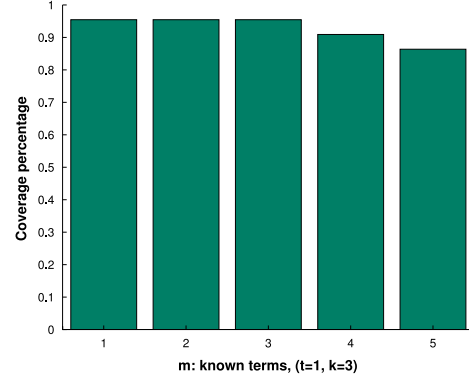
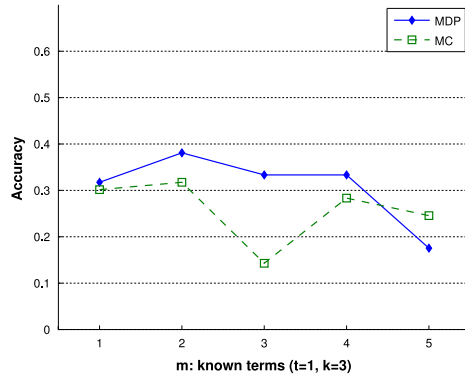


(c) $\alpha = 2$

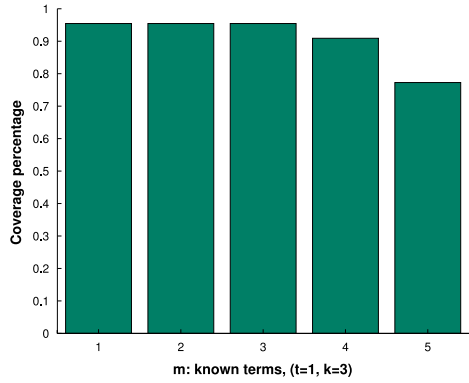
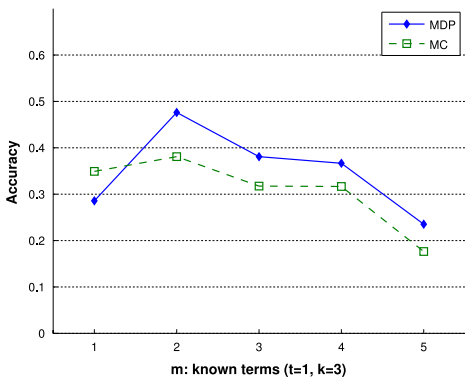


(d) $\alpha = 3$

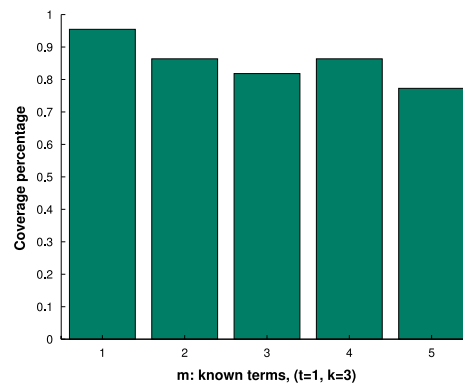
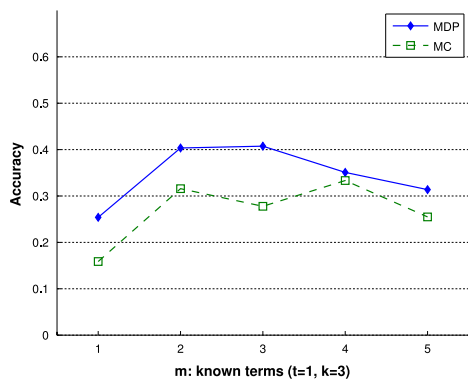
Figure 7–8: Beta values effect on average *accuracy* and *coverage* vs. number of known terms when using course-based pruning (CBP)



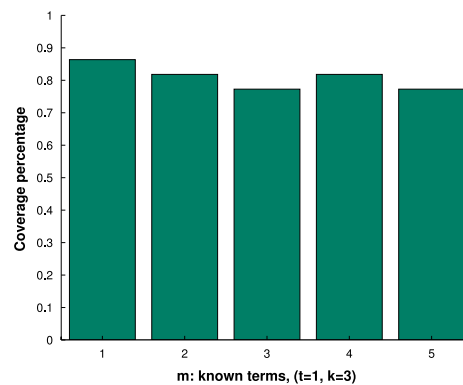
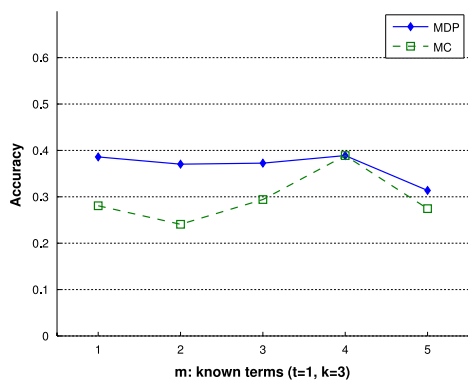
(a) $\beta = 1$



