NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

A MULTILAYER NETWORK APPROACH FOR REAL-TIME ADAPTIVE PERSONALIZED LEARNING

by

Jan-Daniel Cleven

December 2018

Thesis Advisor: Michelle L. Isenhour
Co-Advisor: Ralucca Gera

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** December 2018	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE A MULTILAYER NETWORK APPROACH FOR REAL-TIME ADAPTIVE PERSONALIZED LEARNING	**5. FUNDING NUMBERS** CHUNK
6. AUTHOR(S) Jan-Daniel Cleven	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) AETC, Randolph AFB, TX 78150	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

Traditional classroom instruction provides all students with the same topics, at the same time, at the same speed. However, personalized learning systems that take into account students' individual interests, skills, and preferences have been shown to be more effective. Our goal is to develop an adaptive teaching-learning method for enhanced and personalized education, which we call Curated Heuristic Using a Network of Knowledge for Continuum of Learning (CHUNK Learning). CHUNK Learning will provide a curated way of moving through a network of modules composed of educational material joined together by common attributes (i.e., tagged with competency or skill levels). Here, we use a network science approach to develop an initial software prototype, which recommends learning content to students based on their current interests. We establish methods to automatically extract metadata tags to describe course content and student profiles. These tags are used by the software to match courses with individual interests. Moreover, our software is able to simulate student feedback, which helps us test the updating features of our software. This allows us to adapt course recommendations to interest changes throughout the study period and to improve course recommendations for future students. This prototype builds the basis for a CHUNK Learning platform, in which the learner can heuristically discover or learn based on personal background, interests, and skills.

14. SUBJECT TERMS personalized adaptive learning, network modeling, network analysis, modular learning, network of educational modules, continuum of learning, learning object metadata, educational modules tagging, learning analytics	**15. NUMBER OF PAGES** 129
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

A MULTILAYER NETWORK APPROACH FOR REAL-TIME ADAPTIVE PERSONALIZED LEARNING

Jan-Daniel Cleven
Major, German Army
Dipl.-Ing. (FH), University of German Armed Forces Munich, 2009

Submitted in partial fulfillment of the
requirements for the degrees of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

and

MASTER OF SCIENCE IN APPLIED MATHEMATICS

from the

NAVAL POSTGRADUATE SCHOOL
December 2018

Approved by: Michelle L. Isenhour
Advisor

Ralucca Gera
Co-Advisor

W. Matthew Carlyle
Chair, Department of Operations Research

Wei Kang
Chair, Department of Applied Mathematics

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Traditional classroom instruction provides all students with the same topics, at the same time, at the same speed. However, personalized learning systems that take into account students' individual interests, skills, and preferences have been shown to be more effective. Our goal is to develop an adaptive teaching-learning method for enhanced and personalized education, which we call Curated Heuristic Using a Network of Knowledge for Continuum of Learning (CHUNK Learning). CHUNK Learning will provide a curated way of moving through a network of modules composed of educational material joined together by common attributes (i.e., tagged with competency or skill levels). Here, we use a network science approach to develop an initial software prototype, which recommends learning content to students based on their current interests. We establish methods to automatically extract metadata tags to describe course content and student profiles. These tags are used by the software to match courses with individual interests. Moreover, our software is able to simulate student feedback, which helps us test the updating features of our software. This allows us to adapt course recommendations to interest changes throughout the study period and to improve course recommendations for future students. This prototype builds the basis for a CHUNK Learning platform, in which the learner can heuristically discover or learn based on personal background, interests, and skills.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

1 Introduction **1**
1.1 Motivation 1
1.2 Problem Statement. 1
1.3 Purpose 2
1.4 Thesis Structure. 2

2 Background **3**
2.1 Hierarchy of Educational Material 3
2.2 Personalized Learning 4
2.3 Network Science in the Context of Personalized Learning 5
2.4 Network of Knowledge 6
2.5 Personalized Learning Framework 7

3 Data **11**
3.1 Data Description 11
3.2 Limitations. 18

4 Methodology **21**
4.1 Presentation of Educational Material as a Network of Knowledge 21
4.2 Personalized Learning Framework 24
4.3 Simulation of Student Experiences 34

5 Results and Analysis **43**
5.1 Automatically Tagging Metadata from Course Syllabi 43
5.2 Determination of Course Prerequisites 46
5.3 Additional Information Extracted from the Student Transcript Data 50
5.4 Simulation 52
5.5 Verification of the Software Prototype 60

6 Discussion **73**
6.1 Summary and Recommendations for Future Work 73
6.2 Conclusion and Outlook 77

Appendix **80**

A Results from Automatic Tagging of Metadata **81**
A.1 Extraction of Metadata from Course Syllabi 81
A.2 Extraction of Metadata from Course Content 84

B Determination of Course Prerequisites **91**
B.1 Determination of Course Prerequisites from the NPS Catalog and Student Transcripts 91

C Experimental Design **99**
C.1 NOLH Experiments 99
C.2 Distributions of Simulation Output Metrics 103

List of References **105**

Initial Distribution List **109**

List of Figures

Figure 2.1 Schematic Representation of the LoM Data Model 9

Figure 3.1 OA4801 Word Cloud 13

Figure 3.2 OS3113 Word Cloud 13

Figure 3.3 MA4404 Word Cloud 14

Figure 3.4 Network of Courses in the OR Department 16

Figure 3.5 Network of Shared Courses between OR Cohorts 18

Figure 4.1 Example: Multilayer Network of Knowledge 22

Figure 4.2 Example: Network Perspective for Course Metadata 24

Figure 4.3 Framework for Personalized Learning 25

Figure 4.4 Illustration of Tag Extraction from a Course Syllabus 27

Figure 4.5 Overview of Student Feedback 29

Figure 4.6 Example: Tag Updating Process 30

Figure 4.7 Course Network Builder 31

Figure 4.8 Sketch of the Recommender System 32

Figure 4.9 Simulation Overview 37

Figure 4.10 UML Representation of Students, Courses and Tags 40

Figure 5.1 Grade Histogram from Student Transcript Data 51

Figure 5.2 Change in Student Profile Tags over Time 54

Figure 5.3 Change in Course Content Tags over Time 55

Figure 5.4 Change in Course Content Tags over Time for a Single Course . . 56

Figure 5.5 Comparison of Individual Learning Paths Generated by Static Versus Dynamic Recommender System 58

Figure 5.6 Change in Tag Weights over Time with $\delta = 0$ 61

Figure 5.7 Change in Tag Weights over Time with Exclusively Positive Feedback Distributions 62

Figure 5.8 Change in Tag Weights over Time with Exclusively Negative Feedback Distributions 64

Figure 5.9 Correlation and Scatterplot Matrices of the Designed Experiment 66

Figure 5.10 Interaction of Input Variables and Readout Parameters in the NOLH Experiment 68

Figure 5.11 Average Dynamic Recommender Score versus Number of Student Profile Tags 69

Figure 5.12 Impact of Feedback Distributions on the Recommender Score . . 71

Figure C.1 Distributions of Simulation Output Metrics 103

List of Tables

Table 3.1 List of Courses 12

Table 3.2 Example Excerpt from the NPS Course Catalog 15

Table 3.3 Summary Statistics from the 10 OR Cohorts 17

Table 5.1 Five Most Common Words Extracted from OA Course Syllabi . . 44

Table 5.2 Most Common Words Extracted from MA4404 Course Materials . 45

Table 5.3 Derived Course Prerequisite Relationships 47

Table 5.4 Quantitative Analysis of Dynamic Versus Static Recommender System 59

Table 5.5 Quantitative Analysis of Recommender Systems ($\delta = 0$) 61

Table 5.6 Quantitative Analysis of Recommender Systems (100% Positive Feedback) 63

Table 5.7 Quantitative Analysis of Recommender Systems (100% Negative Feedback) 64

Table 6.1 Future Work 79

Table A.1 Term and Frequency List of Most Common Words from Detailed Analysis of Course Syllabi 81

Table A.2 List of Excluded Words from Detailed Analysis of Course Syllabi 83

Table A.3 Term and Frequency List of Most Common Words from Detailed Analysis of the MA4404 Course Content 84

Table A.4 Term and Term Frequency-Inverse Document Frequency (TF-IDF) Score of Most Common Words from Detailed Analysis of the MA4404 Course Content 86

Table B.1 Derived Prerequisites for Each Course in the Operations Research (OR) Department 91

List of Acronyms and Abbreviations

CHUNK	Curated Heuristic Using a Network of Knowledge for Continuum of Learning
DOD	Department of Defense
HSI	Human Systems Integration
IRB	Institutional Review Board
k-NN	k-Nearest-Neighbor
LoM	Learning Object Metadata
MOVES	Modeling, Virtual Environments and Simulation
NOLH	Nearly Orthogonal Latin Hypercube
NPS	Naval Postgraduate School
OA	Operations Analysis
OFAT	One Factor at a Time
OR	Operations Research
PDF	Portable Data Format
PPTX	Microsoft PowerPoint Open XML Presentation File
SEED	Simulation, Experiments, & Efficient Designs
TF-IDF	Term Frequency-Inverse Document Frequency
UI	User Interface
UML	Unified Modeling Language

THIS PAGE INTENTIONALLY LEFT BLANK

Executive Summary

Traditional classroom education teaches fixed contents at a fixed time and speed, which is generally adapted to the average student, but leaves individual preferences, interests or skills behind. Personalized learning, where educational content is presented to students in an individualized manner that takes into account their personal backgrounds and learning goals, is an emerging concept that is shown to be more effective. Still, such an approach makes great demands on the administration, organization and personalized selection of learning content.

Our goal is to establish a network science based real-time and adaptive teaching-learning method for enhanced and personalized education, which we call "Curated Heuristic Using a Network of Knowledge for Continuum of Learning" (CHUNK Learning). CHUNK Learning will provide a network of modules, composed of educational material, which are linked by common attributes (i.e., metadata tags describing content, competency or skill level, prerequisites, etc.). It recommends to the learner educational modules that match his/her individual profile, thereby adapting to any profile changes occurring during the period of study.

In this thesis, we develop an initial software prototype for this recommender system that will provide a basis for the CHUNK Learning platform. We use a network science perspective where educational material is organized in a multilayer network. Courses (represented as nodes) are linked by different categories of metadata tags (e.g., content, prerequisites,...) with each category forming one layer in the network. First, we analyze data from all courses offered by the Operations Research (OR) department at the Naval Postgraduate School (NPS) as well as data from former students enrolled in the OR curriculum. We use this data to automatically extract and assign tags describing learning content, course prerequisites, and student profiles. Next we develop an adaptive tag weighting system that reflects the relevance or priority of assigned tags for a course or in a student profile. Tag weighting allows us to dynamically adapt the importance of a specific tag for a course or profile in a real-time manner based on student feedback. Thus, the system is able to implement feedback and adapt course recommendations accordingly in a real-time manner. Furthermore, it can be used to improve course recommendations to future students, thus continuously optimizing

the learning platform. We then develop a dynamic recommender system algorithm that is able to match course content with individual student interests, thereby implementing feedback and adapting tag weights accordingly. The amount by which feedback affects course weights can be continuously adjusted via an input variable. Our software program is able to simulate student feedback based on input variables representing several feedback scenarios, such as defined feedback distributions with students influencing each other or not.

We compare this adaptive dynamic recommender system to a static system, in which tag weights remain constant throughout the experiment, but which works analogously in all other aspects. We can show that tag weights in the dynamic system do change over time, and that this results in changes in the recommended course path for individual students. Verification experiments confirm that these changes are indeed based on the implementation of feedback, and depending on the settings can improve the matching score of course content with student profiles. Lastly, we use a Nearly Orthogonal Latin Hypercube (NOLH) experimental design to modify input variables in order to identify and evaluate relationships between these variables and their overall impact on the system. Although this experiment should be expanded to a larger scale, results reveal information on ideal numbers of tags, the impact of feedback distributions on the recommender score and the overall changes in course paths.

Taken together, the work described in this thesis provides a prototype for a real-time adaptive personalized learning framework. From the network science perspective, we mainly focus on one layer (course content), but the methods suggested and established here can be applied to other layers. Implementation of these other layers in future work will greatly improve the quality of individual course recommendations as intended in the CHUNK Learning concept.

Acknowledgments

I want to thank my advisors, Michelle Isenhour and Ralucca Gera, for providing this interesting topic and letting me participate in all the inspiring discussions on the overall project. While you gave me enough space and freedom for my own ideas, you continuously advised, challenged my thinking and supported me. I particularly appreciate your detailed review of the thesis.

Furthermore, I want to thank Simona Tick for being so helpful with the IRB process, Michael Andersen for providing me the course catalog data, Jessica Duerstine for the preparation of the student transcripts, and Timothy Newlin for the inspiring discussions when I started working on this project.

I thank my classmates for being such great colleagues, for the fun we had in and outside courses, and for their helpfulness throughout the studying period.

I want to thank my son, Maximilian, for being such a happy little boy. You gave me all the entertainment and distraction I needed to recover and reset my mind.

Last, I want to thank my wife, Julia, for her strong support throughout the time at NPS. Thank you for keeping my shoulders free, for always motivating me in challenging phases, and for all the great experiences together, making this an unforgettable time in our lives.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

1.1 Motivation

In the modern world, which is marked by perpetual scientific progress and the swift exchange of data and knowledge on a global scale, students often struggle with the problem of information overload. Educational institutions are increasingly challenged to provide effective, focused, and well structured learning systems to guarantee on-demand high quality education across a large variety of topics. Given that millennials best perform in deep and rich learning environments, the ultimate goal is to streamline the learning experience by making information convenient and pushing information to the student rather than pulling it off the web or from professors (Stiles 2000; Herrington and Herrington 2008).

1.2 Problem Statement

In traditional educational approaches, learners interact with the same content, mostly in the same order, together at the same time and pace. This environment creates carbon copied educational experiences and similarly skilled learners, impeding diversity and performance and failing to enhance the individual learner's skills (Russell 2018). In today's technologically connected world, we all have personalized profiles on sites such as Amazon, Netflix, and Facebook, that push information to us based on our previous on-line behaviors and experiences. Yet, this is not the case for education. In traditional educational methodology, there is no educational profile that captures a learner's background, skills, interests, and experiences and matches the profile with recommended educational content.

The emerging concept of personalized and adaptive learning claims to fill this gap. Personalized education engages the learner, transitioning the focus from educational curriculum to skills and career, supporting life-long learners that have information at their fingertips, anytime, anywhere, and across many stages of life (Russell 2018; Pane et al. 2017).

However, a personalized learning strategy brings along new requirements: (1) students need to be profiled according their current (and future) personal interests, skills, talents and

learning preferences, and (2) learning material needs to be tagged in a way that optimal educational material can be matched to the student's profile. The current metadata associated with publicly available free educational content makes this connection difficult and does not provide a convenient delivery method to the learner.

1.3 Purpose

We aim to generate a learning environment that offers personalized adaptive learning, which we call the "Curated Heuristic Using a Network of Knowledge for Continuum of Learning" (CHUNK Learning) concept. The aim of this thesis is to explore a network science approach to offer a framework for personalized learning, in which students can move through a network of modules of learning material joined together by common attributes like skill level, competency, etc.

We specifically address the improvement of tagging of learning content with metadata in a way that allows its algorithm-driven identification and selection to fit the needs of the individual learner. Furthermore, we simulate students' learning paths as they progress through learning content, using real data from the Naval Postgraduate School (NPS). We analyze these data sets to set up learners' profiles, thereby exploring ways to implement adaptive feedback from the (simulated) students. This will build the basis for a CHUNK Learning algorithm that facilitates a personalized educational experience.

1.4 Thesis Structure

Chapter 1 provides the problem statement and the motivation for the thesis. Chapter 2 provides all necessary definitions, introduces a network science perspective to our problem at hand, and examines important literature in the field. Chapter 3 gives an overview of all sources of data used for this thesis and describes the preparation of our data. Chapter 4 presents the methodology used in this work: (1) a network science perspective, (2) an adaptive tagging model, (3) a hybrid recommender system, and (4) a simulation that generates student feedback. In Chapter 5, we show, interpret and discuss the results, and in Chapter 6 we provide a conclusion and suggest future work.

CHAPTER 2:
Background

In this chapter, we give an overview of background knowledge that is relevant for the thesis. We define important terms as we understand and use them throughout the following chapters, summarize important previous findings from this field of research as can be found in the literature, and highlight what we think needs to be further investigated or improved.

2.1 Hierarchy of Educational Material

Educational content is generally organized in a hierarchic structure. Students subscribe to a certain curriculum that consists of several courses. Each course has themes organized into chapters, and each chapter consists of several sections. In general, the smallest level of educational material relevant for learning is a section, whereas the smallest level a student can choose from is at the course level.

We now introduce terms describing the hierarchy of educational material, which are relevant for our CHUNK Learning approach and the models presented in this thesis.

- Microcredits: Generally, at most educational institutions or on learning platforms, students have to pass the course to get credits. We would like to explore the idea of a *microcredit* taught through a module that captures educational information at a different granularity than the course credit. This allows for choices on what microcredits can form the equivalent of credit hour for a course. Of course, restrictions on what microcredits can be combined to give a whole credit need to be put in place.
- Activity: This term is used for the smallest subdivision of educational material students experience or engage with. Activities can be videos, webinars, codes, games, articles, assessments, demonstrations or similar, and represent the atomic element of educational content.
- CHUNKlet: A CHUNKlet is a coherent collection of one or more activities. CHUNKlets form the basic building blocks in the assembly of a four part CHUNK.
- CHUNK: Course content is divided in several modules, so called CHUNKs. A CHUNK consists of several CHUNKlets arranged in a conventional structure usually

consisting of four parts: why, how / what, methodology, and assessment. This is the level where we envision the possible awarding of microcredits.

- Course: A course consists of several CHUNKs, may be taught or curated by one or more instructors, and is analogous to the typical credit hour course structure comprising a standardized period of study (e.g., quarter, trimester, or semester).
- Curriculum: A curriculum is a sequence of courses required to obtain a certification or degree. While some courses in the sequence are usually mandatory, some can be chosen, but typically need to be approved by the program of study.

2.2 Personalized Learning

Traditional teaching methodology mostly relies on a relatively rigid structure of educational content presented to a heterogeneous group of students. However, recent studies indicate, that learning efficiency can be dramatically improved, if personal prerequisites, skills, and individual learning preferences of students are taken into account (Pane et al. 2017).

Ideally, learning content would be presented to students in a personalized framework, reflecting their individual interests, previous knowledge, susceptibility for different educational methodology, and speed, often referred to as *personalized learning*.

Definition 2.2.1 *Personalized Learning*

> *Personalized learning refers to instruction in which the pace of learning and the instructional approach are optimized for the needs of each learner. Learning objectives, instructional approaches, and instructional content (and its sequencing) may all vary based on learner needs. In addition, learning activities are meaningful and relevant to learners, driven by their interests, and often self-initiated. U.S. Department of Education (2017).*

This implies that a variety of educational materials should be presented to students, based on their different learning profiles, delivered at different speeds, with variable content and depth of knowledge, which we call a *personalized learning framework.* Thus, we introduce the following definition:

Definition 2.2.2 *Personalized Learning Framework*

> *A personalized learning framework is the overall conceptual structure used to present educational material to an individual student, taking his/her interests, prior knowledge, learning preferences and learning goals into account.*

However, such an individualized approach requires both detailed profiling of the student's personal learning preferences, as well as an extraordinary collection and annotation of educational material. The latter is necessary in order to identify educational material best fitting the student's profile. Network science can provide important tools that help to achieve these goals, as we explore in this thesis.

2.3 Network Science in the Context of Personalized Learning

Network science is a relatively young discipline, which in particular helps to understand relations between entities and has great interdisciplinary application (Newman 2010). In the context of learning platforms, network science provides great tools to analyze, structure and connect metadata describing existing educational material. Furthermore, network science can help the user visualize and interact with available learning materials as a network of knowledge. Moreover, we can analyze the students' learning paths and behaviors, learn from them, and make better path suggestions to future students.

As the field of network science has emerged from a variety of research fields with different backgrounds, terminology in the literature is not always standardized. In order to avoid misunderstandings, the following section will define the most important terms as they are used in this thesis. We begin with the following definition of a *network*:

Definition 2.3.1 *Network*

> *A network is, in its simplest form, a collection of points joined together in pairs by lines ... A network is a simplified representation that reduces a system to an abstract structure capturing only the basics of connection patterns and little else (Newman 2010).*

We use the term network to highlight connections between different forms of educational content as well as relationships between educational material and its metadata. Referring to the above definition, single educational content or metadata are *points*, whereas their connection / relation are the *lines* joining them. In this thesis, we also use the term *nodes* for points and the term *edges* for lines. We use the term *directed edge* if there is a one-directional relationship between two nodes (e.g., one course is a prerequisite for another course). We draw a line connecting these two nodes with an arrow on one side, indicating the direction. Since we desire to display our educational content using multiple networks, we next define a *multilayer network*:

Definition 2.3.2 *Multilayer Network*

> *A multilayer network consists of a set of vertices, and a collection of layers, each of which represents a separate network of connections between the vertices.*
>
> $$M = (V_M, E_M, V, L), \tag{2.1}$$
>
> *where V is the total number of vertices in the Network, $L = \{L_a\}_{a=1}^{d}$ for elementary layers L_a for each aspect a, $V_M \subseteq V \times L_1 \times \ldots \times L_d$, and $E_M \subseteq V_M \times V_M$ (Kivelä et al. 2014).*

In this definition, *vertices* is used as another term for *points / nodes*. Multilayer networks are networks consisting of a fixed set of points / nodes / vertices but with variable lines / edges, where each combination of points and lines represents one *layer*. Each of those layers reflects one specific type of relation between points, referred to as *aspect* above. This provides an ability to look at a common set of points with more flexibility and from different perspectives.

2.4 Network of Knowledge

The expression *network of knowledge* is often used for the exchange of new scientific findings on a global level (Kogleck 2016; Nesshöver et al. 2016). This most commonly implies publication or presentation of new research findings in scientific journals, books, or at international conferences, but can be applied to all sorts of educational material.

In this thesis, we mostly refer to the latter and offer the following definition:

Definition 2.4.1 *Network of Knowledge*

> *A network of knowledge is the representation of educational material as a (multilayer) network, in which educational content builds nodes that are linked by common attributes in several categories, such as content, prerequisites, instructors, level, etc. Each category of attributes links the nodules in one layer, which provides an ability to filter the entire collection of material with different perspectives.*

From a network science point of view, single instances of educational material—on every level of the hierarchy described earlier—can be seen as nodes that combine to form a huge networked repository of information. Depending on the perspective, a network of knowledge can be a certain course, where course content or modules are the nodes, and then the prerequisites, instructor and previous knowledge of the students could represent edges. A simple version of this has been applied by the Khan Academy, which offers course content as nodes in a map, where different lessons are linked to each other if the knowledge of one lesson is a prerequisite for the next lesson (Khan Academy 2018).

To our knowledge, nobody has tried to extend this simple network model by adding further edges or layers. We hypothesize that this simple model could be extended up to a global level comprising all existing educational material combined in one multilayer network. Therefore, as part of this thesis, we aim at establishing a first theoretical model of a multilayer network of knowledge.

2.5 Personalized Learning Framework

Ideally, a personalized learning framework selects and offers educational content to students in an individually adapted order, speed, and learning methodology.

This brings up two major requirements: First, the system must set up and maintain an individual student profile that accounts for all factors which influence his / her optimal learning behavior. Second, all available educational content must be tagged and structured

in a way that allows the system to identify, select, and present optimal learning material to the student. For both requirements, an appropriate assignment of *metadata* is essential.

Definition 2.5.1 *Metadata*

Metadata is data that provides information about other data (Metadata 2018).

We next describe the selection and assignment of metadata as *tagging*.

2.5.1 Tagging of Educational Content

To ensure that learning content is reusable and can be found, selected, or exchanged, it is crucial that tagging is done in a meaningful, reproducible and ideally standardized manner (Keßler et al. 2013). In 2002, a standard for tagging of educational content was published by the Institute of Electrical and Electronics Engineers Standards Association, New York (IEEE-1484.12.1), the so called Learning Object Metadata (LoM). This standard provides a formal orientation, describing which type of metadata should be assigned to each learning object (Figure 2.1). Among others, these include: type of object, author, owner, terms of distribution, format, and pedagogical attributes, such as teaching or interaction style. Furthermore, they define what vocabulary should be used for these descriptions (Barker 2005). A handbook published by Steven J. Miller in 2011, which may serve as a guide for novices in need of experience in the work with digital material or metadata, takes this LoM standard as a basis (Miller 2011).