(b) $\beta = 2$



(c) $\beta = 3$



(d) $\beta = 4$

CHAPTER 8

Conclusions

8.1 Summary

In this thesis we have considered the problem of generating recommendations in domains where items are subject to strong inter-dependency constraints, in particular for course selections by university students. In this domain, the main challenge is to account for ordering constraints on courses due to hard and soft prerequisites while maximizing user preferences. Our approach is based on a Markov Decision Process (MDP) formulation of the recommendation problem. The main advantage of this approach is that it combines both student interests and course co-occurrence patterns to recommend personalized sequences of courses that take into account item inter-dependencies.

In order to estimate student preferences in courses, we proposed using two approaches: collaborative filtering, which predicts student interests by aggregating opinions of other like-minded students, and content-based filtering, which estimates interests in future courses based on opinions on similar courses that were previously taken. We have seen that using a combination of the two methods gives better preference estimation results than using each of them individually. We have found that liking a course and excelling in it are correlated phenomena, and hence we get similar preference estimation results when using each of them as basis for user preferences in courses.

Another interesting challenge in this class of recommendation problems is the combinatorial problem domain, which leads to a very sparse coverage of the recommendation space making the sparsity problem especially severe. By ignoring the unobserved subspace of the recommendation space and using frequency-based pruning, we have been able to obtain good recommendations.

Our experiments are based on real transcript data collected from students at McGill University through a web survey. These experiments suggest that combining both student interests and course co-occurrences give better performance results than employing each of them separately. Results show that our approach can outperform the two non-hybrid approaches in predicting the enrollment of students across several academic sessions in terms of accuracy and coverage values. Finally, we have shown that course plans recommended by our approach conform to course prerequisites in a similar degree to real transcript data, which is a result of employing course co-occurrences learned from the data.

8.2 Future Directions

This research can be extended in several interesting directions. Our approach can be directly applied to define the best course plan that satisfies specific career objectives of the student. This can be implemented by asking students to explicitly rate advanced courses based on their career objectives, and then find the maximum reward course plan that leads to the objective courses of the student. Alternatively, we can infer rewards of advanced courses based on association of these courses with a predefined list of majors or career objectives. Another interesting extension to our course recommendation approach is accounting for program requirements. In

fact, a useful course plan would be not only composed of courses that students will like, but would also help them to graduate. An initial study of considering program requirements in course recommendation has been discussed in [33].

Our reward prediction method is a hybrid method of collaborative filtering and content based filtering. We can point out several aspects that can be improved in this method in future work. We have chosen to use a memory-based method for collaborative filtering reward prediction for its simplicity and clear intuition; however, as the number of users in the system increases, the algorithm becomes computationally expensive. Model-based methods, on the other hand, provide better scalability. Methods such as SVD and SVD++ [26] have proven superiority in Netflix movie recommendation, and it would be interesting to investigate their performance in the academic courses domain. Additionally, we can observe from our experiments that the content-based reward prediction did not give good ranking results when used individually. We can use different approaches for building the user profile as described in Section 4.2, which might give better performance but with an extra time cost. Taking a linear combination of the collaborative and content-based methods gives the best performance, yet other combination strategies can be also tried for hybrid recommendations as suggested by Burke [7].

An interesting extension of our work is providing explanations to the user that justify the recommended course plan. Explanations in recommender systems help in increasing trust in the system and convincing users to accept the recommendation. In our case, the MDP policy chooses an action which maximizes the expected future reward, and hence the choice of courses might not be obvious to the student. A

similar idea was suggested by Dodson et. al. [11], where they used natural language explanations for policies generated by an MDP model used for academic advising.

The experimental evaluation of our course plan recommender system can be extended in several directions. A basic extension is to collect more Computer Science and Software Engineering transcripts, or to add data from other departments. Having more transcript samples can provide better prediction accuracy of the system as it can cover more of the course selections space. Additionally, we can investigate additional performance measures that go beyond the the standard information retrieval ones (accuracy and coverage). For example, we can examine the novelty of our recommendations by finding the unexpected course selections that our system is able to generate. A simple novelty measure can be implemented by the number of courses taken by a student which do not belong to the student's program. We can also use the collaborative filtering estimates for course reward and course similarity explained in Section 4.2 to create a measure of serendipity in our recommendations. Serendipitous courses are expected to have high collaborative filtering scores but low similarity to previous courses of the student. Another interesting future direction in evaluating our approach is by conducting qualitative evaluation of the designed course plans based on either an expert's experience (academic adviser) or the student's opinion.