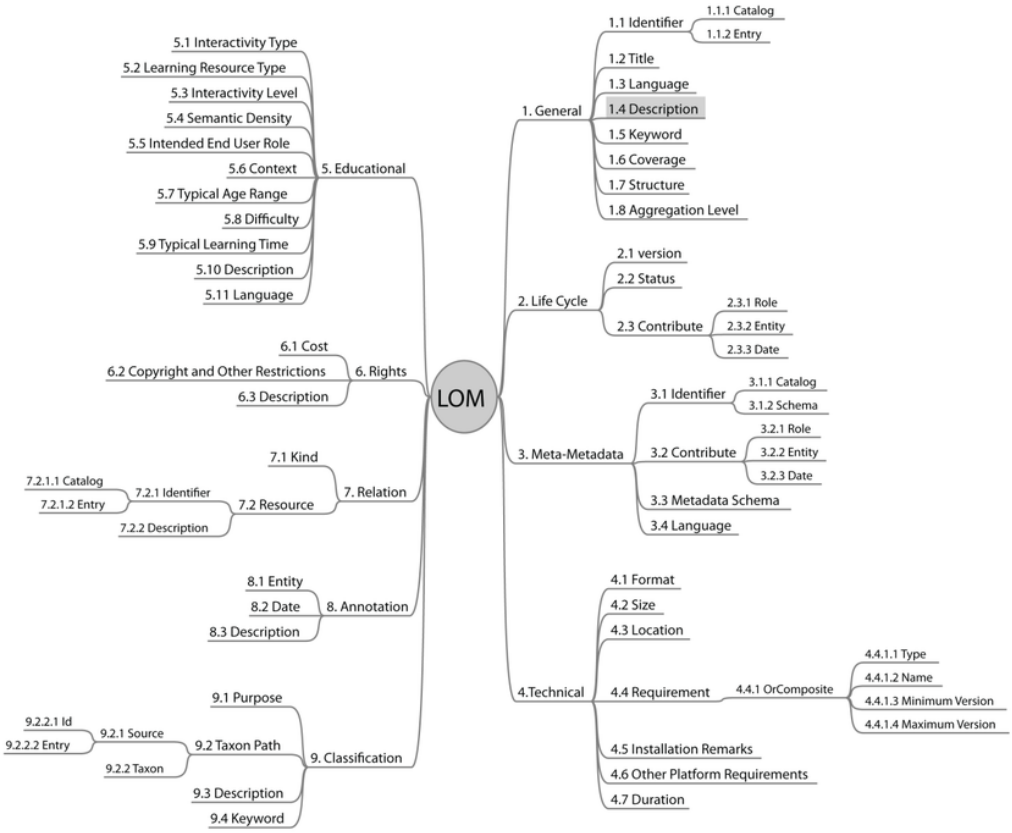


Figure 2.1. Schematic representation of the hierarchy of elements in the LoM data model. Adapted from Barker and Campbell (2010).

The LoM standard's main intention is to provide a general, universally applicable structure for learning content, that allows students, teachers, or automated software to identify and share learning content (Barker 2005). However, it does not include information that allows the flexible assignment of educational material to individual students according to their personal learning preferences.

2.5.2 Establishing Individual Learning Profiles

Personalized learning is an approach that allows students to select the educational material and methodology out of a large pool, that meets their unique preferences and abilities best. However, if selection of the appropriate material is left to the student, he/she might quickly be overwhelmed by the numerous kinds of learning materials, resources and activities (Wu

et al. 2015). Therefore, so called recommender systems (Burke 2002), which automatically generate personalized recommendations of appropriate material to the students, are important. This is the 21st century environment, in which people are accustomed to information being personalized and pushed to them, rather than being pulled from books and newspapers, for example.

According to Wu et al. (2015), such recommender systems for e-learning activities need to meet the following requirements: They need to be able to deal with the complexity of learning content as well as with the individual learner's learning characteristics, which includes personal background, learning goals, prior knowledge, learner characteristics, etc. (Wu et al. 2015). Furthermore, exterior circumstances, such as barrier-free accessibility for people with special needs, should be considered (Lazarinis et al. 2011). Finally, pedagogical aspects (e.g., avoiding extensive repetition of content, over- or undertaxing) should be considered to keep the learner motivated. All these aspects should be reflected when establishing the individual learning profile of students.

Finally, students' profiles must update during the course of studies. Learners' prior knowledge will increase, their focus or main interest might change, and variability in the learning method might as well increase their motivation and learning success. We thus consider methods to constantly include students' feedback into both, their personal profile and the tagging of learning content. This will also improve the accuracy of the tagging.

To involve student feedback into frameworks to improve them for the general public, representative feedback data is required. This requires either (*a*) trials with a large number of students, or (*b*) a modeling approach where student profiles, their feedback and its effect on the framework are simulated. Due to a lack of commonly representative data, this thesis takes the modeling approach. To our knowledge, this type of agent-based feedback approach has not been published before.

In the following chapters, we describe our approach to meet some of the challenges an adaptive personalized learning platform encounters. We particularly address the topics of tagging educational material and student profiles, establishing a recommender systems that matches the tags, and developing a simulation software for students' feedback, which will be used to update personalized course recommendations.

CHAPTER 3:
Data

In this chapter, we describe all the data we use as examples of learning content. We also present our pre-processing of data to derive content tags to build a network representation, to simulate learning feedback, and to recommend learning content. For all these computational tasks, we use the Python programming language.

3.1 Data Description

We analyze four data sets in this thesis: (1) a set of 84 syllabi describing courses offered in the Operations Research (OR) department at NPS; (2) the entire collection of educational materials (Microsoft PowerPoint Open XML Presentation File (PPTX) slides) from a single course; (3) an extract of the NPS course catalog database for the academic year 2018; and (4) anonymous academic transcript data from 10 former student cohorts in the Operations Analysis (OA) Curriculum in the OR department at NPS.

3.1.1 Data from Course Syllabi

The data we use for this thesis consists of 84 syllabi from the OR Department at NPS, 45 provided as Portable Data Format (PDF) documents and 39 provided as Microsoft Word documents (Operations Research Department at the Naval Postgraduate School 2016). The list of courses described in these syllabi is shown in Table 3.1.

Table 3.1. List of Courses

OA1600	OA2801	OA2900	OA3101	OA3102	OA3103	OA3201	OA3301
OA3302	OA3304	OA3401	OA3402	OA3411	OA3412	OA3413	OA3501
OA3602	OA3611	OA3801	OA3802	OA4101	OA4106	OA4108	OA4109
OA4118	OA4201	OA4202	OA4203	OA4205	OA4301	OA4333	OA4401
OA4406	OA4408	OA4414	OA4415	OA4602	OA4603	OA4604	OA4607
OA4611	OA4613	OA4655	OA4656	OA4657	OA4702	OA4801	OS2080
OS2103	OS3006	OS3007	OS3008	OS3080	OS3081	OS3082	OS3104
OS3105	OS3111	OS3112	OS3113	OS3180	OS3211	OS3307	OS3311
OS3380	OS3604	OS3640	OS3680	OS3701	OS4010	OS4011	OS4012
OS4013	OS4080	OS4081	OS4082	OS4083	OS4106	OS4118	OS4621
OS4680	OS4701	OS4702	OS4703				

These documents do not share a common template and differ in format. We extract the raw text from the PDF documents via the *PyPDF2* package (PyPDF2 2016). We then parse these documents for their most common words, excluding stop words, common English words and organizational words typically used in syllabi, such as days of the week, exams, lectures and so on. From these results, we generate a list of the top ten most common words found in each document (or all words if less than 10 were found). We use these most common words as initial tagging candidates to describe the content of each course in the form of keyword-like metadata.

Figure 3.1 shows a word cloud of the common words in the course OA4801 - Spreadsheet Modeling for Military Operations Research. The word cloud analysis reveals descriptive candidates, thus representing an example where this simple approach leads to useful results. Figure 3.2 shows the respective word cloud for OS3113—Data Analysis for Human Systems Integration (HSI) and Modeling, Virtual Environments and Simulation (MOVES). This word cloud consists of rather generic words, thus representing a less useful example of using word clouds to extract keywords from a syllabus. In these word clouds, font size is proportional to the frequency of the respective word in the original document. This figure is created with the Python package *wordcloud* (Mueller 2012).

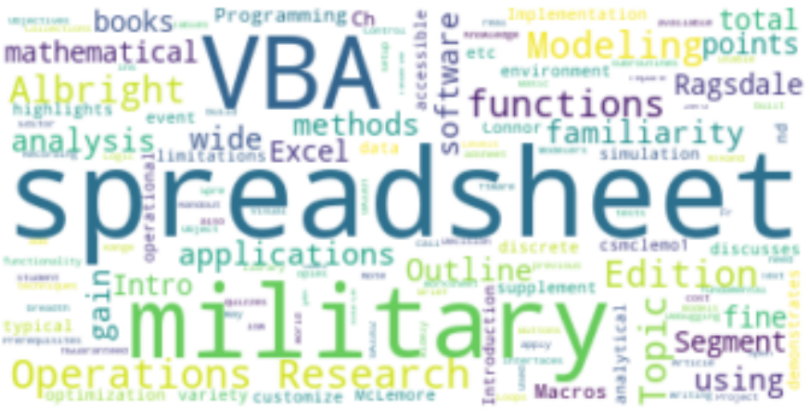


Figure 3.1. Word cloud of the most common extracted words for OA4801.

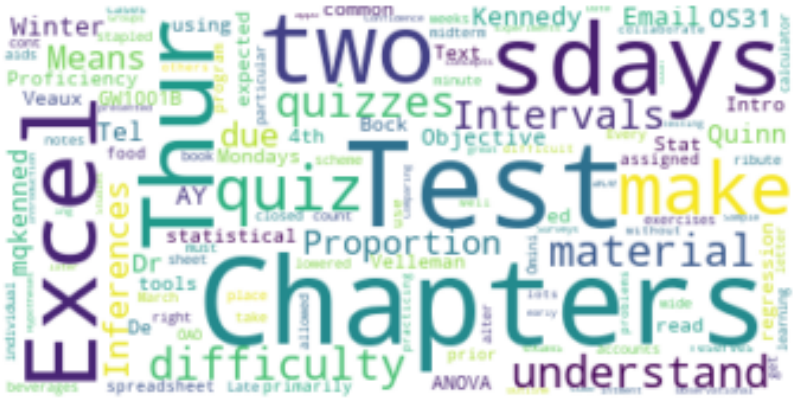


Figure 3.2. Word cloud of the most common extracted words for OS3113.

3.1.2 Data from Course Materials

In an attempt to identify better keyword tags for the course content, we perform an additional word count analysis on the entire collection of educational material for the course MA4404—Structure and Analysis of Complex Networks, for which the slides are publicly available (Gera 2018). We analyze the course as a whole, as well as each chapter separately. Figure 3.3 shows the resulting word cloud of lesson 2 as a representative example. As can be seen, the majority of words in this approach describes the actual content of the course. As this is representative for the other lessons of the course as well as for the entire course, we conclude that extracting common words from course material is a good way to automatically identify keyword tags.

Figure 3.3. Word cloud of the most common extracted words for MA4404 – Lesson 02.

3.1.3 Data from the NPS Course Catalog

To complement the list of courses from the OR department, we also obtained an extract of the NPS course catalog database for the academic year 2018, containing the complete list of courses offered by all departments at NPS in 2018 (Andersen 2018).

Table 3.2. Example Excerpt from the NPS Course Catalog

OA4202;4;0;N;None;As Required;OA3201 "Network Flows and Graphs Introduction to formulation and solution of problems involving networks, such as maximum flow, shortest route, minimum cost flows, and PERT/CPM. Elements of graph theory, data structure, algorithms, and computational complexity. Applications to production and inventory, routing, scheduling, network interdiction, and personnel management. PREREQUISITE: OA3201.";;;;;;
OA4602;4;0;N;None;As Required;None "Joint Campaign Analysis This course studies the development, use, and recent applications of campaign analysis in actual procurement, force structure and operations planning. Emphasis is on formulating the problem, choosing assumptions, structuring the analysis, and measuring effectiveness. Interpreting and communicating results in speech and writing is an important part of the course. In the last three weeks students conduct a broad gauge, quick reaction campaign analysis as team members. PREREQUISITES: A course in basic probability and statistic theory and operational experience in military environments.";;;;;;

From the course catalog extract shown in Table 3.2, it is straightforward to read course names. Yet, prerequisites are not listed in a standardized manner in the course descriptions: some refer to other NPS courses, others ask for prior knowledge in certain fields, etc. Thus, parsing the whole extract gives a variety of hits that cannot be directly compared. We therefore focus on the courses that refer to NPS courses as prerequisites, and use a web scraping technique to extract prerequisite information from the NPS catalog at nps.smartcatalogiq.com for validation (Naval Postgraduate School 2018).

Figure 3.4 shows a network of courses, where nodes are courses in the OR department and edges represent the prerequisite relationships. This figure is created with Gephi (Bastian et al. 2009).

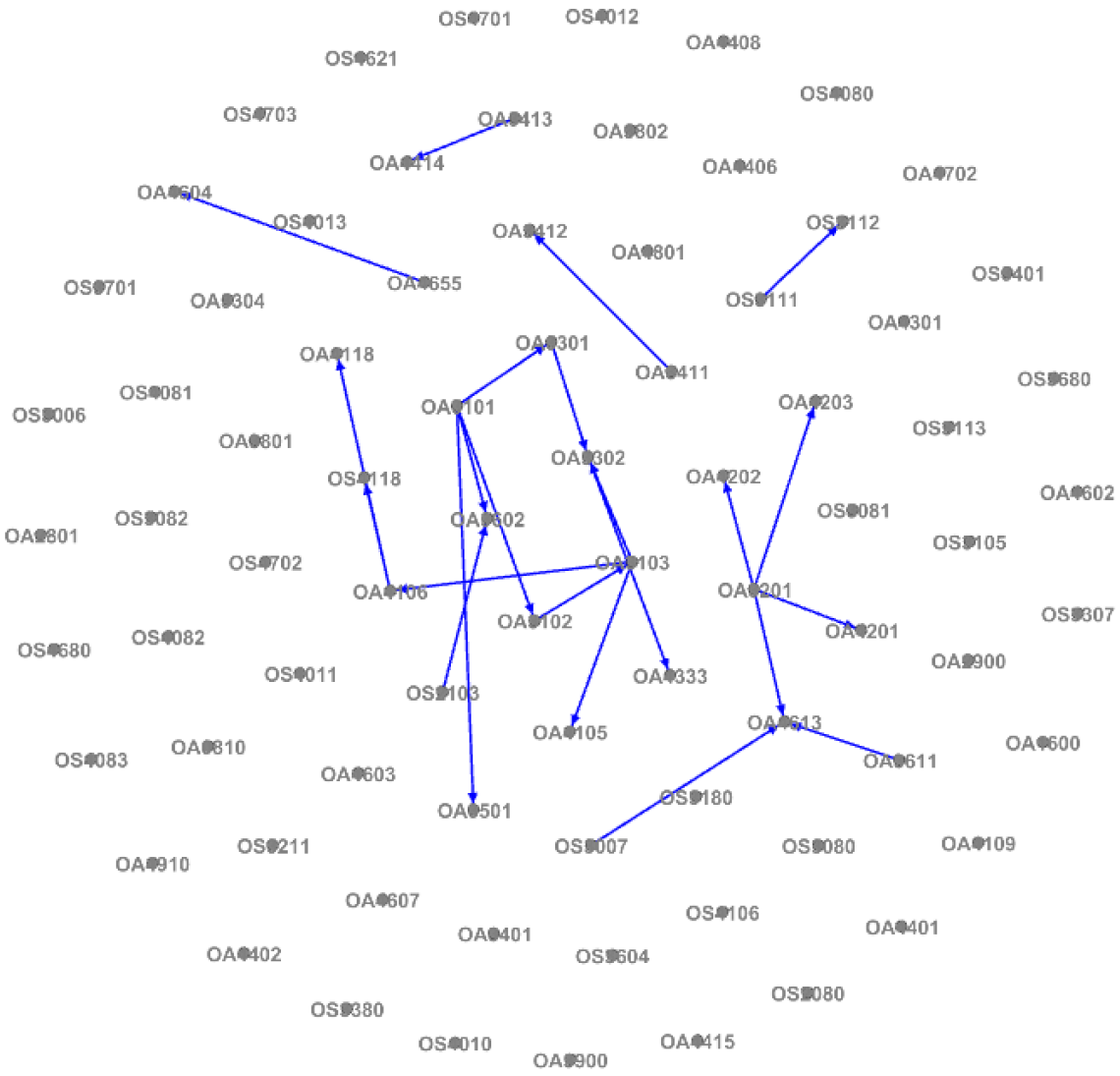


Figure 3.4. Network of courses in the OR department.

We use the data from Figure 3.4 to build the different layers of the multilayer network of courses for our research.

3.1.4 Data from Student Transcripts

Additionally, we use anonymous academic transcript data from 10 former student cohorts in the OA Curriculum in the OR department at NPS. We selected the transcripts at random, equally distributed from the 10 cohorts over a period dating from academic year 2001 to

2015. Usage of this data is approved by the Institutional Review Board (IRB) at NPS (Protocol Number: NPS.2018.0079-IR-EM1-A). The data set consists of 182 different courses taken by 97 students. Table 3.3 shows a breakdown of the demographics from the randomly chosen students. The transcripts include information on the quarter a course was taken and on the grade achieved.

Table 3.3. Summary Statistics from the 10 OR Cohorts

Gender		Nationality		Branch	
Female	13	U.S. students	83	Navy	55
Male	84	International students	14	Army	24
				Marine Corps	15
				Air Force	2
				DoD	1

We will use such data in future work to improve the simulation of course recommendations by adding layers covering student background, prerequisites, success rate or grading distributions to the personalized learning framework. In this thesis, we use the cohort data to automatically extract prerequisite courses.

Figure 3.5 shows a representation of the network of courses within the OR department, where each node represents a course. The node size indicates whether students from few (small node) or many (big node) cohorts have taken the course. Edges are drawn between two courses if they are offered to the same cohort. The edge weight—depicted by the saturation of green color—represents how many different cohorts are shared. This figure is created with Gephi (Bastian et al. 2009).

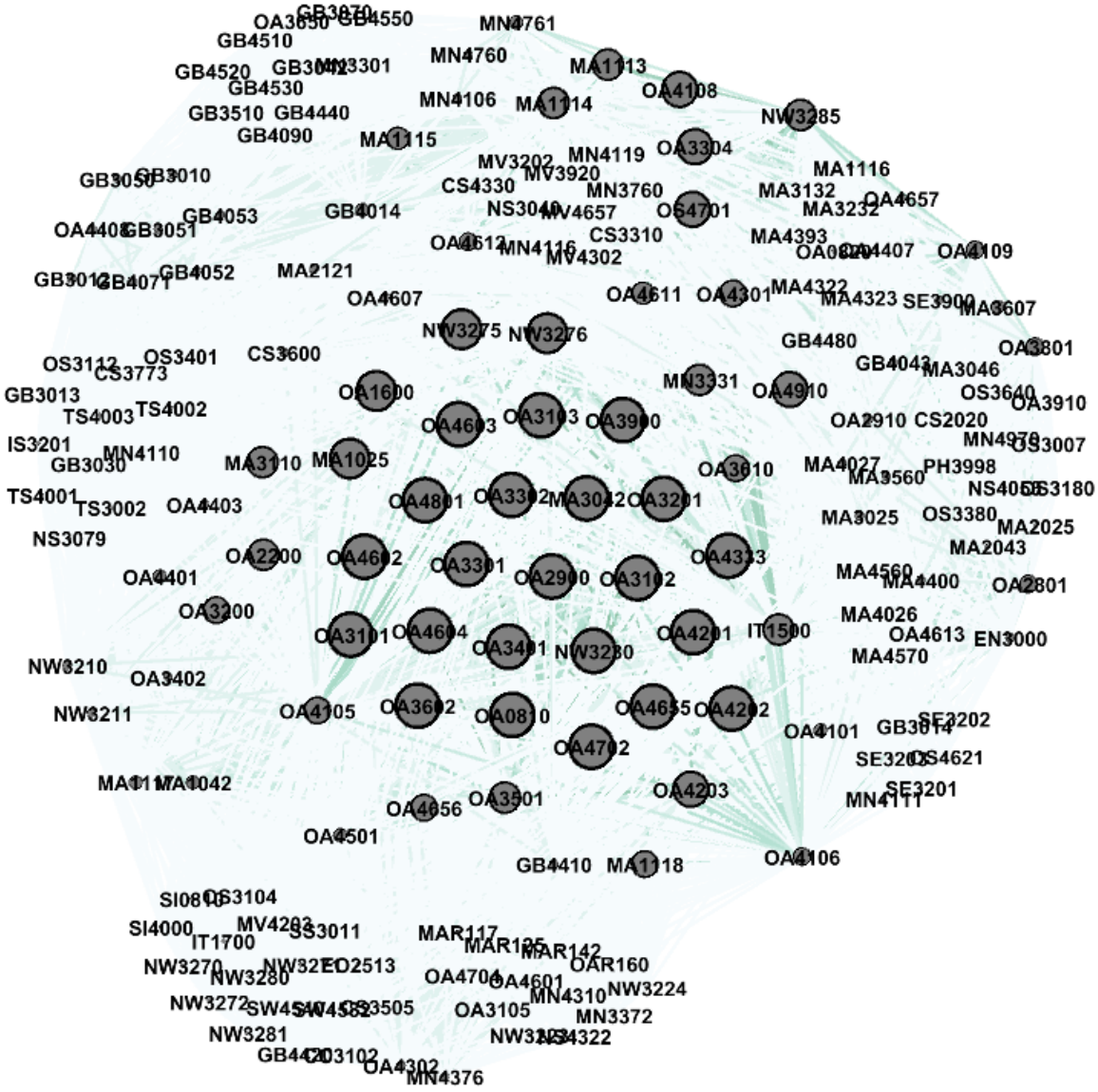


Figure 3.5. Network of shared courses between cohorts in the OR department.

3.2 Limitations

While the desired application of our overall learning approach is to general, publicly available courses and learners, the data analyzed here is limited to students from the OR department at NPS. This implies a preponderance of military students from different branches of service and different nationalities, with a prevalence of U.S. Navy officers, all

interested in the same curriculum. Furthermore, our data analysis stops at the course level and is limited to only OR courses. Further studies with data from other schools or learning platforms should be conducted to confirm the generality of our results. Also the male-to-female ratio should be carefully considered in the future, as NPS specific demographics might present skewed results.

More specifically, whereas the transcripts include data on all courses taken by a student, they do not provide information on which courses were mandatory for the degree program. Lastly, the non-standard format of the syllabi required the use of multiple parsing techniques. Thus the relevancy of the key words we obtain by parsing these syllabi are limited by the chosen algorithm (here: word count) and the ability of libraries for parsing different document types to handle all ways text could be contained within these files.

Taken together, this chapter describes the different kind of data used in this thesis and how it was processed. We present a more detailed description of the methodology behind our data analysis in the next chapter.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 4: Methodology

This chapter describes the methods we use to set up our network science approach to generate a framework for real-time personalized adaptive learning. In Section 4.1, we introduce our approach to represent educational content and all related information using tools from network science. Then, in Section 4.2, we describe how we organize relevant data by assigning and adapting metadata tags for the student and content, and how we use this data to establish an individualized recommender system which provides a personalized network of knowledge for each student. Lastly, in Section 4.3, we describe our algorithm to simulate student profiles and run the recommender system to automatically generate individual learning paths that take students' feedback during the studying period into account.

4.1 Presentation of Educational Material as a Network of Knowledge

The first step in establishing a network science based platform for personalized adaptive learning is to transfer educational material into a network, which we call the *network of knowledge* (see Definition 2.4.1).

Our network of knowledge has two components: The first component supports the student by connecting all courses and educational material available (based on prerequisites) which provides the student with a global view of how the learning materials fit together. The second component captures the metadata that is assigned to all educational content, as well as to user profiles, the latter allowing the system to select optimal learning content for the individual student.

We generate differing network perspectives for these two components, which allows us to utilize all relevant information in different ways depending on the purpose.

4.1.1 Multilayer Network

In the first perspective, which is most tangible to the students, each network node represents an educational module in the repository (i.e., a course and corresponding learning material). The metadata is represented by edges which link the nodes, thereby capturing different relationships between the modules. Depending on the type of metadata, a multilayer network is formed, where each layer focuses on a specific relationship, such as instructor, degree, type of course, and so on. Thus, within each layer, students only see a scaled portion of all courses available based on their profile preferences to avoid information overload. Furthermore, this multilayer approach allows students to filter educational material or traverse a specific metadata layer based on a particular interest (e.g., finding a course curated or offered by a particular instructor). Figure 4.1 shows examples of how courses can be connected in different network layers.

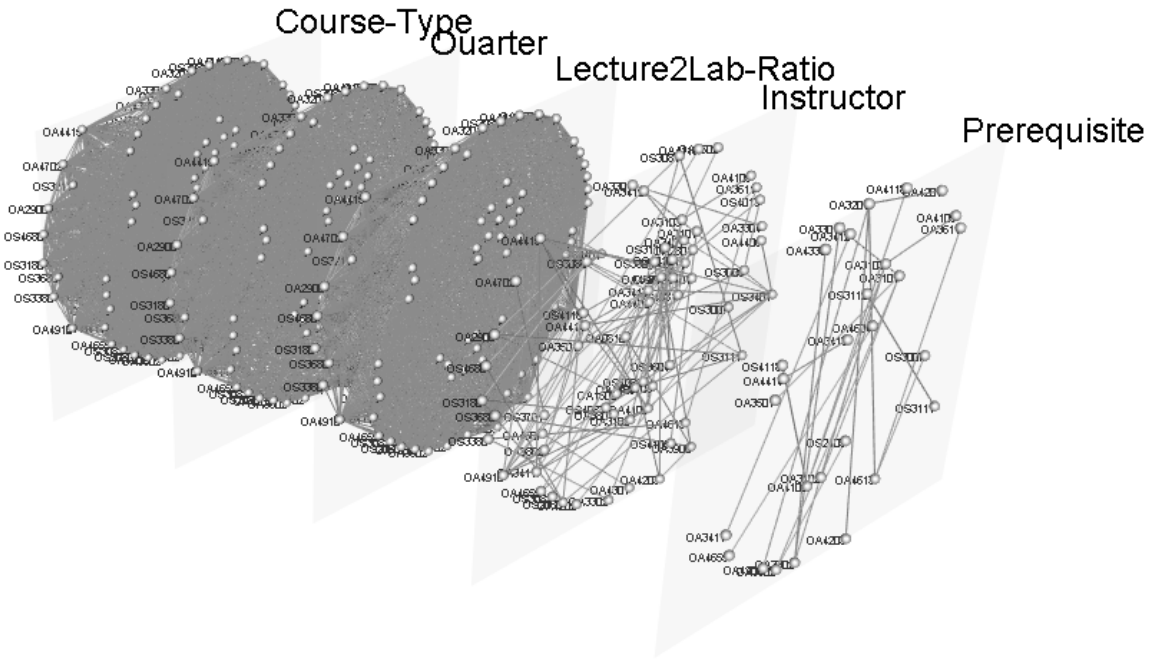


Figure 4.1. Example of different layers in a multilayer network of knowledge. The nodes represent courses offered in the OR department at NPS, linked by common tags associated with the indicated categories.

In the example shown in Figure 4.1, the first (right) layer shows *prerequisite* relationships of modules, with directed edges indicating that knowledge from one course is required to start the linked courses downstream of it. In the other layers, adaptive metadata connects courses if these two courses share the same *instructor* (second right), or have the same *lecture to lab ratio* (middle layer), and so on. This approach can be extended to various forms of metadata and filtered based on individual interest or preference.

To create the multilayer network shown in Figure 4.1, we first use the Python programming language to create nodes for the courses in the OR department and to build the different layers based on attributes for these courses from the NPS catalog (see Table 3.2). We then transform the data into the input format required by *muxViz*, a tool for multilayer network visualization and analysis (De Domenico et al. 2015).

4.1.2 Metadata Attributes

In addition to the attributes defined by the LoM standard (see Section 2.5.1), we tag each educational material with metadata of different categories that are relevant for personalized recommendations, such as content keywords, prerequisites, instructor, etc. Each student's profile is also tagged with individual attributes such as interests, prior knowledge, instructor preferences, etc. Using these tags, we generate a second network perspective for each course or each individual profile, in which the course or profile is the central node linked to all attributes as surrounding nodes. The linking edges in this perspective represent the weight of the linked tag, which serve to measure the relevance of the respective tag for the course or profile, respectively. This second perspective allows the visualization of tags from different categories for a course or profile at the same time, as well as information on the tag weights.

Figure 4.2 shows a schematic example of this network perspective for tags from several different categories.

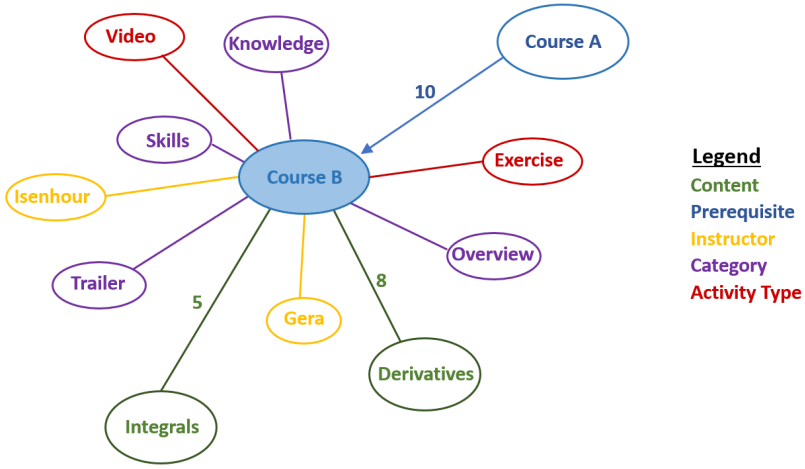


Figure 4.2. The center node is the course visible to the learner, surrounded by its metadata tags from different categories (highlighted in different colors). The numbers next to edges represent tag weights.

In Figure 4.2, different tag categories are highlighted in different colors. Other courses that are a prerequisite for that particular course (course A) are linked by directed edges, accordingly. Numbers next to edges indicate the weight of the respective tag. Tag weights can be changed during the period of study according to students' feedback, thus allowing a dynamic adaptation of both user profiles and course tags. These weights allow the system to continuously integrate user feedback and update the relevance of tags, which helps to improve future course recommendations for the current student as well as path recommendations for future students.

4.2 Personalized Learning Framework

In order to develop a system that takes into account all currently available information and continuously updates as new information becomes available during the personalized learning process, we structure the relevant components into three related layers[1]:

- a data model,
- an optimization model, and
- analytics.

[1]Here, the term layer refers to a part of the framework, see Definition 2.2.2, not to the layers as defined for the multilayer network, see Definition 2.3.2.

The data model incorporates all offered educational material including assigned metadata for each material, as well as all student profiles with respective tags. It also implies an adaptive tagging model in which tag weights can be updated according to collected feedback.

The optimization model identifies an individual learning path that takes into account the student profile, as well as the overall student User Interface (UI) experience and feedback. Based on this information, the optimization model then recommends relevant courses to the student, so the student experiences learning materials related to existing skills and information captured by the data model.

The analytics component monitors and analyzes course progress and feedback of the students, both individually and collectively. It provides instant feedback and gratification to the student regarding course completion and learning path progress. Furthermore, it provides insights on success rates and course perception by students to the instructors, content curators, and administrators of the educational material. Most importantly, it feeds this information back into the data model to update tags and into the optimization model to improve future course recommendations.

A schematic overview of this 3-layered learning framework is shown in Figure 4.3.

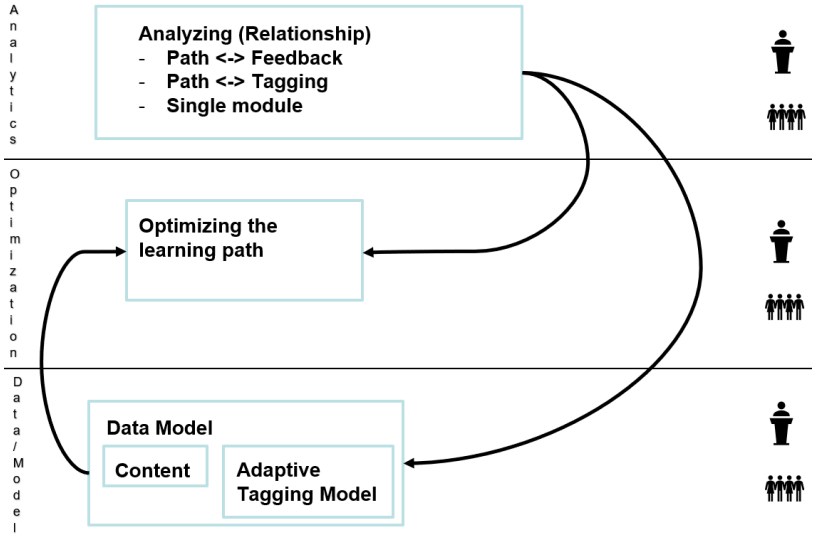


Figure 4.3. Components for Personalized Adaptive Learning.

In the following sections, we explain our vision and methodology for each layer of the personalized learning framework in more detail.

4.2.1 Data Model

The data layer manages all data relevant to the system. It consists of a repository containing all relevant data including courses, instructors, educational material, students, etc, and the settings required to manage the learning platform. Ideally, it provides an option to store, structure and identify all existing learning content, as well as current student data, with an option to be extended anytime in a consistent manner. Thus, the data layer must ensure data persistence and consistency (i.e., prevention of accidental data loss or redundancy) and maintain the relationships between the data. Furthermore, it should allow the addition, modification, or removal of data including metadata tags to/from the repository.

In this thesis, we focus on connecting educational content and prerequisites from the repository with student interests from the user profile data. In the future, we envision the addition of instructor/curator/administrator profiles, expansion of educational methods and extension of profile data to improve the system.

To enable the connection of educational content and students, a crucial part of the data model is the establishment of a tagging model which allows the consistent assignment of metadata for each instance of (newly added) learning content, as well as the adaptation of existing tags according to feedback or profile updates. This methodology is further described in Section 4.2.1.2.

4.2.1.1 Extraction of Course Tags

As previously stated in Chapter 2, we plan to use the LoM standard as a basis to tag all educational material, but complement it with tags describing course content, prerequisites, type of learning activity, etc., as these attributes are relevant in the development of our personalized learning framework. Manually finding these tags for all of the educational materials can be very time consuming and somewhat subjective. Thus, we look into ways to automate this process, thereby focusing on keyword tags which describe course content. Figure 4.4 illustrates the process of extracting content tags from a course syllabus.

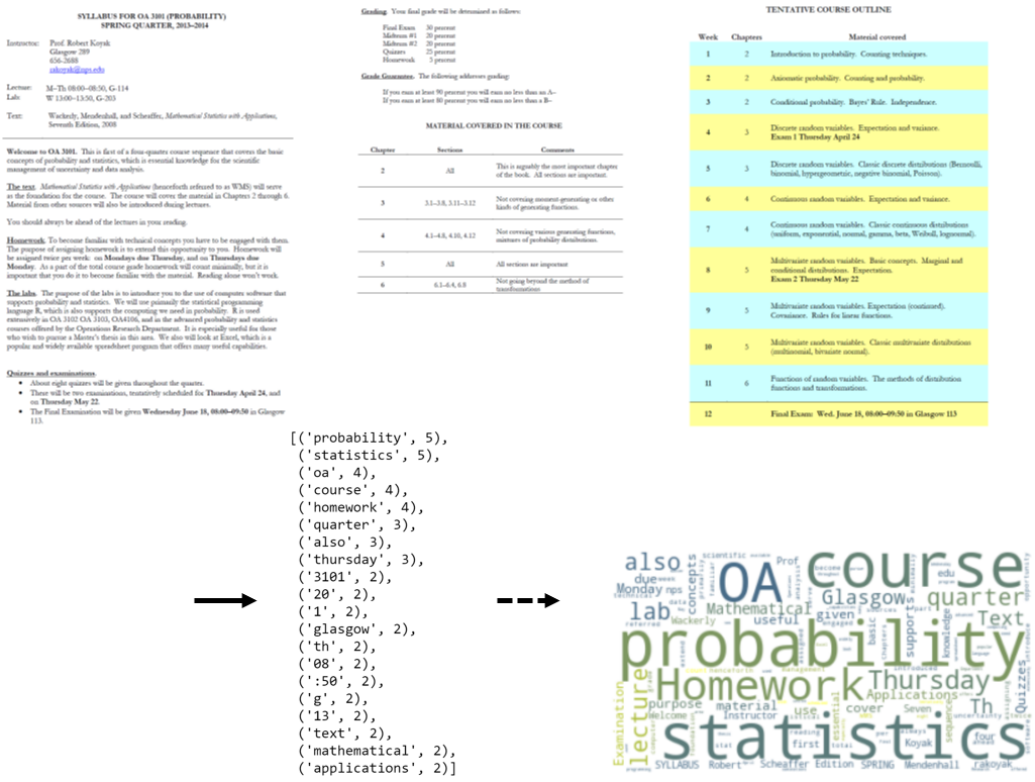


Figure 4.4. Illustration of the tag extraction process from a course syllabus.

At the top of Figure 4.4 is the three-page syllabus from OA3101, a course offered by the OR department. The syllabus was provided as a PDF document, as described in Chapter 3. We then extract the raw text from the PDF document via the *PyPDF2* package (PyPDF2 2016) and parse the document in order to find the most commonly occurring words, as shown in the bottom middle part of Figure 4.4, manually excluding stop words, common English words and organizational words typically used in syllabi, such as days of the week, exams, lectures and so on. A list of excluded words including exclusion criteria can be found in Table A.2.

For a visual representation, a word cloud is automatically generated from the course syllabus, as seen in the bottom right corner of Figure 4.4. From this analysis, we generate a list of the top ten most common words contained within the OA3101 syllabus. We use these words as our initial keyword tags to describe the content of OA3101 in the form of metadata. This automated tag extraction process was repeated for every syllabus from every course in the OR department (see Table 3.1).

A very similar method is used to extract the most commonly occurring words out of actual course materials. To extract content keyword tags for each lesson or module separately, we complement the above process by a term frequency-inverse document frequency (TF-IDF) score (Ramos 2003), which identifies the most relevant words for each document in the content collection (i.e., lecture materials).

4.2.1.2 Adaptive Tagging Model

In order to continuously update and improve tags, we envision and introduce an adaptive tagging approach, that takes into account assessment and evaluation of tags by both instructors and students. First, instructors select tags from a predefined pool. New tags can be added to this pool, but should be audited in accordance with predefined standards. The instructor then assigns weights between 1 and 10 to all tags associated with the course. Alternatively, an automated tag extraction system as described in Section 4.2.1.1 generates and weighs content keyword tags according to their frequency. Only one tag of each category can be assigned the highest weight which avoids ambiguity. The weight cannot fall below 1 or increase above 10.

For the purpose of our analysis, we obtain the starting weight for course content tags from the extraction method described in Section 4.2.1.1, giving the most common word a weight of 7.5, the second most frequent word a 7, and so on down to 3 for the 10th most frequent word. We choose not to exploit the whole spectrum from 1 to 10 as starting weights to allow for adjustments in both directions for all tags. This enables us to verify that the simulation implements both positive and negative feedback correctly. Similarly, we assign all interest tags in the student profile a starting weight of 5, which can increase or decrease accordingly as the student progresses along his/her learning path.

Learners have the option to provide feedback on whether the course was relevant for them, whether the content met their expectations, and/or whether they would recommend the course to others. This feedback is used to adapt the weight of both course tags and user profile tags.

Figure 4.5 schematically shows the relationships between student feedback, student profile, and course tags.

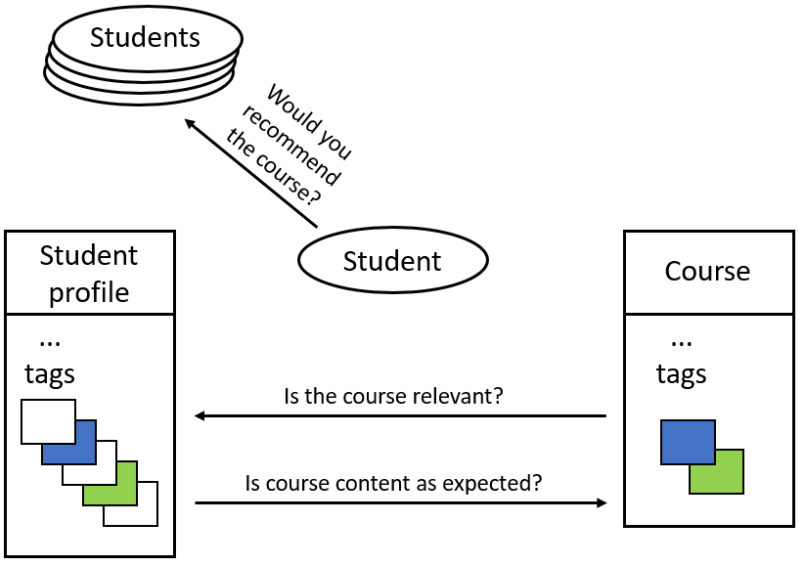


Figure 4.5. Overview of Student Feedback.

In this thesis, student feedback is collected on the following three questions: (1) Was the course relevant for you? (2) Did it meet your expectations? and (3) Would you recommend it to others? For simplicity reasons, only two feedback grades (positive or negative) are considered. A positive or negative answer on question (1) will increase or decrease, respectively, the weight of interest tags in the student profile that were relevant for the course recommendation. Accordingly, feedback from question (2) will positively or negatively affect the weights of course tags and results from question (3) is used to rate the course.

Student feedback invokes changes to the previous tag weight (i.e., the weight this tag had right before feedback) by a specified constant, δ. Positive or negative feedback will increase or decrease the original tag weight by δ, respectively. The value of δ can be varied in the simulation, thus allowing us to modify the overall impact a single feedback response has on the tag weight. Tags falling below a certain weight threshold become suspended. These tags become irrelevant and are no longer considered in the process of matching course content with student interest. In a real case scenario, we imagine the instructor (or content curator) would get informed that students evaluated his/her tag as insignificant.

Figure 4.6 illustrates this process of updating tags.

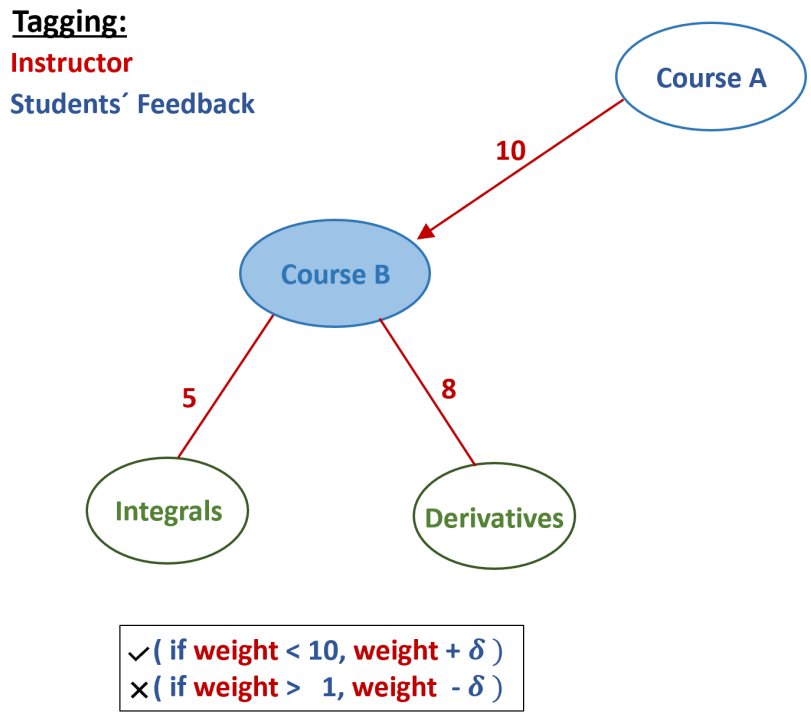


Figure 4.6. Example of an update to course content tags.

In this thesis, we simulate feedback, as described in Section 4.3, and immediately implement the results of the feedback. In a real case scenario, we imagine that feedback would be collected from real students via a survey following completion of the course. As this approach requires a methodology that enables easy selection and weighting of tags by the students (and instructors if initial tag weights are not automatically assigned), we design a program prototype that requires the instructors (or content curators) to add the metadata for a course from a predefined pool of keywords.

To demonstrate this vision, we build a tool in Microsoft Excel, see Figure 4.7, which allows users to set the metadata for each course and generates an interactive visualization of the course network.

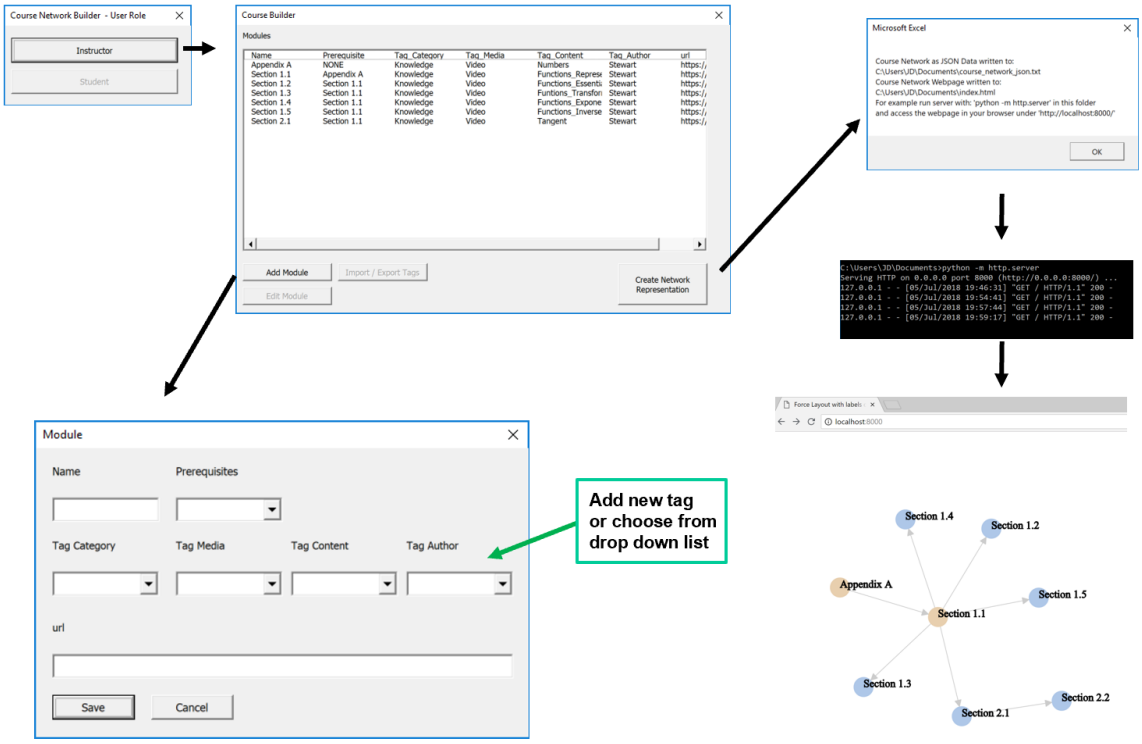


Figure 4.7. A tool to build the relation between course content and to set the metadata.

As can be seen in the lower left corner of Figure 4.7, upon adding a module to the course, the instructor selects tags from our standardized pool. Furthermore, the tool provides an interactive visualization of the course network, shown in the lower right of Figure 4.7. We create this representation using *d3.js* (Bostock et al. 2011) and present it as an interactive web page. Upon clicking on a node the user gets directed to the content, and the respective node changes color, representing completion, which is stored in a cookie.

4.2.1.3 Student Profiles

Ideally, student profiles contain all types of information that would be relevant in the establishment of an ideal (or optimal) learning path. This includes personal information such as name, contact information, age, or profession, as well as educational background such as previous knowledge, experiences, certifications, or completed courses. It might also include subjective items such as learning preferences, personal interests, and goals. In this thesis, we mainly focus on a student's interest in learning content. As our analysis derived

from real student data does not contain information on interests, we simulate this portion of the student profile in the software. To this end, we generate a pool of the content tags from all courses offered, and randomly assign 30 of these tags to each simulated student profile. We choose 30 tags for our experiments to ensure a decent overlap with the 10 extracted course content tags from Section 4.2.1.1. All interest tags get a starting weight of 5 when the student begins his educational journey, and are increased or decreased by δ based upon positive or negative feedback, respectively.

4.2.2 Optimization

The second major layer of our personalized learning framework is the establishment of a personalized recommender system that suggests appropriate educational material to students based on their individual profiles. The overall aim is to set up a recommender system that is able to identify an ideal learning path for each student, thereby taking into account students' interests and preferences during the whole learning process with each course fitting the students profile throughout the period of study. This includes ensuring courses that were already taken by a student are not recommended again even though they still fit his/her interests. Thus, the recommender system needs to rely on both the learning content and student profiles, both of which are continuously updated according to learning progress and personal feedback. Our recommender system evaluates metadata from student profiles, the tag history of course metadata, and data from other students' profiles (i.e., which courses received positive feedback from other students with similar profile interest tags), as shown in Figure 4.8.

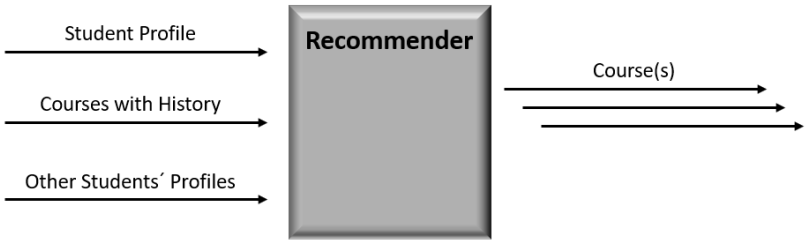


Figure 4.8. Sketch of the Recommender System.

From these three parameters, we calculate a similarity score for all offered courses and recommend the course with the highest score. The methods we recommend using to

determine what we call a recommender score is described in the following two sections.

4.2.2.1 Similarity between Student Profile and Course Tags

To identify the course that best matches a student's current profile, our system looks for the highest similarity between tags which describe the student's interests (contained in the profile metadata) and the tags assigned to the offered courses. We represent the different attributes of a student's profile as a vector and the available course metadata also as a vector. We then calculate the distance between the student profile and each course offered, excluding courses that the student already took. We use the cosine similarity to measure this distance, as this allows us to identify partial tag matches independent of the overall quantity of tags. This is important for matching course content, as a student profile usually contains many more tags than the courses. Thus, even if a student profile contains a large amount of information not related to course content, and consequently the overall ratio of overlapping tags is small, a course containing some of the profile tags can be identified as a match. We then sort the results, exclude courses that were previously taken by the student, and suggest the courses that fit the student's profile best.

4.2.2.2 k-Nearest-Neighbor (k-NN) of Course Ratings

To identify courses that should be recommended to a student based on other students' experience and feedback, we use the k-NN method. This method compares either courses or student profiles with each other, looking for the k, k being an integer, courses or student profiles, respectively, with the highest proximity, which are then called the k nearest neighbors. Among all offered courses that a student has not previously taken, we identify the k nearest neighbors based on feedback received from other students who have completed the course. For each of the courses, a k-NN score is calculated which indicates how many neighboring courses overlap with courses that were previously taken by a student. The course with the highest score is most likely to fit into the student's learning path.

The idea behind comparing student profiles is based on our opinion that students with similar profiles are likely to have similar optimal learning paths. Thus, we compare student profiles, look for those students with the best matching profile tags, and suggest courses that were previously taken and positively evaluated by the student with the most similar profile.

4.2.2.3 Recommendations and Path Optimization

Within the simulation, the course that is eventually recommended to the student is selected using a hybrid approach which combines several techniques. For the software prototype we establish to support this thesis, we combine the content similarity and the k-NN method. As the k-NN method relies on previous experience by other students, during the initial phase of the simulation only the content similarity approach is used to make course recommendations. Once enough course history is established (e.g., once 50% of all courses have been taken at least one), the k-NN method is used.