We have assumed in this thesis that the MDP states can be represented as a table with one entry for each state. This is a reasonable assumption given the number of states (course tuples) observed from our dataset. When transcripts from all departments are incorporated, however, the number of states will enormously

increase, which will not only require large memory but also very long processing time. Therefore, a very important extension is to investigate *function approximation* methods for the value function of states that can generalize from the observed subset of state space, and to use Monte Carlo or Temporal-Difference learning algorithms for finding the optimal value function.

Finally, although we have focused our implementation on the academic courses domain, our approach can be applied to other domains where items are subject to ordering constraints. For instance, books can have strong ordering dependencies and therefore should be recommended in sequence. Another interesting domain is recommending web pages, where hyperlink structure and user browsing patterns can define dependencies between pages. Combining web browsing patterns with user personal preferences can be used to design a sequence of pages that should be viewed in order. A concrete application can be developed for Wikipedia ¹ pages. Wikipedia currently implements an Article Feedback Tool ², which assesses the quality of articles based on user opinions. Although the feedback is not personalized, it can be used in recommending sequences of high quality articles.

¹ <http://www.wikipedia.org/>

² http://en.wikipedia.org/wiki/Wikipedia:Article_Feedback_Tool

References

- [1] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowl. and Data Eng.*, 17:734–749, June 2005.
- [2] C. Anderson. *The long tail: Why the future of business is selling less of more*. Hyperion Books, 2008.
- [3] M. Balabanović and Y. Shoham. Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, 1997.
- [4] N. Bendakir and E. Aimeur. Using association rules for course recommendation. In *Proceedings of the AAAI Workshop on Educational Data Mining*, pages 31–40, 2006.
- [5] D. Billsus and M.J. Pazzani. User modeling for adaptive news access. *User Modeling and User-Adapted Interaction*, 10(2):147–180, 2000.
- [6] Daniel Billsus, Michael J. Pazzani, and James Chen. A learning agent for wireless news access. In *Proceedings of the 5th international conference on Intelligent user interfaces*, IUI '00, pages 33–36, New York, NY, USA, 2000. ACM.
- [7] R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.
- [8] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)*, 22(1):143–177, 2004.
- [9] Mukund Deshpande and George Karypis. Selective markov models for predicting web page accesses. *ACM Trans. Internet Technol.*, 4:163–184, May 2004.
- [10] Yi Ding and Xue Li. Time weight collaborative filtering. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, CIKM '05, pages 485–492, New York, NY, USA, 2005. ACM.

- [11] T. Dodson, N. Mattei, and J. Goldsmith. A natural language argumentation interface for explanation generation in markov decision processes. *Explanation-aware Computing ExaCt 2011*, page 1, 2011.
- [12] R. Farzan and P. Brusilovsky. Social navigation support in a course recommendation system. In *Adaptive Hypermedia and Adaptive Web-Based Systems*, pages 91–100. Springer, 2006.
- [13] A. Felfernig and R. Burke. Constraint-based recommender systems: technologies and research issues. In *Proceedings of the 10th international conference on Electronic commerce, ICEC '08*, pages 3:1–3:10, New York, NY, USA, 2008. ACM.
- [14] Matthew Garden and Gregory Dudek. Mixed collaborative and content-based filtering with user-contributed semantic features. In *proceedings of the 21st national conference on Artificial intelligence - Volume 2*, pages 1307–1312. AAAI Press, 2006.
- [15] Y. Ge, H. Xiong, A. Tuzhilin, K. Xiao, M. Gruteser, and M. Pazzani. An energy-efficient mobile recommender system. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 899–908. ACM, 2010.
- [16] D. Goldberg, D. Nichols, B.M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.
- [17] David Heckerman, David Maxwell Chickering, Christopher Meek, Robert Rounthwaite, and Carl Kadie. Dependency networks for inference, collaborative filtering, and data visualization. *J. Mach. Learn. Res.*, 1:49–75, September 2001.
- [18] J.L. Herlocker, J.A. Konstan, L.G. Terveen, and J.T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53, 2004.
- [19] Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR '99*, pages 230–237, New York, NY, USA, 1999. ACM.