The k-NN method or a hybrid of both methods are contained in the software prototype described in Section 4.3, and can be switched on and off. However, an analysis on how they affect course recommendations in comparison to just the cosine similarity approach is not part of this thesis and will be addressed in future work (see Section 6.1.2).

4.2.3 Analytics

Generally, the analytics layer is meant to provide instant feedback to students regarding their progress in terms of course completion and grading. Furthermore, it shall give feedback to instructors or administrators of the educational platforms on course evaluation by students (satisfaction, relevance, educational approach, fulfillment of expectations, etc.), statistics on participation, grading outcome, and so on. In this thesis, we include in our prototype the analysis of several of such metrics, including the tracking of tag weights over time, courses over time, and course feedback over time. In order to keep things simple during the initial phase of this research, we work with simulated students instead of real users and do not yet include instructors or content curators. Therefore, feedback important to both these entities is not a focus of this work and respective methodology should be included as part of future research (see Section 6.1.2).

4.3 Simulation of Student Experiences

We employ a surrogate simulation approach for students' feedback. This simulation includes personalized course recommendations to students based on their simulated profiles, models their feedback based on a statistical distribution, and adapts further course recommendations based on this feedback, resulting in a specific learning path for each individual student. The

simulation is implemented in the Python programming language. The following sections describe the inputs, processes, and outputs for our initial prototype, and the last section describes the limitations of our simulation model.

4.3.1 Simulation Input

Whereas the simulation generally runs through a fixed sequence of steps (see Figure 4.9), we construct the simulation so that several input parameters and settings can be varied. This enables us to investigate how controllable parameters, that can be influenced by an instructor or administrator (e.g., decision factors), affect the overall result while taking into account additional factors that cannot be externally manipulated or controlled. For example, instructors could influence the quality of educational material or the grading distribution, but cannot directly influence student feedback. Thus, varying the input parameters describing educational material or the grading system in the simulation will help to predict the influence of these factors on student feedback, helping the instructor choose the best strategy for her/his course design.

Listing 4.1 provides an example of the simulation configuration file. The listing shows the default input parameters required to initialize a simulation run. Through the use of a configuration file, we are able to not only vary values of the parameters, but we are also able to define statistical distributions. The values shown in Listing 4.1 are the parameter values and distributions used to simulate student experiences during our initial simulation run, and we consider these values to be the default or baseline for our simulation.

Among others, we made the following initial assumptions in Listing 4.1: 100% of the students provide feedback on the course and on content tags; 70% of this feedback is positive and 30% is negative; 70% of the students pass a course, 10% fail and 20% drop the course; and δ is set to 2.5 to allow a rather strong effect of feedback on tag weights.

Furthermore, we define the number of students simulated at each time step as 50, the total simulated period of time as 40, and the period of study for each simulated student as 8, where each time step represents 1 simulation loop, which in our case is one academic quarter.

Listing 4.1. Configuration File—simulation.ini

```
[DEFAULT]
distribution_grade = [0.7, 0.1, 0.2]
dict_of_grades = {"1": "passed", "0": "failed", "-1": "dropped"}
distribution_feedback = [0.56, 0.14, 0.3]
dict_of_feedbacks = {"1": "positive", "-1": "negative", "0": "none"}
distribution_tag_feedback = [0.7, 0.3, 0.0]
distribution_tag_feedback_trend_neg = [0.20, 0.80, 0.0]
distribution_tag_feedback_trend_pos = [0.80, 0.20, 0.0]
distribution_tag_feedback_learner = [0.7, 0.3, 0.0]
trend_persistence_time = 5
settings_trend_included = False
community_join_probability = 0.5
settings_communities_included = False
communities_number_min = 1
communities_number_max = 5
settings_recommendation_cos_sim = True
settings_recommendation_knn_course = False
settings_recommendation_knn_profile = False
settings_recommendation_hybrid = False
settings_simple_example = False
tag_max_weight = 10
tag_min_weight = 1
tag_automatic_replacement = False
tag_replacement_lower_treshold = 2
settings_sim_replications = 1
number_of_courses_per_time_step = All
number_of_tags_per_student_profile = 30
number_of_courses_picked_per_student_per_time_step = 1
number_of_courses_to_recommend = 1
print_output = False
delta = 2.5
sim_seed = thesis
number_of_students = 50
student_study_duration_min = 8
student_study_duration_max = 8
student_profile_content_tag_weight_min = 5
student_profile_content_tag_weight_max = 5
sim_clock = 0
sim_time = 40
or_courses_list_file = data\input\OR_courses_list.json
courses_obj_list_file = data\input\courses_obj_list_tags_2.5_7.5.obj
temp_tags_obj_list_file = data\output\temp_tags.obj
tagsfile = data\input\tags\stackoverflow_tags.csv
temp_students_with_tags = data\output\temp_students_with_tags.obj
output_students_obj_list_file = data\output\sim_all_students_with_tags.obj
output_courses_obj_file = data\output\sim_all_courses_with_tags.obj
previous_experiments_file = data\input\previous_experiments.json

[exp_00-0001]
```

Within the configuration file, it is possible to define experiments containing only those settings which differ from our default values, under an entry of the form *[experiment name]* (e.g., *[exp_00-0001]*). This experimental configuration file (see Listing 5.1) will adjust the default parameter values in order to run the desired experiments and replicates, and create output files with file names containing the experiment name and replication number.

We use these input variables as default settings for our initial simulation experiments and change them in further experimentation, which is described in more detail in Section 5.4.

4.3.2 Simulation Process

In this section we describe the different steps in the simulation process. A simplified overview of these steps is depicted in Figure 4.9. A detailed description of the process at each step follows throughout the remainder of this section.

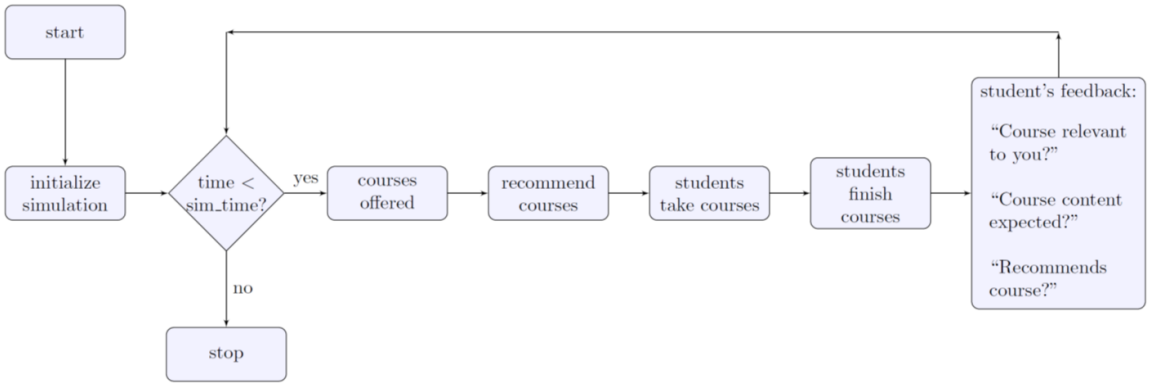


Figure 4.9. Simulation Overview

initialize simulation: Once the simulation begins, the program reads in all course data including corresponding content keyword tags from a data file. Next, the simulation generates the desired number of students (per the `number_of_students` input variable) and randomly assigns content keyword tags from all courses to the student profiles as "interest" tags. Using the information previously defined in the configuration file on the number of tags to assign per profile (`number_of_tags_per_student_profile`), as well as the range for their weights (`student_profile_content_tag_weight_min` and `student_profile_content_tag_weight_max`), the program distributes the weights within this range to the profile tags. This generates virtual student profiles with randomly

distributed interests in the offered courses. If desired, the program creates communities (`settings_communities_included`) and students join them according to the probability specified in the configuration file (`community_join_probability`). Communities may inherit a trend (`settings_trend_included`) in providing feedback about the course as defined by the first student who takes a course (e.g., the other students are more likely to like a course because the first student likes it), and this trend persists only for a predefined time (`trend_persistence_time`). For comparison purposes, we also generate a static course tag matrix that is necessary for the static recommender system. To obtain reproducibility, a seed (`sim_seed`) is set for the random numbers. Finally, the program stores the generated students and corresponding tags in an output file.

time < sim_time?: The program sets the overall time period for the simulation according to the `sim_time` input variable. This value defines the total number of loops in the simulation. Thus, after each loop (representing one quarter), the system checks if it has to run further loops. For the first loop, all students generated during the initialization phase enter the simulation, run through all following steps and can then either continue to study (i.e., enter the next loop) or leave the system. The program removes the departing students, archives their data, and generates new student profiles, as required.

courses offered: This comprises either all available courses or the number of randomly selected courses that is specified in the input settings (`number_of_courses_per_time_step`).

recommend courses: From the offered courses, courses that were previously taken by the student are excluded. The program makes course recommendations based on one of two options as defined in the input settings (e.g., `settings_recommendation_cos_sim`): (1) the program randomly recommends remaining courses to students without regard to the generated profile tags, or (2) the program compares the course tags for the remaining courses with the student profile tags in order to select courses with high similarity. The second approach can use either static or adaptively updated tags, for both the course tags and the student profile tags. The number of courses recommended to the student can be specified in the input settings (`number_of_courses_to_recommend`). During this step, the program adds the recommended course(s) to the student's profile.

students take courses: The number of courses each student takes during a quarter is also defined in the input settings (`number_of_courses_picked_per_student_per_time_step`).

During this step, the program randomly selects the course(s) from the list of recommended courses that was offered to each of the students and adds a record of courses taken by each student to his/her student profile.

students finish courses: Students can drop, fail or pass the course, and receive grades according to a grading distribution defined in the input settings (`distribution_grades`). During this step, the program generates course completion information for each student and adds this information to the student profiles.

student's feedback: "Recommends course?": The program simulates feedback on whether students liked or disliked a course in accordance with the feedback distribution defined in the input settings (`distribution_feedback`). The program then adds this information, whether course feedback was positive or negative, to the student profiles.

student's feedback: "Course content expected?": In this step, the program simulates feedback on whether the tags assigned to a course were appropriate or not. The program generates feedback based on one of three distribution options as defined in the input settings: (1) each student evaluates the tags independently, according to a predefined feedback distribution (`distribution_tag_feedback`); (2) each student evaluates the tags based on positive or negative trends, which are set by the first student's evaluation (`distribution_tag_feedback_trend_neg` and `distribution_tag_feedback_trend_pos`); and/or (3) students who are part of a community evaluate the tags in accordance with the evaluation by other students of the same community, again with a probability distribution defined in the input settings (`distribution_tag_feedback_trend_neg` and `distribution_tag_feedback_trend_pos`). In contrast to (2), only students of the same community follow the trend existing in this particular community. The program generates the feedback and then adds this information to the student profiles. The simulated feedback influences the weight of the evaluated tags by the predefined value of δ (`delta`).

student's feedback: "Course relevant to you?": The program simulates feedback on whether or not a course was relevant to the individual student following a predefined probability distribution (`distribution_tag_feedback_learner`), and the program adds this information, course relevant or not, to the student profiles.

stop: Once the predefined time frame for the simulation is complete, the program archives all remaining students, regardless of whether or not they completed their full learning path. The program stores all student and course information on the file system as simulation output.

Figure 4.10 shows how we represent students, courses and tags as objects (here, only a subset of the inner fields is shown) within the simulation programming software. The software allows us to define tags of different type (e.g., content, media type, or instructor) and to assign these tags either to the student profiles or to the courses. In this thesis, we only focus on course content as our tag type, but the program is ready for the addition of other tag types.

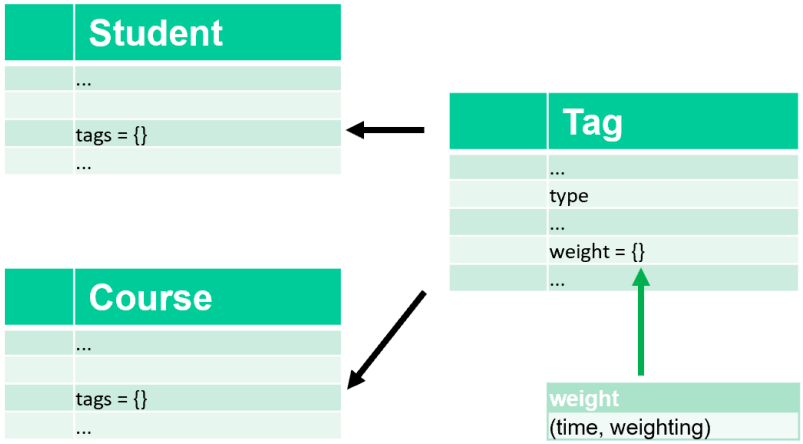


Figure 4.10. Simplified UML representation of students, courses and tags, as represented in the simulation.

4.3.3 Simulation Output

After each replication of an experiment, we obtain four data files. The first data file contains all the student profile information that was recorded during each loop of the simulation run (`output_students_obj_list_file`). This student data file contains information on which courses were recommended either by the static (without feedback-dependent adaptations) or dynamic (with feedback-dependent adaptations) system, which courses were taken, which grades were achieved, and what feedback in terms of course recommendations was given. The second data file contains all course information which includes information

about the course content tags with their respective weights over the course of the simulation (`output_courses_obj_list_file`). The third data file is a graphical representation of the courses and tags (see Figure 5.3). The fourth, and final, data file summarizes the statistics for each replication of the experiment.

4.3.4 Limitations of the Simulation

The simulation and course recommendation algorithm designed as part of this thesis is an initial prototype that allows the analysis of a number of parameters that we consider essential for our personalized learning platform. It is able to simulate student profiles by first randomly assigning interest tags to them, and subsequently updating student interest according to the courses a student took and the feedback the student gave following completion of the course. Furthermore, the system compares student profiles and tags assigned to all courses currently offered in the OR department at NPS, and analyzes a recommendation from this data as we described it in Section 4.2.2.

As we show in Listing 4.1, several parameters including numbers of simulated students and offered courses, overall time period, and probability distributions for grading, taking a course, and giving feedback on it can be set as input parameters in the simulation software. More importantly, the prototype is able to update both, the student profiles and the course tags, thereby taking student feedback into account. This dynamic analysis can be directly compared to a static approach, where the simulation uses the same parameters but does not update tags based upon student feedback, and it provides the possibility to switch between different modes how feedback distribution is expected (independent students, according to trends, or within communities).

However, there are some limitations that we believe need to be addressed in future work. Most importantly, the simulation data needs to be verified and validated, ideally by comparing it to empirical data from real students (Sargent 2013). As the cohort data we analyze in this thesis did not contain enough relevant information concerning interests and student feedback on course content, we randomly assign these to our generated profiles in the simulation. Still, this might not adequately represent the real situation, where a number of factors like previous experience, education goal and career ambition, as well as personal background will certainly effect the distribution of these tags in the student profiles. Sim-

ilarly, a variety of factors will affect students' feedback on a course. We start to address some of them by adding the option to include the trend- or community-based feedback distributions, but more factors like instructors, grading and overall success rate will need to be taken into account for a more realistic scenario. This also applies for course tags that are relevant for the recommender systems. Here, we focus on course content keyword tags and compare them to simulated student interests, but variables like instructor preferences or educational methods should certainly be integrated into a more advanced software prototype. Moreover, the recommender system we describe here does not consider student performance in previously completed courses when making future course recommendations.

In the next chapter, we present the results we obtain by applying this methodology. We describe the results obtained from the automatic tagging of course syllabi, the determination of course prerequisites, and discuss other information extracted from the student transcripts. We then show the results from our recommender algorithm prototype, including the simulated student feedback.

CHAPTER 5:
Results and Analysis

The current research aims to outline the initial steps to establish a prototype for an algorithm-driven, network science based, personalized adaptive learning platform. We analyze data from students and courses in the OR department at NPS with the aim of establishing an automated system that assigns useful metadata tags to the learning content as well as to student profiles. Our goal for the extracted metadata is to allow the set up of an automated individualized recommender system for course selection, adapted to each student's profile based on his/her feedback and learning progress.

In this chapter, we first describe the results we obtain from our analysis of automatically tagging the available NPS courses, as well as extracting information from the NPS catalog and real student transcripts. We discuss whether the analysis reveals useful keyword tagging candidates for course content, identifies course prerequisite relationships, and reveals any other information beneficial in the creation of our student profiles. We then show and analyze the simulation output from our recommender algorithm prototype, including the simulated student feedback.

5.1 Automatically Tagging Metadata from Course Syllabi

Tagging educational material with metadata that describes the content and other features of the respective material is a crucial requirement for an automated recommender system. We investigate the automated extraction of useful tags—describing the content of courses offered by the OR department at NPS—from the syllabi of these courses. The idea behind the extraction of tags from course syllabi is that we reasonably expect the highly relevant words for a course to be mentioned fairly frequently in the syllabi and thus, the most frequently used words from the syllabi could represent useful keyword-like tags to describe the course content.

Our analysis is based on the identification of the ten most common words in each syllabus. For privacy protection and to boost the results, we exclude personal information on instructors and stop words, as listed in Table A.2.

For reference, Table 5.1 shows the results of this analysis for the five most common words among five OA courses, and a full list comprising all courses is shown in Table A.1. For brevity, the table depicts the five most common words; however, in our prototype we utilize ten most common words, whenever ten content-related words were extracted.

Table 5.1. Five Most Common Words Extracted from OA Course Syllabi

course	**(common word, count)**
OA1600	('introduction', 9), ('programming', 9), ('analysis', 6), ('savage', 4), ('topic', 3)
OA2900	('thesis', 5), ('operations', 3), ('research', 3), ('opportunities', 3), ('pass/fail', 2)
OA3101	('probability', 5), ('statistics', 5), ('oa', 4), ('homework', 4), ('quarter', 3)
OA3102	('statistical', 6), ('statistics', 3), ('gl', 3), ('00', 3), ('0', 3)
OA3103	('data', 7), ('analysis', 6), ('glasgow', 5), ('oa3103', 4), ('set', 3)

As can be seen from the five examples in Table 5.1, some of the identified tags are appropriate words to describe the course content: for example "programming" and "analysis" for OA1600 (Introduction of Operations Analysis I). However, others like "topic," "thesis," or "homework" relate to common syllabi course organization topics rather than to specific course content. Moreover, others describe permanent record type information, such as meeting time and location.

The quantity of useful tags varies greatly between each of the courses, which is most likely due to a lack of a common or standardized format for syllabi. Given the inconsistent format of the course syllabi, we find that our fully automated tool is unable to consistently provide content keyword-like tags across all courses.

To improve the robustness of our automated metadata tool, we consider the extraction of the most commonly occurring words from all educational materials used in each of the courses. We explore this concept for the course MA4404, Structure and Analysis of Complex Networks, taught in the Department of Applied Mathematics at NPS. We choose this course because all of the educational material is publicly available (Gera 2018). We apply a method similar to our syllabi extraction process to extract the most frequent words over the whole course. To extract content keyword tags for each lesson or module separately, we complement this effort with a Term Frequency-Inverse Document Frequency (TF-IDF) algorithm (Ramos 2003), which identifies the most relevant words for each document in the

content collection (i.e., a lesson or chapter).

Table 5.2 shows the results of this analysis for four PPTX lesson files, and a full list comprising all PPTX lesson files is shown in Table A.3.

Table 5.2. Most Common Words Extracted from MA4404 Course Materials

topic	**(common word, count)**
01-RealComplexNetworks	('networks', 94), ('network', 29), ('source', 21), ('internet', 16), ('“', 12), ('”', 12), ('nodes', 11), ('social', 11), ('newman', 11), ('information', 10), ('web', 10), ('different', 10), ('data', 8), ('graphs', 7), ('edges', 7)
02-NetworkScienceOverview1	('networks', 23), ('network', 15), ('graph', 13), ('graphs', 13), ('science', 12), ('theory', 12), ('complex', 11), ('eulerian', 9), ('random', 9), ('social', 9), ('”', 9), ('“', 8), ('königsberg', 8), ('small', 7), ('cited', 6)
03-NetworkScienceOverview2	('networks', 32), ('model', 29), ('random', 22), ('graphs', 19), ('world', 19), ('network', 17), ('nodes', 16), ('small', 15), ('path', 14), ('average', 12), ('source', 12), ('www', 11), ('degree', 10), ('scale', 9), ('free', 9)
05-NetworkScienceOverview2	('networks', 36), ('model', 29), ('random', 24), ('graphs', 21), ('world', 20), ('network', 16), ('nodes', 16), ('small', 15), ('degree', 14), ('path', 13), ('average', 12), ('scale', 11), ('free', 11), ('complex', 10), ('distribution', 10)

This method reveals tags that accurately describe course, chapter, or lesson content. The automatic extraction of tags for the whole course and each separate lesson at the same time has the additional advantage that course content could be offered in a finer granularity to students, which is one of our overall goals for the CHUNK Learning system (see Section 2.1).

While this approach looks very promising, we do not have access to the entire collection

of educational material for all courses offered in the OR department. We therefore use the keyword tags that we obtain from our syllabi extraction analysis for our initial experiments described here, keeping in mind that automated methods to tag course content collections should be included in future work (see Section 6.1.1.1).

5.2 Determination of Course Prerequisites

It is crucial for our recommender system to have access to data that captures prerequisites, and therefore, we aim to assign tags describing prerequisites to all courses. In order to automatically identify tags describing course prerequisites, we compare two different approaches: (1) reading prerequisite information from the course catalog and (2) inferring prerequisite relationships from student transcript data. We summarize a portion of the results from both approaches in Table 5.3. Table B.1 shows the complete list of courses.

Table 5.3. Derived Course Prerequisite Relationships

	Method 1		Method 2	
Course ID	**Prerequisite Description from NPS Catalog**	**Extracted NPS Courses**	**Prerequisite Derived from Student Transcripts**	**Number of Students**
OA1600	None	None		62
OA4105	OA3103 or consent of the instructor	OA3103		19
OA4201	OA3201	OA3201	OA3201	97
OA4202	OA3201	OA3201	OA3201	97
OA4203	OA3201	OA3201	OA3301, OA3102, OA3201, OA3101, MA3042, OA4201, OA2900	18
OA4301			OA3301, OA3201	26
OA4333	OA3302	OA3302	OA3301, OA3103, OA3201, OA3102, OA3101, MA3042, OA4202	35
OA4401			MA3042, OA2900, OA3101	7
OA4402			Not in data set.	-
OA4406			Not in data set.	-
OA4408			OA3301, OA3103, OA3201, OA3102, OA3101, NW3230, OA4801, OA3302, OA4655, MA3042, OA4201, OA3900, OA3304, OA2900, OA3401, OA4202	3
OA4414	OA3413	OA3413	Not in data set.	-
OA4415			Not in data set.	-
OA4602	A course in basic probability and statistic theory and operational experience in military environments			69
OA4603				37
OA4604	OA4655 or consent of instructor	OA4655	OA3101	67
OA4607			OA3301, MA1025, OA3103, OA3102, OA3101, OA3201, OA4655, OA4801, OA3302, OA0810, MA3042, OA4201, OA3900, OA3304, OA1600, OA2900, OA4602, OA4202	2
OA4613	OA3201 or OS3007 or OA3611 or permission of instructor	OA3201, OS3007, OA3611	MA1025, OA3103, OA3102, OA3101, OA3201, OA4301, OA3302, MA3042, OA3610, OA4201, OA4106, OA2801, MA1118, OA3301, MA1113, MA1114, OA4333, OA1600, OA3501, OA2900, OA4202	1
OS3112	college algebra and OS3111	OS3111	IT1500, OA3103, OA3200, OA3102, OA3101, OA3201, OA4301, OA4801, OA3302, OA0810, MA3042, OA4203, OA4201, TS3002, OA2200, OA3301, TS4003, TS4001, OA4655, TS4002, MA1115, OA3602, OA3900, OA3304, OA2900, OA3401, OA4202	1

We now discuss the two approaches.

5.2.1 Method 1: Determination of Prerequisite Relationships from the NPS Course Catalog

In the first approach (Method 1 in Table 5.3), we analyze and extract course prerequisite information from the NPS course catalog. We automatically search the description and the prerequisites field for a pattern capturing different ways to define prerequisites using regular expressions, and extract the information following this pattern (see Column 2 in Table 5.3). This approach reveals different hits ranging from "none", meaning that no previous knowledge is required, to references to other NPS courses.

Since our aim is to establish a network of courses that are linked by directed edges which indicate prerequisite relationships, we focus on results referencing other NPS courses [2] and "none." As all NPS courses are described by a unique combination of 2 letters followed by 4 digits, we use regular expressions to search the results in column 2 for that pattern or the word "none" (see Column 3 in Table 5.3).

This approach is straightforward and provides reliable results for a portion of the courses. Still, it requires the data in the catalog to be formatted in a specific manner (containing a reference to an existing NPS course or labeled as "none"). However, it appears that prerequisite information is not standardized across all courses in the NPS course catalog, and therefore "no prerequisite" tags could possibly be assigned to courses where the prerequisite information was incorrectly interpreted.

5.2.2 Method 2: Inferring Prerequisite Relationships from Student Transcript Data

We investigate a second approach (Method 2 in Table 5.3), which is based on actual transcript data from student cohorts. The basic idea is that if course A is a prerequisite for course B, then all students in course B must have already completed course A (in a previous quarter).

Therefore, we analyze for all students in one particular course, what courses they have previously completed. If there are courses that were completed by every student attending

[2] Noteworthy, the analysis described here also includes prerequisite courses which are offered by other departments at NPS. The networks generated in this thesis consist exclusively of courses in the OR department. Thus, prerequisite courses from other departments do not show up in our network layers. Still, in an institution-wide approach, this would be resolved as all courses offered by that institution would be included in the multilayer network.

that particular course, we infer that these courses are prerequisites for the current course. Prerequisite courses identified according to this analysis are listed in Column 4 of Table 5.3 (expanded in Table B.1). Column 5 shows the number of students in the available data set that attended the courses we inferred as prerequisites.

Whereas this approach is able to assign prerequisite courses to many courses where the first approach failed, there are two major limitations that can lead to false positive or false negative results, respectively. The first limitation is present when the analyzed number of students in one course is very low, increasing the probability of obtaining a false positive. Extreme examples of this are courses like OA4613 or OS3112, which were attended by just one student out of the 97 students available for analysis. In this case, the number of courses taken by all students of this course—and thus showing up as prerequisite courses —is equal to every course that was previously taken by that particular student, leading to false positive results. This limitation could be circumvented by analyzing a much larger data set that ensures each course is taken by a high number of students, or by defining a minimum threshold for the number of students in each course.

The second major limitation may be caused by students that come from other learning institutions and get transferred credit. These validated courses, based on other institutions' credit, will not be recognized as prerequisite courses by our system, which so far only recognizes the code pattern for NPS courses. Moreover, these students will not have attended the prerequisite course at NPS. Therefore, the NPS prerequisite course will not have been taken by all students taking the follow-on course, thus leading to a false negative result. Again, this limitation could be circumvented, either by excluding students who are in their first quarter from the analysis, or by defining a shared prior-course threshold to identify the prerequisites. The current threshold is 100%, which then misses some of the course dependencies. Identifying a threshold empirically, and then using verification and validation, would minimize the risk of false positive or false negative results.

With empirically derived threshold limits, larger scale analysis of student transcripts, and careful interpretation of the results, we think that method 2 can successfully be used to identify prerequisite course relationships. This is reinforced by the observation that the two approaches described here provide overlapping results, whenever a comparison is possible (i.e., a high number of students are available for method 2, and results are obtained by both

approaches). With this in mind, method 2 might be a good option to complement the first method based on course catalog data to identify prerequisite courses.

5.3 Additional Information Extracted from the Student Transcript Data

In addition to prerequisite course relationships, we try to automatically extract additional information from the student transcript data. We find that this information is relevant for creating student profiles based on interests or learning goals, and evaluating students' feedback based on courses taken. We are able to use the data to automatically assign personal data like gender, member of military service, and grades to each student. An overview of the empirically-computed distributions for the student demographic information is given in Table 3.3. Furthermore, we obtain information on how many people from the same cohort attended a particular course together, and generate the network of shared courses between cohorts (see Figure 3.5).

Finally, we extract the grading distribution for each course. Figure 5.1 shows the grade distribution for all courses, where each color represents one course. For anonymity reasons, we omit the legend explaining what color represents which course.

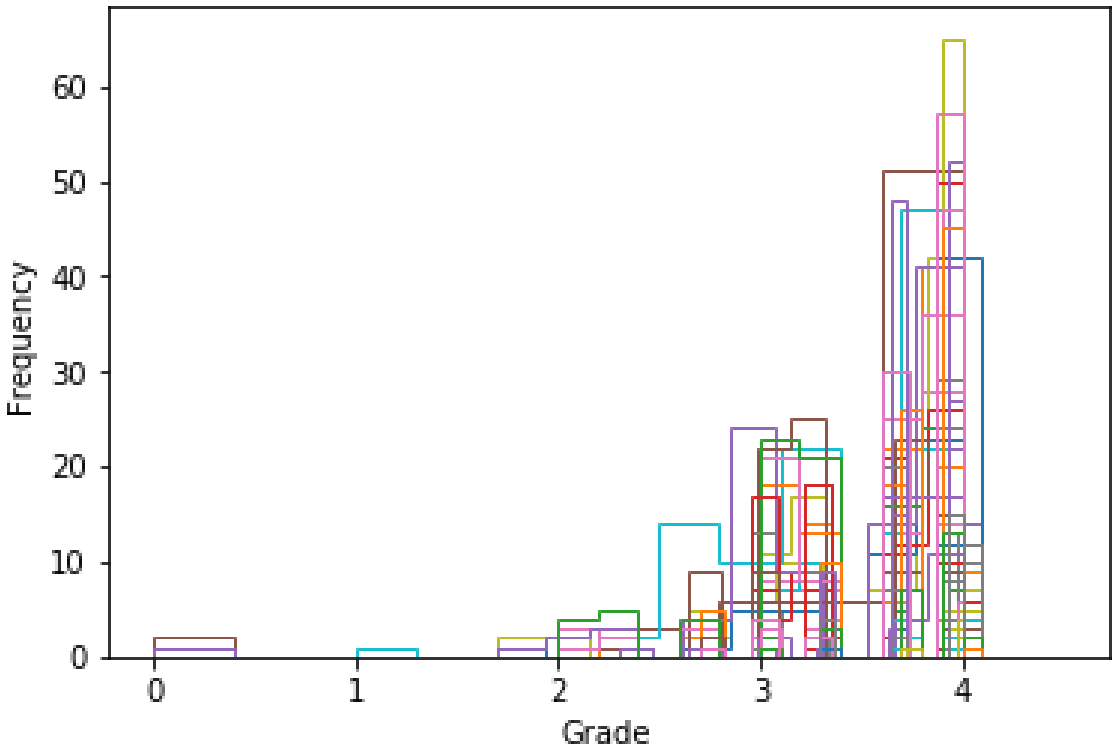


Figure 5.1. Cohort grade histogram derived from student transcript data over all courses.

As can be seen in Figure 5.1, for most courses, 4 (A) and 3 (B) are the prevalent grades. At NPS, an average score of 3.0 over all courses is required for graduation. Thus, grades above that threshold need to be achieved by most students in the majority of courses. The grading distribution might be of particular interest when modeling student feedback. We believe that courses where students receive predominantly good grades (i.e., a high amplitude at grades 3 and 4 in Figure 5.1) can reasonably be expected to receive more positive feedback from students than courses with a more moderate grade distribution, as students will perceive a course more positive if they are successful. Furthermore, from the grades distribution, one can evaluate the pass/fail rate, which again affects the statistics, tags for that course, and might give a hint to the difficulty level, all relevant information for the recommender system.

For the establishment of a prototype of an adaptive recommender system based on student

feedback, we are mostly interested in student interests, learning preferences or study goals. However, the data obtained for this thesis did not contain this type of information. Thus, we decide to simulate these parameters, as we describe in Section 5.4.

5.4 Simulation

In this section, we describe results obtained from our software prototype for a real-time adaptive personalized learning framework. Within our simulation, the prototype's recommender system suggests courses to students by matching course content (from tags) with student interests, both of which update during the course of study.

From the three simulated feedback questions offered to the student, we use two of them to update the student profile and course content tags: (1) Did the course content meet your expectations? and (2) Was the course content relevant to you?. A positive or negative answer to the first question will increase or decrease the weight of all course tags, respectively (see Figure 4.6). This assumes that the student took the course because the course content (via tags) matched the student's interest from his/her stored profile. Similarly, a positive or negative answer to the second question will increase or decrease the tags associated with the student's profile. Note that the program only updates the profile tags that match the tags for this particular course. All other (non-matching) tags in the student's profile remain unchanged. During each loop of the simulation (e.g., one academic quarter), we simulate student feedback and adjust tag weights for courses and profiles by adding or subtracting the predefined constant δ, according to the process outlined in Figure 4.6.

In the following sections, we analyze how the incorporation of student feedback within the simulation changes the tag weights in both student profiles and course content tags temporally, and what effect this process has on courses that are recommended to a student and the order in which courses are recommended, in terms of course sequencing. We compare this dynamic recommendation system with a static system, in which all tags retain their original weight throughout the simulation. To this end, we calculate a recommender score as a measure of how well a recommended course matches the interests of a student which takes into account the number of matching tags as well as their weight (see Section 4.2.2). Then, we systematically change some input variables in the software to verify that our simulation is performing these calculations correctly. Lastly, we use an experimental design approach

to analyze the overall effectiveness of our recommender system. Specifically, we look at how a stronger or weaker implementation of the feedback as well as different feedback distributions affects course recommendations.

5.4.1 Effects of Simulated Student Feedback based on Tag Weights and Course Recommendations

For the initial experiments, we use the default settings described in Section 4.3. The assumption of the model is that all students provide feedback independently, with a default distribution of 70% providing positive feedback (on average) and 30% providing negative feedback (on average) on either question, and we set δ to 2.5, thus allowing student feedback to have a clearly measurable effect on the tag weights. These settings are used for all experiments, unless indicated otherwise. A complete list of all parameters and default values for this experiment can be found in Listing 4.1 and a discussion of how these parameters are used within the simulation in Section 4.3.2.

5.4.1.1 Updating Student Profile Tag Weights according to Student Feedback

First, we analyze if and how implementation of student feedback affects the weight of student profile tags. Figure 5.2 shows the temporal evolution of one student's profile tags where we represent each tag with a different color and each dot represents the new tag weight (only those tags which changed weights are shown at each new time step). Tags that do not match with recommended courses remain constant throughout the simulation, and thus do not show up in the plot at all.

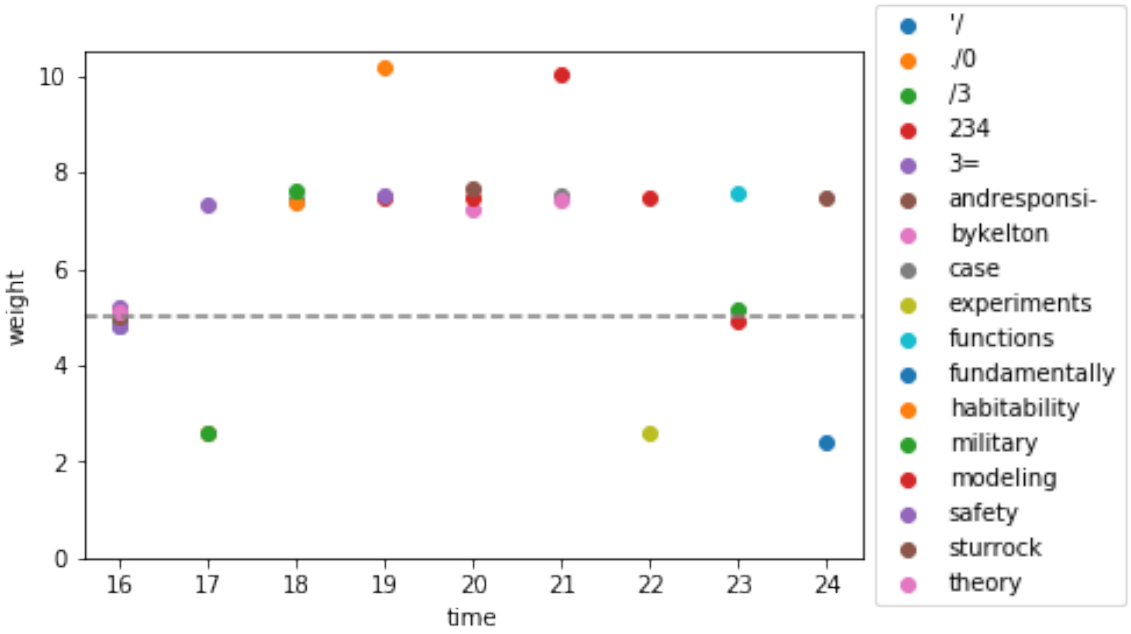


Figure 5.2. **Change in student profile tags over time.** An example of one simulated student's eight term period of study. Each tag is represented by a different color and only those tags which changed weights are shown at each new time step. A jitter is added to visualize overlapping dots.

From Figure 5.2, it is clear that the interest tags within the student profile change in our personalized learning prototype. The student shown here studied 8 quarters, beginning his/her course of study during simulation time step 16 and finishing during simulation time step 23. In this example, out of the student's 30 interest tags, 17 of these interests show up in the plot, indicating that the student took courses with tags that matched these interests. Following completion of each course, the student provides feedback and the program reevaluates the relevance of these tags, leading to adjustments in tag weight. When the student begins his/her period of study, all student interest tags begin with a default weight of 5, as defined in our configuration file. Throughout the period of study, we see that the majority of tags are increased to a weight of 7.5 and some increase to 10, while others are decreased to a less relevant weight of 2.5. This is expected due to the 70% prevalence of positive feedback and a δ of 2.5.

5.4.1.2 Updating Course Content Tag Weights according to Student Feedback

Next, we analyze if and how the implementation of student feedback affects the weight of course content tags. Figure 5.3 shows the temporal evolution of course content tags for all courses within the prototype.

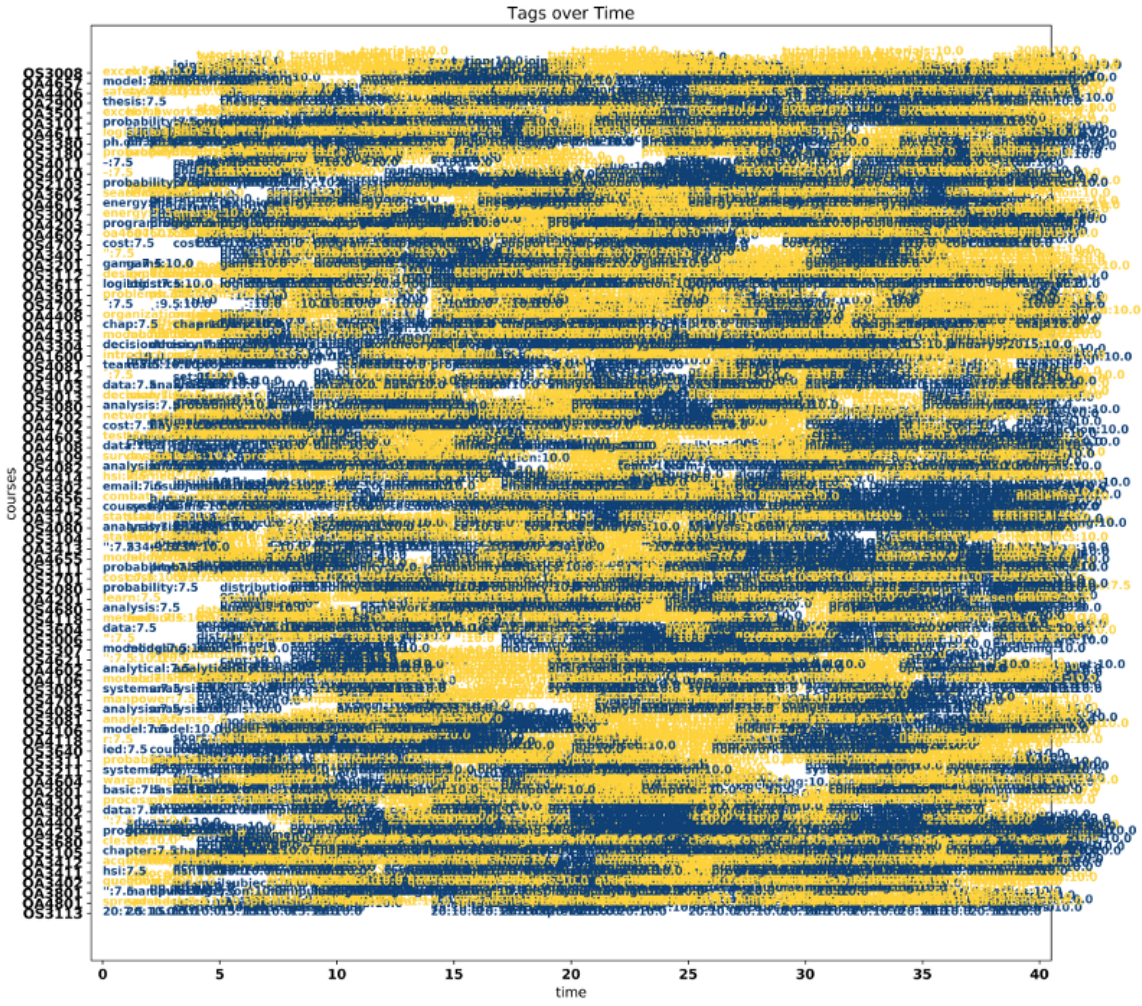


Figure 5.3. **Change in course content tags over time.** For all offered courses, each course tag is printed on the row of the corresponding course every time a change in tag weight occurs. The alternating row colors of blue and yellow are used to distinguish between the different courses. The plot does not indicate if course weights were increased or decreased.

The congestion in Figure 5.3 indicates that content tag weights undergo frequent changes throughout our simulation. In contrast to our method of updating interest tags in the student

profiles, where updates occur for only matching tags, all course content tags are updated according to student feedback. Furthermore, course content tags update throughout the entire simulation (here 40 time steps), whereas student profile tags only update during the respective student's period of study (i.e., up to 8 quarters per default). Figure 5.3 allows us to simultaneously see that course content tag weights are changing for all courses, but does not depict whether a single tag weight was increased or decreased, and to what extent.

Figure 5.4 shows a plot similar to Figure 5.2 where we represented changes in the weights of student profile interest tags. In Figure 5.4, each color represents one course content tag with the initial (default) weight ranging from 3 to 7.5. Each dot indicates a change in tag weight for that particular tag. If a tag repeatedly appears at 10 or 1, this indicates that the respective tag was positively or negatively evaluated, but retains its previous weight of 10 or 1, respectively.

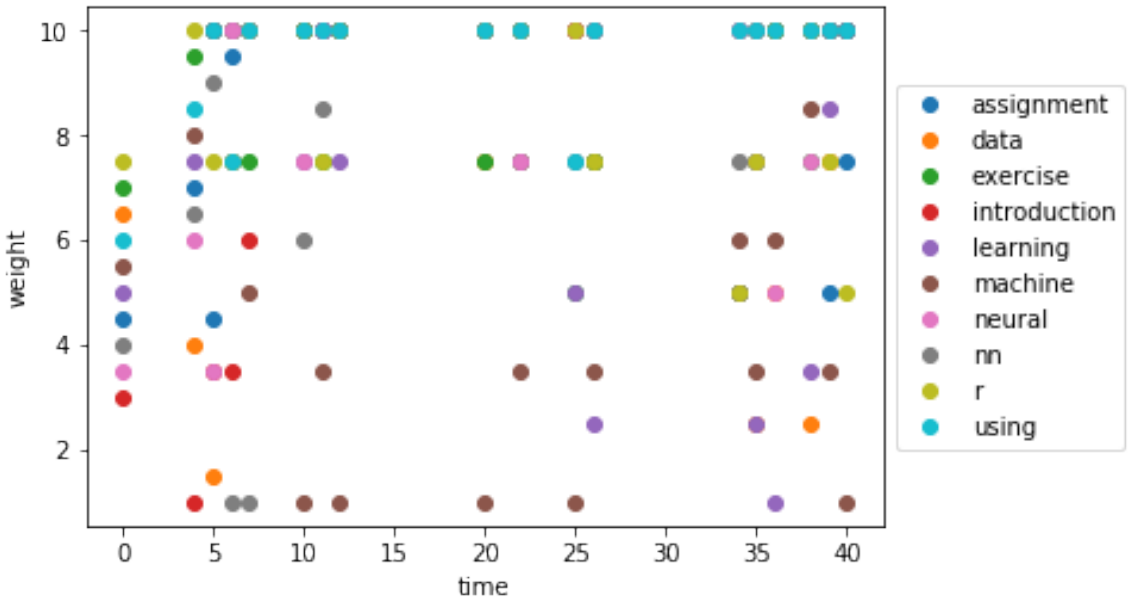


Figure 5.4. **Quantitative changes in course content tags over time for a single course.** An example of tag weight changes for course OA4118. The weights of the content tags are represented by colored dots. Every time students take the course and provide feedback, the tag weights are updated. Tags that have reached the maximal (10) or minimal (1) value retain these weights.

Again, the majority of tags increase in weight due to the feedback distribution of 70%

positive versus 30% negative. Comparing Figure 5.4 to Figure 5.2, we find that course tags change more frequently than student tags. This is expected, since (a) all course tags are updated, not just those which match student profiles, (b) more than one student can take and evaluate a course at a time, and (c) course content tags persist over the whole simulation period compared to the student's period of study which only spans 8 time steps.

Therefore, we conclude that our software prototype for an adaptive personalized learning framework is able to update tag weights according to student feedback, and that changes in tag weights occur for both student profiles and course content tags in accordance with our chosen experimental default settings.

5.4.1.3 Impact of Student Feedback on the Course Recommender System

From an application perspective, we believe that student feedback should impact both course recommendations and the sequencing of courses. Therefore, it is important to see if, within our simulation prototype, student feedback impacts the individual learning paths, and if the tag updating processes previously described impacts the recommender score, which we use as a metric for the matching quality (see Section 4.2.2). This section demonstrates how these two variables, course recommendations and course sequencing, are affected using a dynamic recommender system compared to a simple static course recommendation system.

We first analyze the effects of incorporating student feedback on the recommended learning path. For this, we compare the sequencing of recommended courses suggested by our dynamic system (tag weights are updated according to feedback) with a static model (tag weights remain constant throughout the simulation).

Figure 5.5 shows examples of the recommended learning paths provided by the static versus the dynamic model. We present these for two randomly selected simulated students. The course paths shown in Figure 5.5(a) represent the learning paths recommended by the static model (blue) and our dynamic system (yellow) for one randomly selected student from the initial group of 50 simulated students. The course paths shown in Figure 5.5(b) are the recommended learning paths for one randomly selected student who began his/her period of study at a later point in time, where course content tag weights for the student's first quarter may have already changed (by feedback from earlier students).

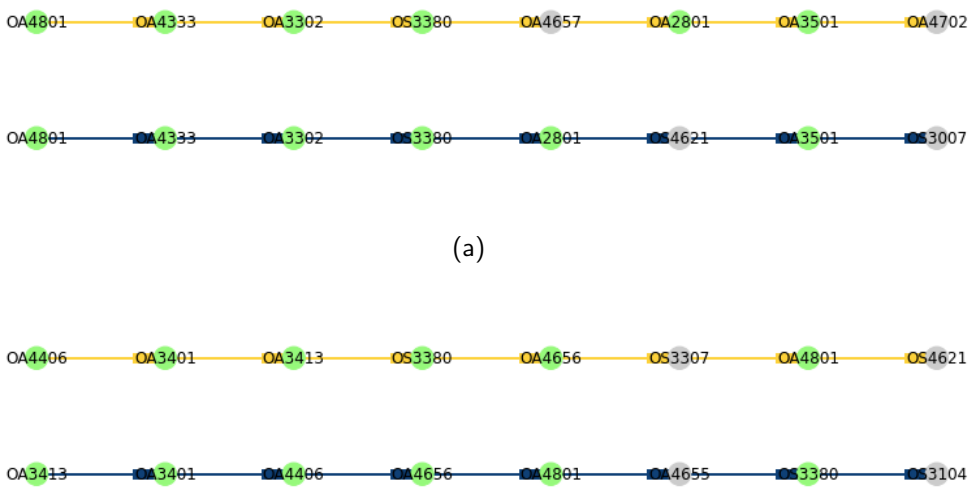


Figure 5.5. **Comparison of individual learning paths by a static versus dynamic recommender system.** Examples of recommended learning paths for a student beginning his/her period of study (a) at the beginning or (b) at a later time of the simulation. The upper (yellow) path represents the course recommendations resulting from the dynamic model, and the lower (blue) path is generated by the static model. Overlapping courses from each recommender system are marked in green, independent of their time of recommendation.

According to Figure 5.5, use of adaptive tags in the dynamic model indeed affects course recommendations to the same student since different course paths appear in the dynamic versus static system for both students shown. Interestingly, at the start of the simulation (Figure 5.5a(a)), we observe a significant amount of overlap between the learning paths, especially in the first few time steps, whereas for the student who began his/her period of study at later point in the simulation (Figure 5.5a(b)), this overlap does not exist. This can be explained by the fact that when the simulation begins the course content tag weights are the same for both the static and dynamic models; however, as the simulation progresses these weights stay the same in the static model and change in the dynamic model.

For a quantitative analysis, we calculate the ratio of courses that are not recommended by both systems ("Deviance of course recommendations"), and the ratio of courses that are recommended at a different time point during the learning path ("Deviance of course order")

among all courses which overlap for each simulated student. Furthermore, we calculate a recommender score (Section 4.2.2) for both, the dynamic and the static system, which serves as a measure how well the recommended courses fit the student profiles. Results are shown in Table 5.4.

Table 5.4. Quantitative Analysis of Dynamic Versus Static Recommender System

Output Parameters	**Mean**	**95% CI**
Deviance of Course Recommendations	0.3250	(0.3072, 0.3428)
Deviance of Course Sequencing (among shared courses)	0.6978	(0.6696, 0.7259)
Average Dynamic Recommender Score	0.1422	(0.1392, 0.1451)
Average Static Recommender Score	0.1297	(0.1272, 0.1322)

With the chosen default settings (see Listing 4.1), the course recommendations generated by the static and dynamic models overlap for approximately 2/3 of the courses. Implementation of student feedback to update course content and student profile tags thus leads to course recommendations that would not have been suggested by a static system for approximately 1/3 of the courses a student takes. With regard to the recommender score, this difference in course recommendations leads to a higher match between course content and student interest in our analysis, indicating that a dynamic recommender system can indeed improve the identification of appropriate learning content in accordance with our default parameter settings.

Out of the overlapping courses, about 2/3 are suggested at a different time point in our analysis. However, our analysis is only based on course content of each course independent of prior knowledge and prerequisite relationships. Implementation of the latter in future work would most likely change this result (see Section 6.2).