- [20] Thomas Hofmann. Collaborative filtering via gaussian probabilistic latent semantic analysis. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, SIGIR '03, pages 259–266, New York, NY, USA, 2003. ACM.
- [21] Thomas Hofmann. Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.*, 22:89–115, January 2004.
- [22] T. Jambor and J. Wang. Optimizing multiple objectives in collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 55–62. ACM, 2010.
- [23] D. Jannach, M. Zanker, and M. Fuchs. Constraint-based recommendation in tourism: A multiperspective case study. *Information Technology & Tourism*, 11(2):139–155, 2009.
- [24] Joseph A. Konstan, Bradley N. Miller, David Maltz, Jonathan L. Herlocker, Lee R. Gordon, and John Riedl. Grouplens: applying collaborative filtering to usenet news. *Commun. ACM*, 40:77–87, March 1997.
- [25] Y. Koren. Collaborative filtering with temporal dynamics. *Communications of the ACM*, 53(4):89–97, 2010.
- [26] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 426–434, New York, NY, USA, 2008. ACM.
- [27] K. Lang. Newsweeder: Learning to filter netnews. In *in Proceedings of the 12th International Machine Learning Conference (ML95)*. Citeseer, 1995.
- [28] T. Mahmood and F. Ricci. Learning and adaptivity in interactive recommender systems. In *Proceedings of the ninth international conference on Electronic commerce*, pages 75–84. ACM, 2007.
- [29] J. Masthoff. Group recommender systems: combining individual models. *Recommender Systems Handbook*, pages 677–702, 2011.
- [30] Judith Masthoff. Group modeling: Selecting a sequence of television items to suit a group of viewers. *User Modeling and User-Adapted Interaction*, 14:37–85, February 2004.

- [31] Bamshad Mobasher. Data mining for web personalization. In Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl, editors, *The Adaptive Web*, volume 4321 of *Lecture Notes in Computer Science*, pages 90–135. Springer Berlin / Heidelberg, 2007.
- [32] Michael P. O’Mahony and Barry Smyth. A recommender system for on-line course enrolment: an initial study. In *Proceedings of the 2007 ACM conference on Recommender systems*, RecSys ’07, pages 133–136, New York, NY, USA, 2007. ACM.
- [33] A. Parameswaran, P. Venetis, and H. Garcia-Molina. Recommendation systems with complex constraints: A courserank perspective. 2009.
- [34] A.G. Parameswaran, H. Garcia-Molina, and J.D. Ullman. Evaluating, combining and generalizing recommendations with prerequisites. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 919–928. ACM, 2010.
- [35] A.G. Parameswaran, G. Koutrika, B. Bercovitz, and H. Garcia-Molina. Rec-splorer: recommendation algorithms based on precedence mining. In *Proceedings of the 2010 international conference on Management of data*, pages 87–98. ACM, 2010.
- [36] M. Pazzani and D. Billsus. Learning and revising user profiles: The identification of interesting web sites. *Machine learning*, 27(3):313–331, 1997.
- [37] Michael J. Pazzani and Daniel Billsus. The adaptive web. chapter Content-based recommendation systems, pages 325–341. Springer-Verlag, Berlin, Heidelberg, 2007.
- [38] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1994.
- [39] G. Salton. Automatic text processing: the transformation. *Analysis and Retrieval of Information by Computer*, 1989.
- [40] J. Sandvig and R. Burke. Aacorn: A cbr recommender for academic advising. *Technical Report TR05-015, DePaul University, Chicago, USA*, 2005.
- [41] Badrul Sarwar, George Karypis, Joseph Konstan, and John Reidl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th*

- international conference on World Wide Web*, WWW '01, pages 285–295, New York, NY, USA, 2001. ACM.
- [42] Badrul Sarwar, George Karypis, Joseph Konstan, and John Reidl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, WWW '01, pages 285–295, New York, NY, USA, 2001. ACM.
 - [43] G. Shani and A. Gunawardana. Evaluating recommendation systems. *Recommender Systems Handbook*, pages 257–297, 2011.
 - [44] Guy Shani, Ronen I. Brafman, and David Heckerman. An mdp-based recommender system. In *Journal of Machine Learning Research*, pages 453–460. Morgan Kaufmann, 2002.
 - [45] R.S. Sutton and A.G. Barto. *Reinforcement learning: An introduction*. The MIT press, 1998.
 - [46] N. Taghipour, A. Kardan, and S.S. Ghidary. Usage-based web recommendations: a reinforcement learning approach. In *Proceedings of the 2007 ACM conference on Recommender systems*, pages 113–120. ACM, 2007.
 - [47] Tong Zhang and Vijay S. Iyengar. Recommender systems using linear classifiers. *J. Mach. Learn. Res.*, 2:313–334, March 2002.
 - [48] A. Zimdars, D.M. Chickering, and C. Meek. Using temporal data for making recommendations. In *Proceedings of the 17th UAI Conference*, 2001.