All in all, the results demonstrate that our methodology is able to simulate student feedback and use the responses to update both, student profile and course content tags, which—at least under the default settings—leads to an improvement in the average recommender score. In the next section, we modify some of the input parameters to a means to verify the functionality of our software prototype.

5.5 Verification of the Software Prototype

In order to verify that our dynamic recommender model is performing properly, we create a series of test cases to systematically modify some of the input parameters related to the tag update process. First, we set δ to 0, which should eliminate all effects of feedback. Then we allow positive or only negative feedback, which should exclusively increase or decrease tag weights, respectively. The results of these verification experiments are described below.

5.5.1 Test Case I: No Change in Tag Weights ($\delta = 0$)

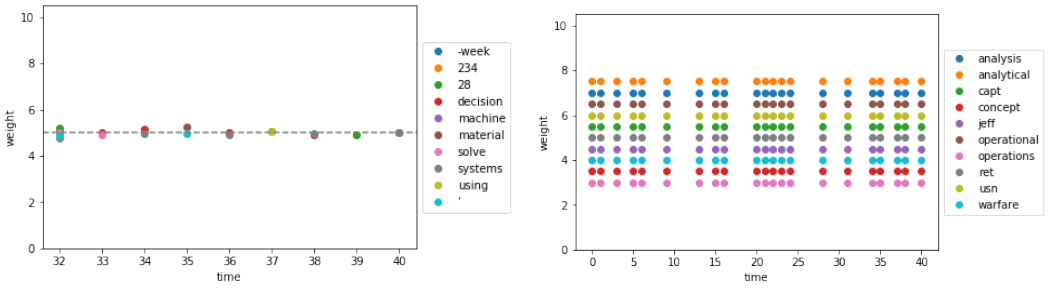
As described in Chapter 4 in more detail, δ is the constant in our system, by which tag weights are increased or decreased upon positive or negative feedback from the students. The value of delta determines the amount of impact the feedback has on the increase or decrease of tag weights. The higher the value of delta, the greater the impact of student feedback.

The first test case we verify is whether a feedback parameter value of `delta=0` results in no change to the weighting of the tags. The experiment corresponding to this test case is shown in Listing 5.1.

Listing 5.1: Test Case I: No Change in Tag Weights

```
[exp_00-0015]
delta = 0
```

In this case, the software prototype still simulates student feedback, but this feedback should not lead to changes in tag weights. To confirm this result, we run the simulation software with the same default parameters except for the value of δ. Figure 5.6 temporally depicts tag weights for the student profile tags of one randomly picked student (a) and for the course content tags of one randomly picked course (b).



(a) (b)

Figure 5.6. **Change in tag weights over time with $\delta = 0$.** (a) An example of one randomly selected simulated student. Each student profile tag is represented by a different color and only those tags which changed weights are shown at each new time step. A jitter is added to visualize overlapping dots. (b) An example of content tags weights for one randomly selected course. The weights of the content tags are represented by colored dots. Every time students take the course and provide feedback, the tag weights are updated.

As expected with $\delta = 0$, the student profile tags and the course content tags remain at the same level throughout the entire simulation. In Figure 5.6a, the slight deviations seen around a weight of 5 are due to the jitter that we randomly add to allow for visualization of overlapping dots. The appearance of dots during a time step indicates that students provided feedback on the respective tag. Nevertheless, this feedback does not affect the updated tag weight as δ is set to zero.

We also conduct a quantitative analysis of the results from the dynamic and static recommender systems when $\delta = 0$. Results are shown in Table 5.5.

Table 5.5. Quantitative Analysis of Recommender Systems ($\delta = 0$)

Output Parameters	**Mean**	**95% CI**
Deviance of Course Recommendations	0.0000	(0.0000, 0.0000)
Deviance of Course Sequencing (among shared courses)	0.0000	(0.0000, 0.0000)
Average Dynamic Recommender Score	0.1330	(0.1303, 0.1357)
Average Static Recommender Score	0.1330	(0.1303, 0.1357)

As expected with $\delta = 0$, no differences can be observed between the dynamic and the static recommender systems. These results verify that even though our software prototype

simulates student feedback, this feedback does not have any effect on the recommended learning path.

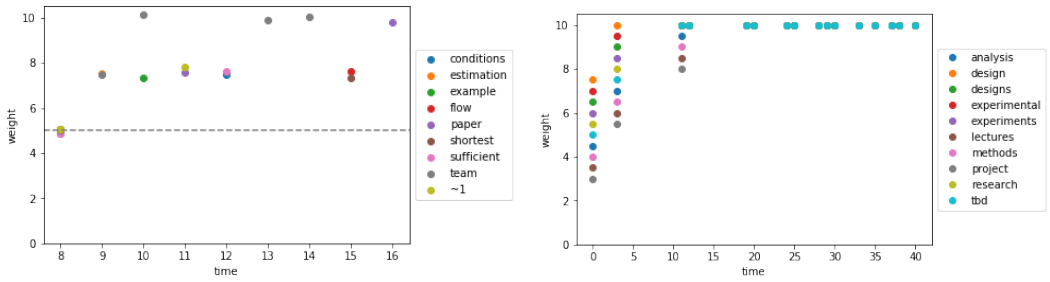
5.5.2 Test Case II: 100% Positive Feedback

The second test case we verify is whether feedback distribution parameter values of 100% positive feedback and 0% negative feedback, for both the student profile tags and course content tags, results in only positive increases to the weighting of the tags. The experiment corresponding to this test case is shown in Listing 5.2.

Listing 5.2: Test Case II: Only Positive Feedback

```
[exp_00-0016]
distribution_tag_feedback = [1.0, 0.0, 0.0]
distribution_tag_feedback_learner = [1.0, 0.0, 0.0]
```

Figure 5.7 temporally depicts the resulting tag weights, again for one randomly selected student (a) and for one randomly selected course (b).



(a) (b)

Figure 5.7. **Change in tag weights over time with 100% positive feedback distributions.** Each student profile tag or course content tag is represented by a different color and only those tags which changed weights are shown at each new time step. Tags that have reached the maximal (10) value retain this weight. (a) An example of one randomly selected simulated student. A jitter is added to visualize overlapping dots. (b) An example of content tags weights for one randomly selected course. Every time students take the course and provide feedback, the tag weights are updated.

As expected with feedback distribution parameter values of 100% positive feedback and

0% negative feedback, all tag weights exclusively increase over time and all course content tags reach the maximal weight (10) after several simulation time steps. We summarize the quantitative effects of these distributions on the static and dynamic recommender systems in Table 5.6.

Table 5.6. Quantitative Analysis of Recommender Systems (100% Positive Feedback)

Output Parameters	**Mean**	**95% CI**
Deviance of Course Recommendations	0.2935	(0.2764, 0.3106)
Deviance of Course Sequencing (among shared courses)	0.6876	(0.6601, 0.7151)
Average Dynamic Recommender Score	0.1472	(0.1434, 0.1511)
Average Static Recommender Score	0.1330	(0.1303, 0.1357)

Exclusively positive feedback and the resulting increased tag weights for the dynamic recommender system lead to differences in course recommendations as compared to course recommendations from the static recommender system. Interestingly, the extent of deviance and average recommender scores are comparable to the quantitative analysis obtained using the default settings summarized in Table 5.4.

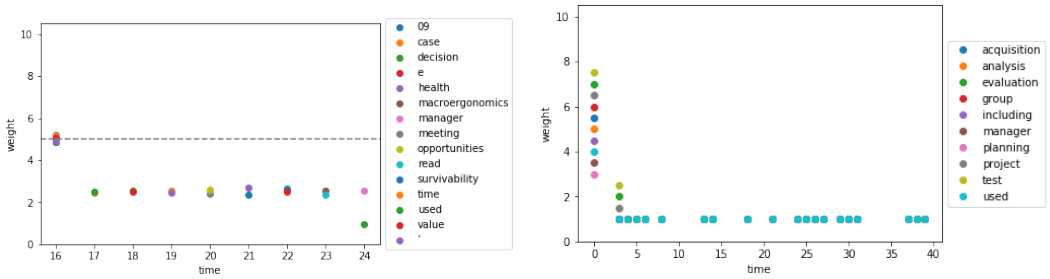
5.5.3 Test Case III: 100% Negative Feedback

The third test case we verify is whether feedback distribution parameter values of 0% positive feedback and 100% negative feedback, for both the student profile tags and course content tags, results in exclusively negative decreases to the weighting of the tags. The experiment corresponding to this test case is shown in Listing 5.3.

Listing 5.3: Test Case III: Only Negative Feedback

```
[exp_00-0017]
distribution_tag_feedback = [0.0, 1.0, 0.0]
distribution_tag_feedback_learner = [0.0, 1.0, 0.0]
```

Figure 5.8 again depicts the resulting tag weights over time for one randomly selected student (a) and for one randomly selected course (b).



(a) (b)

Figure 5.8. **Change in tag weights over time with 100% negative feedback distributions.** Each tag is represented by a different color and only tags which changed weights are shown at each new time step. Tags that have reached the minimal (1) value retain this weight. (a) An example of one randomly selected simulated student is shown. A jitter is added to visualize overlapping dots. (b) An example of content tag weights of one randomly selected course. Every time students take the course and provide feedback, the tag weights are updated.

As expected with distribution parameter values of 0% positive feedback and 100% negative feedback, all tag weights exclusively decrease over time and all course content tags reach the minimal weight (1) after several simulation time steps. We again summarize the quantitative effects of these distributions on the static and dynamic recommender systems in Table 5.7.

Table 5.7. Quantitative Analysis of Recommender Systems (100% Negative Feedback)

Output Parameters	**Mean**	**95% CI**
Deviance of Course Recommendations	0.2825	(0.2622, 0.3028)
Deviance of Course Sequencing (among shared courses)	0.6727	(0.6445, 0.7010)
Average Dynamic Recommender Score	0.1326	(0.1305, 0.1348)
Average Static Recommender Score	0.1327	(0.1299, 0.1355)

Exclusively negative feedback and the resulting decrease in tag weights still leads to differences in course recommendations between the dynamic and static recommender systems. However, in contrast to the previous run, the average recommender score for the dynamic model does not show improvement over the static system.

5.5.4 Modification of Several Parameters by Design of Experiment

So far, in order to verify that our dynamic recommender model is performing properly, we modify one factor at a time (OFAT analysis). We now aim to explore the effect of several variables in our software prototype on the quality of course recommendations as measured by the recommender score. Of particular interest are the following variables:

(1) `delta`, in order to assess magnitude of impact on the tag updating process;
(2) `distribution_tag_feedback`, in order to assess the importance of assigning fitting tags that are likely to be positively evaluated;
(3) `distribution_tag_feedback_learner`, as optimizing this is a major goal of the project;
(4) `number_of_tags_per_student_profile`, to assess a minimum number required for the recommender system to find reliable matches; and
(5) `number_of_students`, to check if a minimum number of students is required for the feedback to have an effect on the recommender score.

To explore the impact of these five variables on our recommender system all at the same time, we employ an experimental design approach with good space filling properties. We desire a design which covers the full variable design space and minimizes the number of necessary experimental runs. To this end, we choose a Nearly Orthogonal Latin Hypercube (NOLH) design, since this provides good space filling properties and will allow us to see interactions between the variables. We use a Microsoft Excel template provided by the Simulation, Experiments, & Efficient Designs (SEED) Center for Data Farming at NPS to generate the experimental design (Sanchez 2011). In accordance with this design, we run the simulation software using the 17 different design points (i.e., combination of parameter settings), with two replications of each design point for a total of 34 simulation runs. We write a program in Python, which automatically converts the experimental design points from the Excel spreadsheet into the the experiment entries needed by our simulation software configuration file (Listing 4.3.1). The settings for these 17 experiments are in the appendix, Listing C.1.

After running our experiments, we use JMP® Pro 13.1.0 statistical software to explore the simulation output (see Section 4.3.3) and generate the following figures. Figure 5.9 shows the correlation matrix as well as a bivariate scatterplot matrix revealing the pairwise

distribution of parameter values across the space of the experimental design.

Correlations

	delta	course feedback	learner feedback	# tags profile	# students
delta	1.0000	0.0030	0.0059	-0.0006	-0.0126
course feedback	0.0030	1.0000	0.0022	-0.0051	0.0051
learner feedback	0.0059	0.0022	1.0000	0.0024	0.0044
# tags profile	-0.0006	-0.0051	0.0024	1.0000	0.0032
# students	-0.0126	0.0051	0.0044	0.0032	1.0000

Scatterplot Matrix

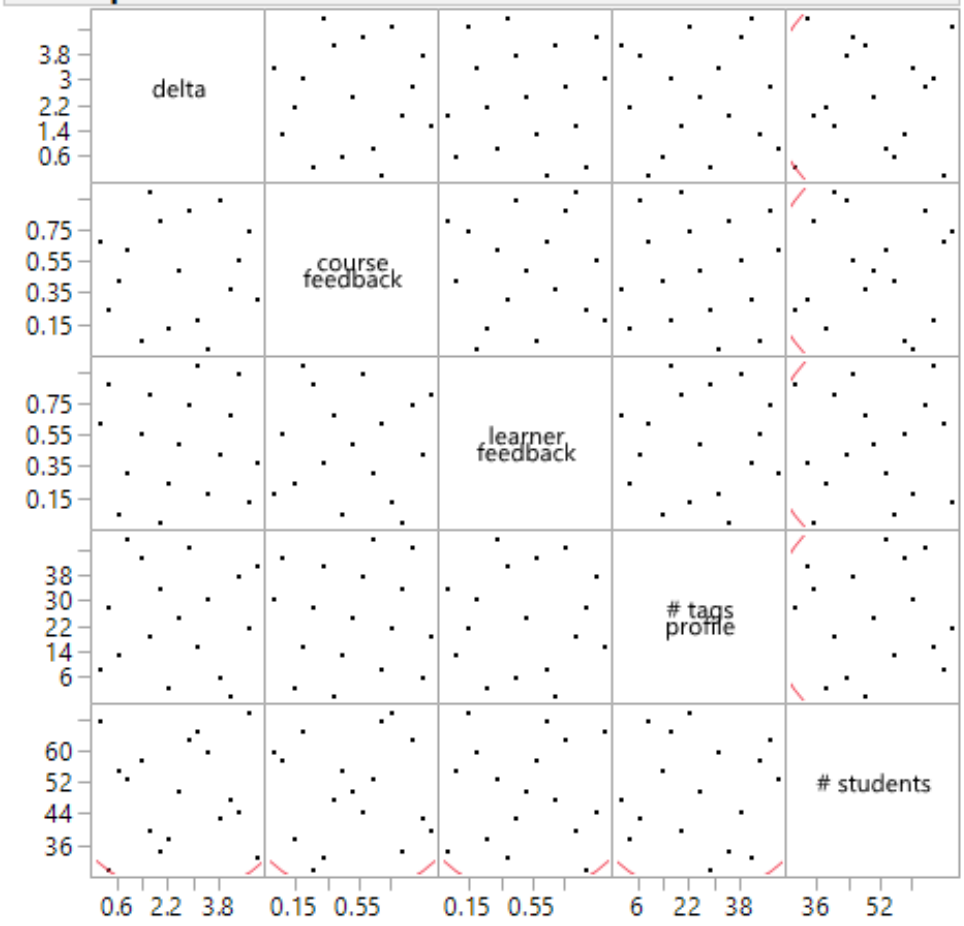


Figure 5.9. **Correlation and scatterplot matrices of the designed experiment.** The NOLH design is chosen to explore the following five variables: delta, feedback distribution on course tags, feedback distribution on profile tags, number of profile tags per student, and number of students. The values close to zero in the correlation matrix indicate that our parameters have little-to-no correlations (and thus are nearly orthogonal). In each of the bivariate scatterplots, each dot corresponds to a combination of parameter settings.

As can be seen from the correlation matrix, the design is almost perfectly orthogonal, with all pairwise correlations having values extremely close to zero indicating that our parameter values have little-to-no correlation with each other. The scatterplot matrix confirms that the parameter settings are chosen in a way that ensures even distribution over the design space.

We then conduct our quantitative analysis, computing values for the following four quantitative response variables across all 17 design points:

(1) Deviance of recommended courses (independent of their order);
(2) Deviance of sequencing (among all shared courses);
(3) Average recommender score in the dynamic recommender system; and
(4) Average recommender score in the static system.

Figure 5.10 gives an overview of how the five independent variables and the four quantitative response variables are related to each other.

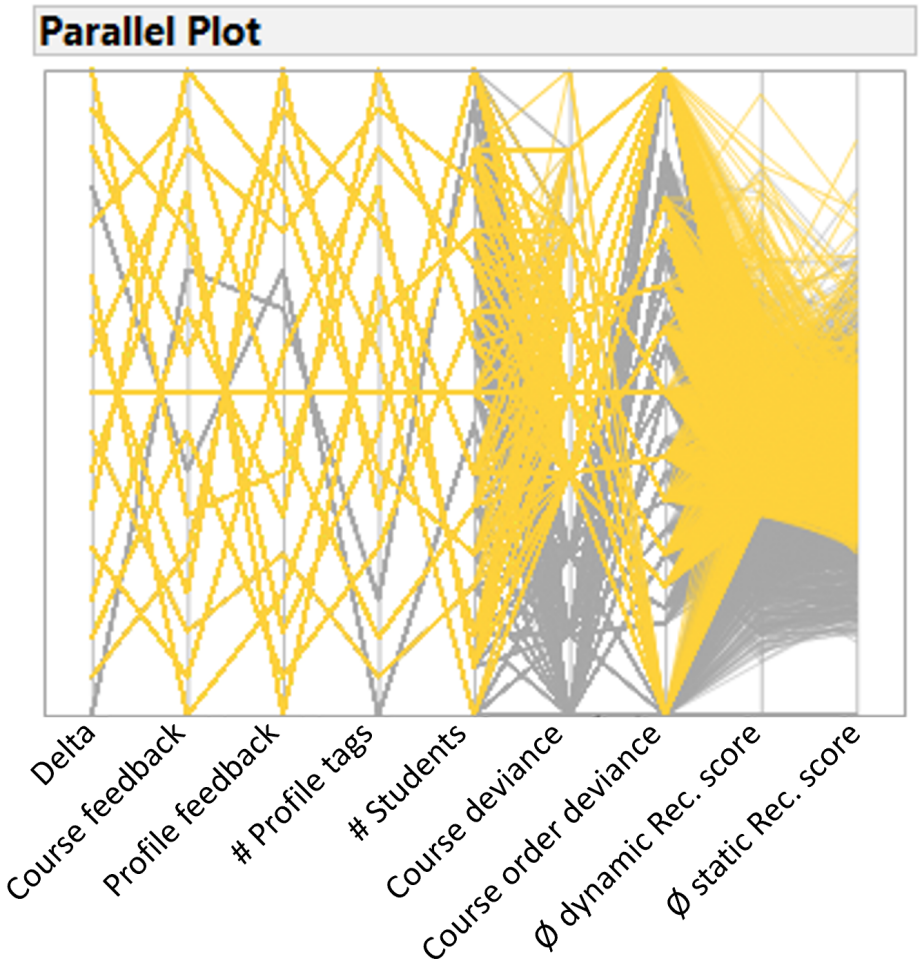


Figure 5.10. **Interaction of input variables and readout parameters in the NOLH experiment.** Data for each individual simulated student are shown for the five input variables: delta, feedback distribution on course tags ("Course feedback"), feedback distribution on profile tags ("Profile feedback"), the number of tags per profile, and the number of students per quarter, as well as for the four quantitative response variables: deviance of recommended courses ("Course deviance"), deviance of recommendation sequencing among shared courses ("Course order deviance"), the average recommender score in the dynamic, and the average recommender score in the static system. Data appear highlighted in yellow if course deviance is above average and the dynamic recommender score is higher than the average static recommender score. For additional details on the distributions of the simulation output metrics see Figure C.1.

As can be seen in Figure 5.10, we observe no clear relationship between the five input

parameters and the sequencing of courses. Generally, the dynamic system is able to achieve higher recommender scores than the static system, and a high recommender score in the dynamic system seems to be associated with a higher deviance of recommended courses. Furthermore, a recommender score and course deviance above average can only be achieved if $\delta > 0$, and if the number of student profile tags does not fall below a certain threshold.

As the number of tags in a student profile is of relevance for establishing a learning platform in the future, we analyze this observation in more detail (see Figure 5.11).

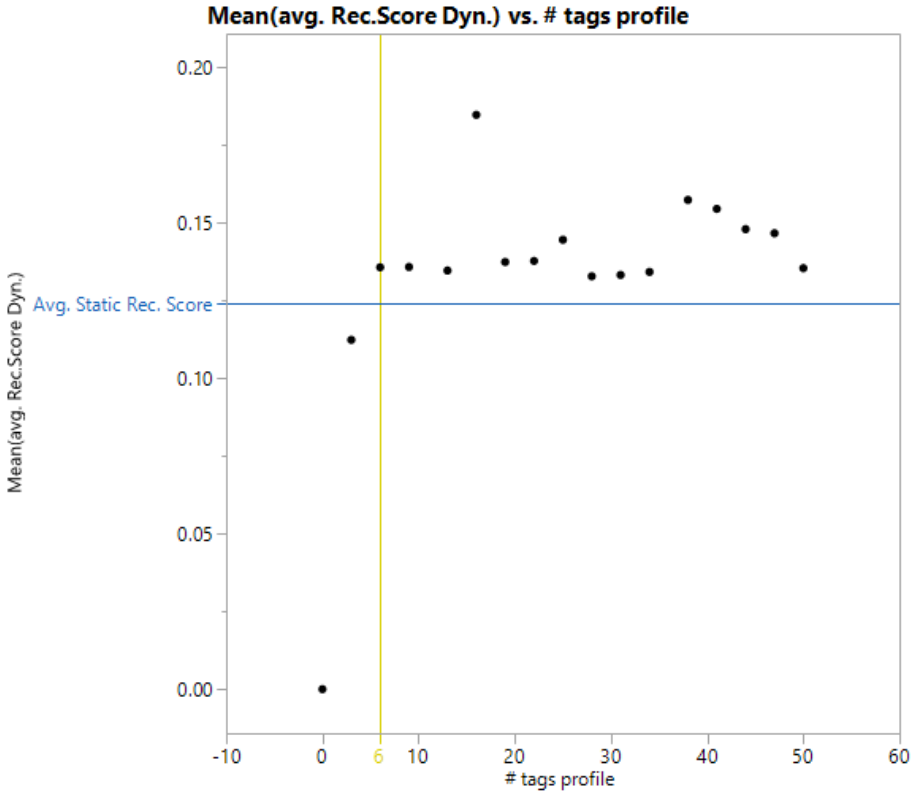


Figure 5.11. **A plot of the average dynamic recommender score versus the number of student profile tages.** This plot shows the average recommender scores from the dynamic recommender system as a function of the number of student profile tags used in each of the 17 experimental design points. The dots represent the average values obtained from the simulations, and the line represents a fitting curve.

According to our data, a minimum number of about 6 tags per profile is necessary to obtain a recommender score above the average static recommender score in our settings. In

Figure 5.11, it appears that 15 tags per profile may be optimal, as the inclusion of more tags clearly do not improve the recommender score. Given the limited scope of our experimental design, as well as our limited number of replications, we recommend using this analysis as a preliminary basis for future studies (see Section 6.1.2). Also, we caution that this number of optimal student profile tags relies on matching course content tags with student interest tags that were generated randomly from a pool of course content tags. In a real scenario, we envision that student profiles as well as course tags will implicate a wider range of tag categories, so that a higher overall number of profile tags is very likely to be optimal.

Next, we analyze how the positive / negative feedback distributions for course content tags and student profile tags affects the recommender score in the dynamic m odel. This is of interest for the future, as the feedback distribution can (at least to some extent) be influenced by the quality of tags. We believe that well tagged course content is more likely to obtain positive feedback in terms of met expectations, and well designed student profile tags will increase the likelihood of identifying relevant learning content for the student. To check for such correlation, we generate a contour plot showing average recommender scores as a function of both types feedback distribution (Figure 5.12).

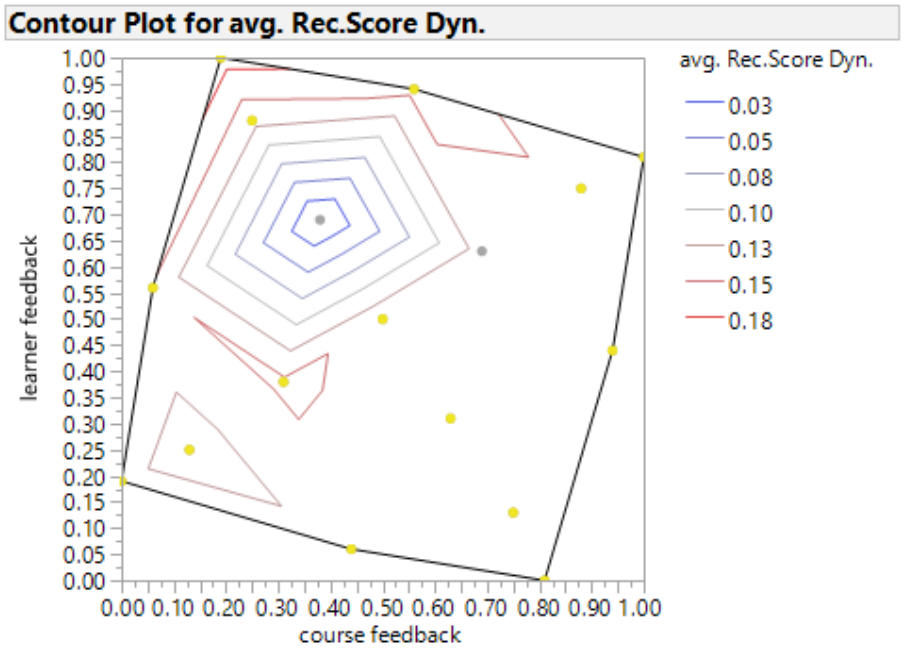


Figure 5.12. **Impact of feedback distributions on the recommender score in the dynamic model.** The x-axis and the y-axis represent the proportion (or percentage) of positive feedback out of 1.00 (or 100%) feedback provided, whereas the remaining feedback is negative.

As can be seen from Figure 5.12, contour lines rely on a very low density of actual data points covered in the experimental design. We thus refrain from drawing any further conclusions out of this figure and suggest to repeat the analysis in a larger experiment consisting of more space-filling design points in the future.

All together, experimental design can be well applied with our software and can be used to analyze relations of different input parameters and output variables. While the small scale experimental design described here was sufficient to give hints on the effect of different values for δ and on a minimum number of profile tags, experiments of a larger scale are necessary to obtain reliable results for relations between feedback distribution and recommender score. This and other suggestions to improve our personalized learning framework are described in the following chapter.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 6:
Discussion

The overall goal of this research project was to generate a personalized and adaptive learning framework. This framework must be able to suggest learning content to an individual student based on his/her interests and learning preferences, and make modifications based on how the student best learns during the period of study. This requires an adaptive recommender system, which is able to match learning content with individual student profiles. Since students' interests and learning preferences often change during the course of study, a functional system must be able to adapt to these changes, update the profiles accordingly, and identify the best fitting learning content at each step. In this chapter, we summarize our progress, discuss several of the challenges, and outline the many opportunities available for continued work.

6.1 Summary and Recommendations for Future Work

To achieve our goal of developing a personalized and adaptive learning framework, we first demonstrated a network science approach to our data model, where we organized all available learning content in a multilayer network of knowledge. Learning content is tagged with metadata from multiple categories, such as content, prerequisites, instructors, etc., where each category represents one layer in the network of knowledge. Course tags are evaluated by students, and updated and improved in accordance with this student feedback. Student profiles also contain multiple tag categories describing interests, learning preferences, previous knowledge, etc., and these tags are adapted according to feedback as well as learning progress.

Next, we established and investigated several methods and techniques to set up a software prototype to demonstrate this personalized learning framework. We analyzed techniques to assign tags to courses and student profiles, we simulated students' feedback on these tags, and we developed a software prototype that is able to match student profile tags and course content tags, recommend courses to students, collect student feedback, and use the feedback to adapt tags in order to improve the recommendation of future courses.

6.1.1 Assignment of Metadata Tags

Assignment of appropriate metadata tags in a standardized, objective and automated manner is a major challenge for the establishment of a personalized learning platform. We tested several automatic approaches to extract tags describing course content, course prerequisites, course grading distributions, and student profiles.

6.1.1.1 Course Content Tags

To obtain course content tags, we designed a program that automatically extracts the most common words out of pre-existing material from NPS, and run it on course syllabi as well as on learning material (PPTX slides). While the tags obtained from course syllabi are heterogeneous and contain many tags which do not actually describe course content, the learning material from the one course we analyzed revealed good tags for the course content. Unfortunately, we did not have access to such material for more courses (especially from the OR department), so we elected to continue working with the tags extracted from the syllabi for our simulation experiments. In future work, content tags extracted from course material should be used, not only because the content would be described better, but also because the educational material would be tagged at a higher fidelity (e.g., by extracting tags for each lesson of the course separately). Nevertheless, our common word extraction approach seems an appropriate automatic tool to assign content tags to educational materials.

6.1.1.2 Course Prerequisite Tags

To tag courses that require the knowledge from a previous course with prerequisite tags, we automatically extracted information from the course catalog by searching for specific word patterns. This revealed appropriate tags for some of the courses, but did not cover the whole list of courses in the catalog. In order to better develop this prerequisite list, we investigated an indirect approach by analyzing student transcript data to identify courses that were taken by students prior to taking a specific course. If a specific course was previously taken by all course participants, we assumed this previous course must be a prerequisite. While this approach was able to fill some of the gaps, and had overlapping results for several courses that were already identified using the first approach, we believe analyzing a larger quantity of student transcript data in the future will lead to greater insights. From the very limited amount of transcript data available for this thesis, we most likely obtained false positive prerequisite course results when the number of students in a particular course is very

small. Furthermore, false negative results could be obtained if students take prerequisite courses at other academic institutions. To overcome these challenges, we recommend, in future work, to consider including the relationships between course content (e.g., subjects or topics) into the analysis. In general, a combination of both approaches—after the improvements mentioned—may provide a useful tool to automatically assign prerequisites to any educational material (not just material tied to a course).

6.1.1.3 Course Grading Distribution

From the student transcript data, we were also able to extract success rates and grading distributions for the courses. Although we did not yet use this data in our software prototype, we feel that in future work it might be used to investigate the effect grades have on student feedback.

6.1.1.4 Student Profile Tags

From the student transcript data, we looked to extract tags that are relevant for establishing student profiles. While we successfully extracted personal information and information on prior education and previous courses, which will be relevant for a more advanced software prototype, we found that we could not extract useful tags describing students' interests or learning preferences. At this initial phase of the project, our aim was to establish a recommender system based on student interests and matching course content. We therefore decided to simulate student profiles by randomly assigning metadata from our available course content tag pool. This allowed us to run several experiments with the software prototype and to analyze the influence of some parameters. However, it is, admittedly, a rather artificial approach that must be improved in the future through the use of real student data, such as data obtained from the use of interest questionnaires or surveys. If enough data is available, interest tags could also be obtained by analyzing students' learning behavior (e.g., which courses are taken, which grades achieved) or by transferring some tags from other students with a very similar learning behavior. Nevertheless, the simulation of student profiles, as established in this thesis, offers a valuable tool to run basic experiments and to simultaneously study the influence of several parameters on the course recommendation algorithm.

6.1.2 Feedback Simulation and Dynamic Course Recommendation

In this thesis, we established a software prototype that is able to simulate student profiles and match these simulated profiles with the optimal course content using tags obtained from course syllabi. In the experiments we ran in this thesis, the matching is exclusively based on cosine similarity, but the software already contains options to use k-NN or a hybrid of the two approaches. Furthermore, the software we developed simulates student feedback after each course, and includes this feedback to adapt both, student profile and course content tags. We achieved this inclusion of feedback by weighting all tags according to their priority or relevance. Negative feedback decreases, and positive feedback increases the tag weight. Furthermore, we included the ability to regulate the effect size of the feedback by setting the value of the δ parameter, with high δ values causing the feedback to have a stronger impact. We demonstrated that this feedback-driven dynamic leads to weight changes in both student profile and course content tags, and that these changes result in altered recommended course paths. More importantly, adaptive tagging in the dynamic recommendation system increased the overall recommender score, which we used to measure the overall quality of the student-to-course match. The system developed here does not currently allow the deletion or replacement of tags. We believe this option can, and should, be included in future work, so that tags which continue to get negative feedback and fall below a certain threshold can be removed from the system. Furthermore, we defined a maximal (10) and minimal (1) weight for tags, without much analysis. We believe the selection of an appropriate weight range, and resulting advantages and disadvantages, should be investigated in the future.

Additionally, we included in our prototype the analysis of several metrics, including the tracking of tag weights over time, courses over time, and course feedback over time. In order to keep things simple during the initial phase of this research, we did not yet include how feedback impacts instructors or content curators. Therefore, feedback important to both these entities should be included as part of future research.

We simulated feedback in our prototype according to distributions that we subjectively defined in our software initialization file. In the future, we encourage the use of a more objective approach. With more data available, empirically-derived distributions could be used. Still, we believe the software prototype, with its ability to simulate student feedback, is a good tool to run fast experiments, to investigate feedback interactions with other variables, and to optimize input parameters such as δ. For the experiments we ran for this thesis, we

assumed that feedback is given by each student independently according to the defined distribution. This is the most simple and basic approach. However, our software prototype is also able to simulate feedback trends, where we assume that students influence each other, which increases the probability of positive or negative feedback depending on feedback by other students. This can be extended to feedback trends within academic communities (i.e., cohorts), where a trend occurs among a certain group of students, and a different trend may occur in a different group. Although we did not investigate the simulation of trends within communities, this option is available in the current software prototype, and parameters like duration of trends or the probability of joining a community can be modified in the input settings. Thus, the software prototype stands ready to simulate more complex feedback scenarios that may paint a picture closer to real student behavior.

We ran several experiments to verify our simulation software. First, we confirmed that the changed course recommendations in the dynamic recommender system are caused by feedback-induced changes of tag weights and not random. We set the parameter δ to zero, thus still simulating feedback, but eliminating the impact of the feedback on the tag weights. The resulting tag weights remained constant throughout the experiment, and the recommended course paths for the dynamic recommender system were 100 % identical to the ones suggested by the static recommender system, proving that all observed changes are indeed based on the student feedback. Furthermore, we tested if exclusively positive feedback increases, and exclusively negative feedback decreases tag weights. We confirmed both, indicating that the software was functioning correctly in this regard. Lastly, we demonstrated that the software can run through a series of input parameters allowing the identification of interactions between different input variables and quantifying the impact on the response (or output) variables. Using a small experimental design approach, we found that a minimum number of profile tags is necessary to obtain decent recommender scores, while very high numbers of profile tags do not necessarily further improve this score. We believe that information such as this should be used in the future to identify and optimize factors that can be influenced by the platform administrator.

6.2 Conclusion and Outlook

The overall goal of the CHUNK Learning project is to offer a framework for personalized learning which utilizes a network science approach and generates a learning environment for

personalized adaptive learning. In this thesis, we established an initial software prototype that is able to adaptively recommend learning content to individual students based on their interests, by implementing student feedback. This prototype must be improved through the inclusion of many other categories like prior knowledge / course prerequisites, learning preferences / type of educational material, intensity levels, instructor information and overall study and application goals, to name a few. Our approach of allowing student feedback to adapt the weighting of tags within a single category could easily be transferred to these new categories. The inclusion of these additional categories and matching with real student profiles will allow more differential matches, but will also generate more complex overlaps. We believe that multilayer networks, like those described in this thesis, will help to deal with this increased complexity.

Our software prototype is able to take feedback by students into account and adapt tag weights accordingly. The overall idea behind this thesis was to design a software prototype, which incorporates a methodology enabling easy selection and weighting of tags by instructors as well as students. Therefore, in future versions, the program should be designed so that it enforces and collects feedback, evaluates it, and adapts tag weights accordingly. This adaptation could also imply that tags with predominantly negative feedback are removed and replaced by better tags.

Our initial software prototype, which includes the simulation of students and methods to extract tags, is a valuable tool for initial investigation into a personalized learning framework, more real data should be implemented in the future. Furthermore, proper verification and validation experiments should be performed and compared to empirical observations and experiences.

Some fine tuning of the methods established here will also help to further improve the software. Details on our ideas for future work were described in the previous section, and are summarized in Table 6.1.

Table 6.1. Future Work

	Current Limitation	**Future Work**
Feedback Simulation	Defined distributions	Sampling from real data or data gathered from surveys
		Agent-based implementation of students (individually different behavior based on own learning experience)
Simulation Implementation	Course data and student data is represented by objects stored in dictionaries	Storing this data in a graph database intuitively extends the ideas discussed in this thesis.
Validation	System is verified but not validated	Validation of the simulation with real data / students.
Update of Tags	Metadata stays in the system	Replacement based on threshold
Recommender	Empty profiles do not receive specific recommendations	Curated recommendations (like top 10)
	Tags are not part of hierarchy	Inclusion of hierarchies

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A:
Results from Automatic Tagging of Metadata

A.1 Extraction of Metadata from Course Syllabi

Table A.1. Term and Frequency List of Most Common Words from Detailed Analysis of Course Syllabi

course	(common word, count)
OA1600	('introduction', 9), ('programming', 9), ('analysis', 6), ('savage', 4), ('topic', 3)
OA2900	('thesis', 5), ('operations', 3), ('research', 3), ('opportunities', 3), ('pass/fail', 2)
OA3101	('probability', 5), ('statistics', 5), ('oa', 4), ('homework', 4), ('quarter', 3)
OA3102	('statistical', 6), ('statistics', 3), ('gl', 3), ('00', 3), ('0', 3)
OA3103	('data', 7), ('analysis', 6), ('glasgow', 5), ('oa3103', 4), ('set', 3)
OA3201	('gams', 5), ('models', 5), ('linear', 4), ('problems', 4), ('understand', 3)
OA3301	('problems', 1), ('ph.d', 1), ('put\\\\3301', 1), (""", 1), ('ine-mailsubject', 1)
OA3302	('email', 1), ('put', 1), ('oa3302', 1), ('insubjectline', 1), ('glasgow254', 1)
OA3304	('decisiontheory', 1), ('coursesyllabus', 1), ('january5,2015', 1), ('ph.d', 1), ('mpatkins', 1)
OA3401	(""", 61), ('1./', 12), ('./0', 10), ("'/", 9), ("'2", 8)
OA3411	('hsi', 18), ('program', 6), ('human', 5), ('systems', 5), ('acquisition', 5)
OA3412	('acquisition', 8), ('systems', 7), ('human', 6), ('integration', 6), ('hsi', 6)
OA3413	(""", 51), ('234', 15), ("'3", 11), ('–', 6), ('3=', 6)
OA3801	(""", 2), ('manipulating', 1), ('displaying', 1), ('davidl.alderson', 1), ('ph.d', 1)
OA4301	('process', 12), ('models', 11), ('understand', 9), ('word', 9), ('poisson', 8)
OA4333	('model', 6), ('simulation', 5), ('analysis', 3), ('apply', 3), ('oa3302', 2)
OA4401	(""", 57), ("'2", 18), ('0', 17), ('8.2', 14), ('.24', 12)
OA4408	('organizational', 5), ('human', 5), ('team', 5), ('macroergonomics', 4), ('factors', 4)
OA4414	('hsi', 12), ('learned', 5), ('previous', 4), ('courses', 4), ('-week', 4)
OA4415	('courses', 5), ('systems', 4), ('program', 4), ('hsi', 4), ('human', 3)
OA4602	('analytical', 6), ('analysis', 5), ('operational', 5), ('usn', 4), ('capt', 3)
OA4603	('test', 12), ('evaluation', 5), ('project', 5), ('group', 4), ('acquisition', 3)
OA4604	('wargaming', 4), ('wargame', 4), ('develop', 3), ('operations', 3), ('analysis', 3)
OA4607	('oa4607', 1), ('coursesyllabus', 1), ('ph.d', 1), ('mpatkins', 1), ('glasgow249', 1)
OA4656	('combat', 10), ('models', 9), ('case', 6), ('aoa', 5), ('studies', 4)
OA4801	('spreadsheet', 6), ('operations', 4), ('research', 4), ('military', 4), ('vba', 4)
OS2080	('probability', 12), ('distributions', 7), ('statistics', 4), ('variables', 4), ('data', 4)

Continued on next page

Table A.1 – continued from previous page

course	common word, count)
OS2103	('probability', 10), ('conditional', 10), ('expected', 10), ('understand', 10), ('definition', 8)
OS3008	('excel', 11), ('operations', 4), ('research', 4), ('techniques', 3), ('using', 3)
OS3081	('analysis', 8), ('analytical', 8), ('systems', 6), ('studies', 5), ('case', 4)
OS3082	('systems', 5), ('analysis', 5), ('analytical', 5), ('studies', 4), ('support', 4)
OS3104	('statistics', 3), ('data', 3), ('engineering', 2), ('student', 2), ('831', 2)
OS3105	('chapter', 6), ('chapters', 4), ('probability', 3), ('lectures', 3), ('tbd', 3)
OS3111	('probability', 5), ('statistics', 4), ('data', 3), ('exercises', 3), ('based', 2)
OS3112	('design', 8), ('experimental', 5), ('designs', 5), ('experiments', 4), ('research', 4)
OS3113	('20', 3), ('15', 3), ('lectures', 3), ('chapters', 3), ('please', 3)
OS3211	('systems', 1), ('optimization', 1), ('spreadsheet', 1), ('modeling', 1), ('decision', 1)
OS3307	('modeling', 4), ('event', 3), ('sample', 3), ('models', 3), ('distributions', 2)
OS3380	('ph.d', 1), ('googlevoice', 1), ('510', 1), ('969-2004', 1), ('simio', 1)
OS3604	('data', 8), ('statistics', 5), ('use', 5), ('get', 4), ('ample', 4)
OS3680	('cle', 4), ('reading', 4), ('naval', 2), ('tactical', 2), ('analysis', 2)
OS4012	('ˇ', 14), ('ˇˇ', 2), ('˜', 2), ('˘˘', 2), ('ˇ', 2)
OS4083	('analysis', 12), ('systems', 5), ('system', 3), ('completion', 3), ('presentation', 3)
OS4621	("", 3), ('ical', 1), ('economic', 1), ('state', 1), ('studentsatnps', 1)
OS4702	('˝', 4), ('', 4), ('‘', 3), ("", 2), ("˚", 2)
OA2801	('basic', 5), ('computer', 4), ('data', 4), ('file', 4), ('python', 3)
OA3402	('question', 24), ('research', 23), ('example', 15), ('empirical', 14), ('step', 14)
OA3501	('excel', 12), ('homework', 12), ('–', 10), ('‘', 8), ('', 8)
OA3602	('search', 19), ('detection', 14), ('hr', 12), ('target', 7), ('discrete', 4)
OA3611	('logistics', 32), ('operational', 17), ('planning', 10), ('information', 10), ('oplog', 8)
OA3802	('data', 14), ('text', 7), ('example', 7), ('tools', 6), ('bash', 6)
OA4101	('chap', 7), ('designs', 6), ('experiments', 5), ('analysis', 5), ('–', 5)
OA4106	('models', 11), ('data', 7), ('regression', 7), ('linear', 5), ('model', 5)
OA4108	('data', 21), ('mining', 8), ('large', 6), ('techniques', 6), ('r', 6)
OA4109	('survey', 14), ('design', 9), ('notes', 8), ('surveys', 7), ('analysis', 6)
OA4118	('r', 14), ('exercise', 14), ('data', 12), ('using', 9), ('machine', 8)
OA4201	('learn', 12), ('optimization', 10), ('convex', 9), ('homework', 9), ('problems', 7)
OA4202	('network', 20), ('problems', 16), ('chapter', 15), ('flow', 12), ('shortest', 12)
OA4203	('programming', 20), ('problems', 12), ('optimization', 8), ('integer', 7), ('linear', 6)
OA4205	('programming', 4), ('conditions', 4), ('optimality', 3), ('optimization', 3), ('algorithms', 3)
OA4406	('safety', 16), ('health', 7), ('survivability', 6), ('occupational', 6), ('habitability', 5)
OA4611	('logistics', 12), ('joint', 5), ('material', 5), ('quizzes', 4), ('participation', 3)

Continued on next page

Table A.1 – continued from previous page

course	common word, count)
OA4613	('energy', 45), ('operations', 18), ('logistics', 16), ('military', 14), ('operational', 12)
OA4655	('modeling', 24), ('models', 20), ('combat', 16), ('warfare', 12), (' 1', 10)
OA4657	('model', 9), ('simulation', 5), ('models', 4), ('dod', 4), ('description', 3)
OA4702	('cost', 24), ('–', 18), ('days', 14), ('analysis', 10), ('estimation', 8)
OS3006	('“', 25), ('”', 25), ('excel', 17), ('homework', 17), ('quizzes', 11)
OS3007	('energy', 18), ('optimization', 13), ('systems', 12), ('•', 10), ('modeling', 9)
OS3080	('analysis', 9), ('probability', 7), ('models', 7), ('data', 5), ('regression', 5)
OS3180	('probability', 14), ('statistics', 12), ("'", 9), ('available', 7), ('synchronous', 7)
OS3311	('probability', 6), ('models', 6), ('military', 5), ('combat', 3), ('situations', 3)
OS3640	('ied', 9), ('counter', 7), ('homework', 5), ('improvised', 4), ('explosive', 4)
OS3701	('cost', 24), ('–', 18), ('analysis', 10), ('hour', 9), ('estimation', 8)
OS4010	('–', 28), ('risk', 22), ('ch', 18), ('g', 13), ('w', 13)
OS4011	('–', 27), ('risk', 23), ('ch', 18), ('g', 13), ('w', 13)
OS4013	('decision', 15), ('analysis', 7), ('modeling', 5), ('solve', 4), ('problems', 4)
OS4080	('analysis', 10), ('ce', 9), ('cost', 6), ('engineering', 6), ('cba', 6)
OS4081	('team', 20), ('project', 16), ('cost', 15), ('presentation', 14), ('analysis', 13)
OS4082	('analysis', 17), ('team', 17), ('cost', 14), ('top', 11), ('project', 11)
OS4106	('model', 12), ('linear', 6), ('models', 5), ('regression', 5), ('response', 4)
OS4118	('methods', 10), ('machine', 6), ('learning', 6), ('techniques', 6), ('statistical', 5)
OS4680	('analysis', 9), ('problems', 7), ('practice', 5), ('homework', 4), ('linear', 4)
OS4701	('manpower', 11), ("'", 11), ('models', 8), ('analysis', 7), ('data', 6)
OS4703	('cost', 16), ('support', 13), ('decision', 10), ('analyses', 10), ('programs', 8)

Table A.2. List of Excluded Words from Detailed Analysis of Course Syllabi

excluded organizational words
'day', 'course', '1', 'montuewedthu', 'Tu', 'Th', 'Fri', 'Mon', 'Hours', 'hours', 'Thursday', 'Monday', 'Aug', 'Jul', 'able', 'assignments', 'final', 'week', 'grade', 'sakai', 'Schedule', 'appoinment', 'Class', 'Reading', 'weekend', 'Please', 'Office', 'bring', 'hours', 'TBD', 'appointment', 'Grading', 'Labs', 'class', 'slides', 'one', 'syllabus', '2', '3', '4', '5', '6', '7', '8', '9', '¥', 'instructor', 'certificate', 'e.g.', 'work', 'courseinformation', 'professor', 'january', 'may', 'hrs', 'lab', 'study', 'phone', 'ph.d.', 'email', 'students', 'textbook', 'supplemental', 'via', 'nps', 'Chapter', 'Homework', 'lecture', 'Lectures', 'Lab', 'exam', 'Exam', 'os3211', 'oa3304', 'OS3380', 'OA3103', 'oa3103', 'syllabusforos3380', 'th', 'GL', 'LCDR',':00', 'Glasgow', 'nps.edu', 'edu', 'http', 'OA'

A.2 Extraction of Metadata from Course Content

Table A.3. Term and Frequency List of Most Common Words from Detailed Analysis of the MA4404 Course Content

topic	(common word, count)
01-RealComplexNetworks	('networks', 94), ('network', 29), ('source', 21), ('internet', 16), ('“', 12), ('”', 12), ('nodes', 11), ('social', 11), ('newman', 11), ('information', 10), ('web', 10), ('different', 10), ('data', 8), ('graphs', 7), ('edges', 7)
02-NetworkScienceOverview1	('networks', 23), ('network', 15), ('graph', 13), ('graphs', 13), ('science', 12), ('theory', 12), ('complex', 11), ('eulerian', 9), ('random', 9), ('social', 9), ('”', 9), ('“', 8), ('königsberg', 8), ('small', 7), ('cited', 6)
03-NetworkScienceOverview2	('networks', 32), ('model', 29), ('random', 22), ('graphs', 19), ('world', 19), ('network', 17), ('nodes', 16), ('small', 15), ('path', 14), ('average', 12), ('source', 12), ('www', 11), ('degree', 10), ('scale', 9), ('free', 9)
05-NetworkScienceOverview2	('networks', 36), ('model', 29), ('random', 24), ('graphs', 21), ('world', 20), ('network', 16), ('nodes', 16), ('small', 15), ('degree', 14), ('path', 13), ('average', 12), ('scale', 11), ('free', 11), ('complex', 10), ('distribution', 10)
05-SyntheticModelsComplexNetworks	('random', 28), ('networks', 23), ('degree', 18), ('model', 17), ('network', 15), ('nodes', 15), ('small', 14), ('world', 13), ('n', 11), ('models', 10), ('“', 9), ('”', 9), ('graph', 9), ('distribution', 8), ('erdős-rényi', 7)
06-CommunitiesModularity	('communities', 32), ('networks', 32), ('community', 32), ('network', 21), ('nodes', 19), ('https', 19), ('et', 15), ('al', 15), ('graph', 13), ('model', 13), ('sbm', 13), ('detection', 12), ('example', 12), ('large', 12), ('algorithm', 12)
07-CommunitiesModularity	('communities', 32), ('networks', 32), ('community', 32), ('network', 21), ('nodes', 19), ('https', 19), ('graph', 13), ('detection', 12), ('example', 12), ('large', 12), ('algorithm', 12), ('method', 12), ('model', 12), ('data', 11), ('structure', 9)
07-CorePeriphery	('networks', 22), ('structure', 14), ('core-periphery', 13), ('communities', 11), ('“', 11), ('core', 9), ('”', 9), ('k-core', 8), ('large', 8), ('j.', 8), ('m.', 8), ('vol', 8), ('properties', 7), ('community', 7), ('p.', 7)

Continued on next page

Table A.3 – continued from previous page

topic	(common word, count)
08-Centralities_1	('1/20', 13), ('networks', 11), ('centralities', 8), ('centrality', 8), ('degree', 7), ('group', 6), ('central', 5), ('social', 5), ('network', 4), ('exploration', 4), ('node', 4), ('table', 4), ('vertex', 4), ('https', 4), ('location', 3)
10-Centralities_Gephi	('gephi', 9), ('nodes', 9), ('centralities', 8), ('centrality', 7), ('graph', 7), ('network', 6), ('python', 5), ('code', 5), ('part', 5), ('top', 5), ('use', 4), ('version', 4), ('try', 4), ('closeness', 4), ('directed', 4)
11-Centralities_2	('0', 182), ('centrality', 50), ('eigenvector', 30), ('closeness', 14), ('example', 10), ('vertices', 9), ('social', 6), ('networks', 6), ('degree', 6), ('13', 6), ('“', 6), ("”", 6), ('deg', 6), ('recall', 5), ('network', 5)
11-Centralities_3	('page', 29), ('pages', 28), ('pagerank', 20), ('centrality', 13), ('web', 12), ('matrix', 12), ('important', 11), ('source', 8), ('degree', 8), ('hyperlinks', 8), ('v', 7), ('”', 7), ('search', 7), ('u', 6), ('algorithm', 6)
11-Centralities_4	('centrality', 18), ('betweenness', 15), ('matrix', 12), ('pages', 11), ('formula', 9), ('degree', 9), ('pagerank', 8), ('page', 8), ('would', 8), ('flow', 7), ('centralities', 6), ('vertex', 6), ('paths', 6), ('”', 5), ('get', 5)
14-ClusteringCoefficient	('clustering', 10), ('networks', 5), ('source', 5), ('scale', 5), ('free', 5), ('coefficient', 5), ('n.', 4), ('local', 4), ('section', 3), ('“', 3), ('”', 3), ('pr^zulj', 3), ('d.', 3), ('g.', 3), ('corneil', 3)
14-Small World	('small', 8), ('avg', 6), ('world', 5), ('effect', 5), ('path', 4), ('networks', 3), ('model', 3), ('example', 3), ('word', 3), ('ernesto', 3), ('estrada', 3), ('clust', 3), ('shortest', 2), ('”', 2), ('milgram', 2)
15-nodeSimilarity	('similarity', 18), ('equivalence', 16), ('regular', 14), ('equivalent', 14), ('nodes', 11), ('structural', 10), ('vertices', 9), ('two', 8), ('neighbors', 8), ('d1', 7), ('q', 7), ('network', 6), ('networks', 6), ('defn', 6), ('many', 6)
16-Homophily	('degree', 21), ('characteristics', 20), ('networks', 14), ('homophily', 12), ('assortativity', 12), ('assortative', 12), ('scalar', 12), ('network', 11), ('newman', 11), ('mixing', 11), ('vertices', 11), ('2003', 11), ('based', 10), ('age', 10), ('high', 8)

Continued on next page

Table A.3 – continued from previous page

topic	(common word, count)
MA4404_schedule_20180105	('outline', 2), ('ma4404', 1), ('schedule', 1), ('excellence', 1), ('knowledge', 1)
Rudy-ML_Graphs1	('learning', 21), ('network', 19), ('model', 16), ('performance', 12), ('training', 11), ('"', 10), ('nodes', 9), ('machine', 8), ('“', 8), ('data', 8), ('algorithms', 8), ('networks', 7), ('problem', 6), ('set', 6), ('degree', 6)
Rudy-ML_Graphs2	('network', 35), ('distance', 9), ('analysis', 8), ('information', 8), ('mean', 8), ('small', 7), ('world', 7), ('nodes', 7), ('classification', 6), ('incomplete', 5), ('statistics', 5), ('erdos-renyi', 5), ('edge', 5), ('sw', 5), ('networks', 4)

Table A.4. Term and TF-IDF Score of Most Common Words from Detailed Analysis of the MA4404 Course Content

topic	word, TF-IDF score
01-RealComplexNetworks	('networks', '0.4718'), ('network', '0.1642'), ('internet', '0.1279'), ('source', '0.1113'), ('source http', '0.0963'), ('biological', '0.0923'), ('introduction newman', '0.0923'), ('www', '0.0859'), ('level', '0.0854'), ('http', '0.0848')
02-NetworkScienceOverview1	('königsberg', '0.1796'), ('networks', '0.1756'), ('eulerian', '0.1602'), ('theory', '0.1517'), ('erdös', '0.1347'), ('science', '0.1323'), ('graph theory', '0.1283'), ('graphs', '0.1253'), ('network', '0.1237'), ('cited', '0.1184')
03-NetworkScienceOverview2	('model', '0.2028'), ('networks', '0.1774'), ('barabasi', '0.1654'), ('world', '0.1619'), ('random', '0.1596'), ('scale free', '0.1564'), ('free', '0.1386'), ('small', '0.1330'), ('small world', '0.1300'), ('graphs', '0.1263')
05-NetworkScienceOverview2	('model', '0.2087'), ('networks', '0.2041'), ('random', '0.1779'), ('world', '0.1743'), ('scale free', '0.1711'), ('free', '0.1516'), ('graphs', '0.1437'), ('small', '0.1369'), ('scale', '0.1358'), ('small world', '0.1338')

Continued on next page

Table A.4 – continued from previous page

topic	word, TF-IDF score
05-SyntheticModelsComplexNetworks	('random', '0.2133'), ('networkx', '0.1821'), ('erdős', '0.1475'), ('world', '0.1466'), ('generators', '0.1463'), ('small', '0.1397'), ('networks', '0.1385'), ('networkx generators', '0.1317'), ('model', '0.1315'), ('erdős rényi', '0.1291')
06-CommunitiesModularity	('community', '0.2920'), ('communities', '0.2779'), ('sbm', '0.1504'), ('networks', '0.1359'), ('et al', '0.1327'), ('et', '0.1237'), ('al', '0.1237'), ('networkit', '0.1157'), ('iti kit', '0.1041'), ('kit', '0.1041')
07-CommunitiesModularity	('community', '0.3111'), ('communities', '0.2960'), ('networks', '0.1448'), ('networkit', '0.1233'), ('recon', '0.1109'), ('kit edu', '0.1109'), ('networkit iti', '0.1109'), ('https networkit', '0.1109'), ('kit', '0.1109'), ('iti kit', '0.1109')
07-CorePeriphery	('core', '0.3768'), ('periphery', '0.2819'), ('core periphery', '0.2637'), ('dense', '0.1758'), ('networks', '0.1445'), ('communities', '0.1300'), ('vol', '0.1299'), ('decomposition', '0.1108'), ('cores', '0.1108'), ('structure', '0.1105')
08-Centralities_1	('20 20', '0.5529'), ('20', '0.2698'), ('centralities', '0.1458'), ('group', '0.1269'), ('exploration', '0.1164'), ('centrality', '0.1083'), ('periodic', '0.1057'), ('networks', '0.1002'), ('central', '0.0911'), ('class exploration', '0.0873')
10-Centralities_Gephi	('gephi', '0.1703'), ('centralities', '0.1641'), ('centrality', '0.1525'), ('export', '0.1311'), ('metricsv2', '0.1311'), ('networkx', '0.1192'), ('closeness', '0.1190'), ('save', '0.1152'), ('python', '0.1084'), ('nodes', '0.1066')
11-Centralities_2	('centrality', '0.3971'), ('eigenvector', '0.3844'), ('eigenvector centrality', '0.3472'), ('closeness', '0.1736'), ('example eigenvector', '0.1536'), ('closeness centrality', '0.1488'), ('class eigenvector', '0.1365'), ('class', '0.0976'), ('bonacich', '0.0853'), ('phillip', '0.0853')
11-Centralities_3	('page', '0.3241'), ('pagerank', '0.3024'), ('pages', '0.2774'), ('important', '0.1313'), ('web', '0.1311'), ('matrix', '0.1189'), ('centrality', '0.1153'), ('hyperlinks', '0.1108'), ('lecture3', '0.1005'), ('degree matrix', '0.1005')

Continued on next page

Table A.4 – continued from previous page

topic	word, TF-IDF score
11-Centralities_4	('betweenness', '0.2796'), ('centrality', '0.2149'), ('betweenness centrality', '0.1864'), ('formula', '0.1831'), ('flow', '0.1796'), ('pagerank', '0.1628'), ('matrix', '0.1600'), ('pages', '0.1467'), ('degree matrix', '0.1353'), ('lecture3', '0.1353')
14-ClusteringCoefficient	('clustering', '0.2167'), ('coefficients', '0.1636'), ('clustering coefficients', '0.1636'), ('local clustering', '0.1636'), ('higher', '0.1298'), ('protein', '0.1298'), ('higher order', '0.1227'), ('free geometric', '0.1227'), ('qbio mn', '0.1227'), ('0404017', '0.1227')
14-Small World	('avg', '0.2539'), ('small', '0.2127'), ('world effect', '0.2116'), ('effect', '0.1670'), ('small world', '0.1664'), ('ernesto', '0.1600'), ('ernesto estrada', '0.1600'), ('clust', '0.1600'), ('path avg', '0.1600'), ('avg clust', '0.1600')
15-nodeSimilarity	('equivalence', '0.3558'), ('equivalent', '0.2737'), ('similarity', '0.2691'), ('regular equivalence', '0.2224'), ('regular', '0.1716'), ('d1', '0.1557'), ('structural', '0.1495'), ('defn', '0.1173'), ('pearson', '0.1112'), ('d2', '0.1112')
16-Homophily	('characteristics', '0.2737'), ('scalar', '0.2260'), ('age', '0.2072'), ('scalar characteristics', '0.1884'), ('mixing', '0.1821'), ('assortativity', '0.1642'), ('homophily', '0.1533'), ('assortative', '0.1519'), ('assortative mixing', '0.1490'), ('degree', '0.1356')
MA4404_schedule_20180105	('outline', '0.5870'), ('outline outline', '0.3339'), ('ma4404 schedule', '0.3339'), ('schedule', '0.3339'), ('schedule excellence', '0.3339'), ('knowledge outline', '0.2935'), ('ma4404', '0.2426'), ('excellence', '0.1554'), ('excellence knowledge', '0.1554'), ('knowledge', '0.1267')
Rudy-ML_Graphs1	('learning', '0.2938'), ('performance', '0.2196'), ('training', '0.2013'), ('network', '0.1577'), ('model', '0.1397'), ('machine', '0.1119'), ('testing', '0.1041'), ('machine learning', '0.0979'), ('model performance', '0.0833'), ('phase', '0.0833')

Continued on next page

Table A.4 – continued from previous page

topic	word, TF-IDF score
Rudy-ML_Graphs2	('network', '0.3420'), ('mean', '0.1940'), ('mean distance', '0.1697'), ('classification', '0.1455'), ('distance', '0.1290'), ('incomplete', '0.1212'), ('sw', '0.1212'), ('world network', '0.1094'), ('small world', '0.1004'), ('incomplete information', '0.0970')

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B:
Determination of Course Prerequisites

B.1 Determination of Course Prerequisites from the NPS Catalog and Student Transcripts

Table B.1. Derived Prerequisites for Each Course in the OR Department

Course ID	Prerequisite Description from NPS Catalog	Extracted NPS Courses	Prerequisite Derived from Student Transcripts	Number of Students
OA0810			OA3103, OA3201	97
OA1600	None	None		62
OA2801	None	None		27
OA2900				92
OA3101				94
OA3102	OA3101	OA3101		94
OA3103	OA3102	OA3102		97
OA3201	MA3042	MA3042		97
OA3301	OA3101 or consent of instructor	OA3101		97
OA3302	OA3103 and OA3301	OA3103, OA3301		96
OA3304				75
OA3401				75
OA3411	None	None	Not in data set.	-
OA3412	OA3411	OA3411	Not in data set.	-
OA3413			Not in data set.	-
OA3501	OA3101 or consent of instructor	OA3101	MA3042, OA2900, OA3101	9
OA3602	OS2103 or OA3101	OS2103, OA3101		60
OA3611	None	None	Not in data set.	-
OA3801	AO2801	AO2801	MA3042, OA2900, OA3101	19
OA3802	OA280l or consent of instructor	OA280	Not in data set.	-

Continued on next page

Table B.1 – continued from previous page

Course ID	Prerequisite Description from NPS Catalog	Extracted NPS Courses	Prerequisite Derived from Student Transcripts	Number of Students
OA3900	Departmental approval			94
OA4105	OA3103 or consent of the instructor	OA3103		19
OA4106	OA3103	OA3103	OA3301, OA3103, OA3102, OA3201, OA3101, MA3042	22
OA4109			OA3301, OA3103, OA3201, OA3102, OA3101, OA3302, MA1114, MA3042, OA4201, OA3304, OA2900, OA4202	9
OA4118	OA4106 or consent of instructor	OA4106	Not in data set.	-
OA4201	OA3201	OA3201	OA3201	97
OA4202	OA3201	OA3201	OA3201	97
OA4203	OA3201	OA3201	OA3301, OA3102, OA3201, OA3101, MA3042, OA4201, OA2900	18
OA4301			OA3301, OA3201	26
OA4333	OA3302	OA3302	OA3301, OA3103, OA3201, OA3102, OA3101, MA3042, OA4202	35
OA4401			MA3042, OA2900, OA3101	7
OA4402			Not in data set.	-
OA4406			Not in data set.	-

Continued on next page

Table B.1 – continued from previous page

Course ID	Prerequisite Description from NPS Catalog	Extracted NPS Courses	Prerequisite Derived from Student Transcripts	Number of Students
OA4408			OA3301, OA3103, OA3201, OA3102, OA3101, NW3230, OA4801, OA3302, OA4655, MA3042, OA4201, OA3900, OA3304, OA2900, OA3401, OA4202	3
OA4414	OA3413	OA3413	Not in data set.	-
OA4415			Not in data set.	-
OA4602	A course in basic probability and statistic theory and operational experience in military environments			69
OA4603				37
OA4604	OA4655 or consent of instructor	OA4655	OA3101	67
OA4607			OA3301, MA1025, OA3103, OA3102, OA3101, OA3201, OA4655, OA4801, OA3302, OA0810, MA3042, OA4201, OA3900, OA3304, OA1600, OA2900, OA4602, OA4202	2

Continued on next page

Table B.1 – continued from previous page

Course ID	Prerequisite Description from NPS Catalog	Extracted NPS Courses	Prerequisite Derived from Student Transcripts	Number of Students
OA4613	OA3201 or OS3007 or OA3611 or permission of instructor	OA3201, OS3007, OA3611	MA1025, OA3103, OA3102, OA3101, OA3201, OA4301, OA3302, MA3042, OA3610, OA4201, OA4106, OA2801, MA1118, OA3301, MA1113, MA1114, OA4333, OA1600, OA3501, OA2900, OA4202	1
OA4655			OA3102, OA3103, OA3201	82
OA4702				79
OA4801			OA3201, OA3102	94
OA4910	A background of advanced work in operations research and departmental approval		MA3042, OA2900, OA3101	21
OS2080	Single variable calculus		Not in data set.	-
OS2103	Single variable differentiation and integration at the MA1113 level and multiple integration at the MA1115 level	MA1113, MA1115	Not in data set.	-
OS3006			Not in data set.	-
OS3007			OA3301, OA3103, PH3998, OA3102, OA3101, OA3201, MN4970, NW3230, OA4801, OA3302, OA3801, MA3042, OA4201, OA3900, OA3304, OA2801, MA1118, OA2900, OA3401, OA4202	3

Continued on next page

Table B.1 – continued from previous page

Course ID	Prerequisite Description from NPS Catalog	Extracted NPS Courses	Prerequisite Derived from Student Transcripts	Number of Students
OS3080			Not in data set.	-
OS3081			Not in data set.	-
OS3082			Not in data set.	-
OS3105			Not in data set.	-
OS3111			Not in data set.	-
OS3112	college algebra and OS3111	OS3111	IT1500, OA3103, OA3200, OA3102, OA3101, OA3201, OA4301, OA4801, OA3302, OA0810, MA3042, OA4203, OA4201, TS3002, OA2200, OA3301, TS4003, TS4001, OA4655, TS4002, MA1115, OA3602, OA3900, OA3304, OA2900, OA3401, OA4202	1
OS3113			Not in data set.	-
OS3180	SI1001 or equivalent	SI1001		1
OS3211			Not in data set.	-
OS3307			Not in data set.	-
OS3380			OA3301, IT1500, OS3640, OA3103, OA3102, OA3101, OA3201, OA4301, OA4801, MA3042, OA3900, OA2801, OA4202	1

Continued on next page

Table B.1 – continued from previous page

Course ID	Prerequisite Description from NPS Catalog	Extracted NPS Courses	Prerequisite Derived from Student Transcripts	Number of Students
OS3401			MA1025, OA3200, NW3210, OA3102, OA3101, NW3230, NW3211, MA1117, OA2200, MA3042, OA3304, OA1600, MA1042, OA2900, MA1118	1
OS3604			Not in data set.	-
OS3680			Not in data set.	-
OS3701			Not in data set.	-
OS4010			Not in data set.	-
OS4011			Not in data set.	-
OS4012			Not in data set.	-
OS4013			Not in data set.	-
OS4080			Not in data set.	-
OS4081			Not in data set.	-
OS4082			Not in data set.	-
OS4083			Not in data set.	-
OS4106			Not in data set.	-
OS4118	OA4106 or consent of instructor	OA4106	Not in data set.	-
OS4621			OA3103, OA3102, OA3101, OA3201, OA4656, OA4301, NW3230, OA4801, OA3302, OA0810, MA3042, OA4201, OA4106, OA2801, MA1118, OA3301, OA4655, OA4702, OA4333, OA3602, OA3900, OA3304, OA2900, OA4602, OA4202	1
OS4680			Not in data set.	-

Continued on next page

Table B.1 – continued from previous page

Course ID	Prerequisite Description from NPS Catalog	Extracted NPS Courses	Prerequisite Derived from Student Transcripts	Number of Students
OS4701				21
OS4702			Not in data set.	-
OS4703			Not in data set.	-

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C:
Experimental Design

C.1 NOLH Experiments

Listing C.1: Configuration File - simulation.ini - OLH experiments

```
[exp_00−0020]
distribution_tag_feedback = [1.0, 0.0, 0]
distribution_tag_feedback_learner = [0.81, 0.18999999999999995, 0]
settings_sim_replications = 2
number_of_tags_per_student_profile = 19
delta = 1.6
number_of_students = 40

[exp_00−0021]
distribution_tag_feedback = [0.25, 0.75, 0]
distribution_tag_feedback_learner = [0.88, 0.12, 0]
settings_sim_replications = 2
number_of_tags_per_student_profile = 28
delta = 0.3
number_of_students = 30

[exp_00−0022]
distribution_tag_feedback = [0.44, 0.56, 0]
distribution_tag_feedback_learner = [0.06, 0.94, 0]
settings_sim_replications = 2
number_of_tags_per_student_profile = 13
delta = 0.6
number_of_students = 55

[exp_00−0023]
distribution_tag_feedback = [0.63, 0.37, 0]
distribution_tag_feedback_learner = [0.31, 0.69, 0]
settings_sim_replications = 2
number_of_tags_per_student_profile = 50
delta = 0.9
number_of_students = 53
```

```
[exp_00−0024]
distribution_tag_feedback = [0.94, 0.06000000000000005, 0]
distribution_tag_feedback_learner = [0.44, 0.56, 0]
settings_sim_replications = 2
number_of_tags_per_student_profile = 6
delta = 3.8
number_of_students = 43

[exp_00−0025]
distribution_tag_feedback = [0.31, 0.69, 0]
distribution_tag_feedback_learner = [0.38, 0.62, 0]
settings_sim_replications = 2
number_of_tags_per_student_profile = 41
delta = 5.0
number_of_students = 33

[exp_00−0026]
distribution_tag_feedback = [0.19, 0.81, 0]
distribution_tag_feedback_learner = [1.0, 0.0, 0]
settings_sim_replications = 2
number_of_tags_per_student_profile = 16
delta = 3.1
number_of_students = 65

[exp_00−0027]
distribution_tag_feedback = [0.88, 0.12, 0]
distribution_tag_feedback_learner = [0.75, 0.25, 0]
settings_sim_replications = 2
number_of_tags_per_student_profile = 47
delta = 2.8
number_of_students = 63

[exp_00−0028]
distribution_tag_feedback = [0.5, 0.5, 0]
distribution_tag_feedback_learner = [0.5, 0.5, 0]
settings_sim_replications = 2
number_of_tags_per_student_profile = 25
delta = 2.5
number_of_students = 50
```

[exp_00−0029]
distribution_tag_feedback = [0.0, 1.0, 0]
distribution_tag_feedback_learner = [0.19, 0.81, 0]
settings_sim_replications = 2
number_of_tags_per_student_profile = 31
delta = 3.4
number_of_students = 60

[exp_00−0030]
distribution_tag_feedback = [0.75, 0.25, 0]
distribution_tag_feedback_learner = [0.13, 0.87, 0]
settings_sim_replications = 2
number_of_tags_per_student_profile = 22
delta = 4.7
number_of_students = 70

[exp_00−0031]
distribution_tag_feedback = [0.56, 0.43999999999999995, 0]
distribution_tag_feedback_learner = [0.94, 0.06000000000000005, 0]
settings_sim_replications = 2
number_of_tags_per_student_profile = 38
delta = 4.4
number_of_students = 45

[exp_00−0032]
distribution_tag_feedback = [0.38, 0.62, 0]
distribution_tag_feedback_learner = [0.69, 0.31000000000000005, 0]
settings_sim_replications = 2
number_of_tags_per_student_profile = 0
delta = 4.1
number_of_students = 48

[exp_00−0033]
distribution_tag_feedback = [0.06, 0.94, 0]
distribution_tag_feedback_learner = [0.56, 0.43999999999999995, 0]
settings_sim_replications = 2
number_of_tags_per_student_profile = 44
delta = 1.3
number_of_students = 58

```
[exp_00−0034]
distribution_tag_feedback = [0.69, 0.31000000000000005, 0]
distribution_tag_feedback_learner = [0.63, 0.37, 0]
settings_sim_replications = 2
number_of_tags_per_student_profile = 9
delta = 0.0
number_of_students = 68

[exp_00−0035]
distribution_tag_feedback = [0.81, 0.18999999999999995, 0]
distribution_tag_feedback_learner = [0.0, 1.0, 0]
settings_sim_replications = 2
number_of_tags_per_student_profile = 34
delta = 1.9
number_of_students = 35

[exp_00−0036]
distribution_tag_feedback = [0.13, 0.87, 0]
distribution_tag_feedback_learner = [0.25, 0.75, 0]
settings_sim_replications = 2
number_of_tags_per_student_profile = 3
delta = 2.2
number_of_students = 38
```

C.2 Distributions of Simulation Output Metrics

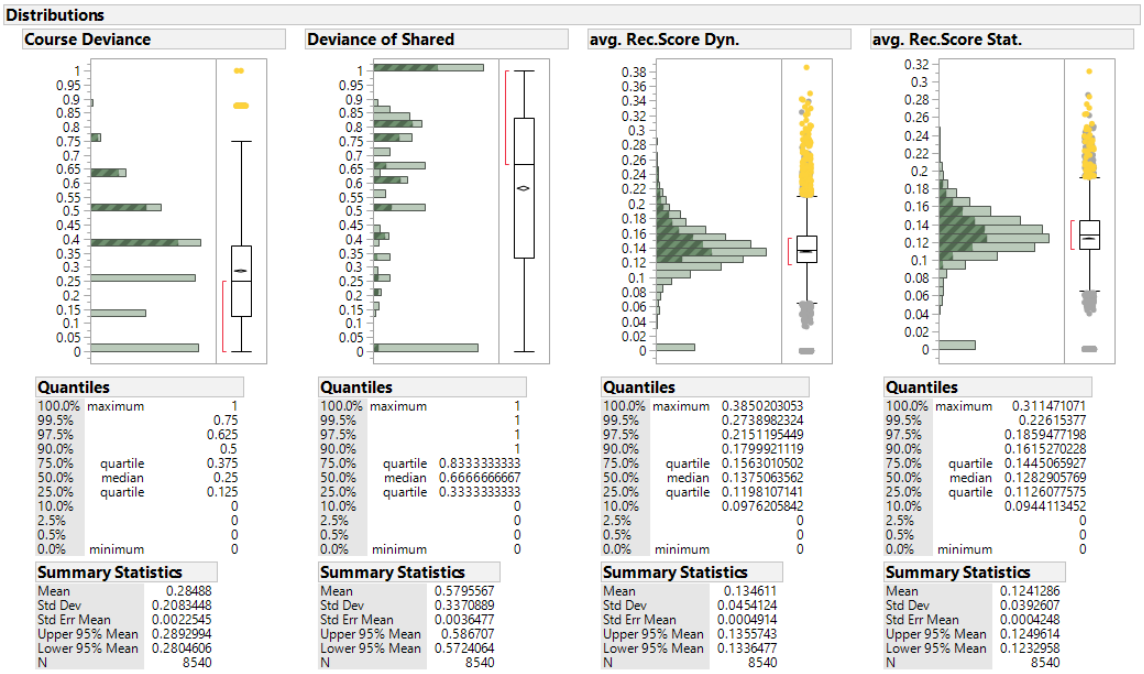


Figure C.1. **Distributions of Simulation Output Metrics**.

THIS PAGE INTENTIONALLY LEFT BLANK

List of References

Andersen PM (2018) Extract of the NPS course catalog database for the academic year 2018. Personal communication.

Barker P (2005) What is IEEE Learning Object Metadata / IMS Learning Resource Metadata? *CETIS Standards Briefing Series, JISC (Joint Information Systems Committee of the Universities' Funding Councils)*.

Barker P, Campbell LM (2010) Metadata for learning materials: An overview of existing standards and current developments. *Technology, Instruction, Cognition and Learning* 7(3-4):225–243.

Bastian M, Heymann S, Jacomy M (2009) Gephi: An open source software for exploring and manipulating networks. http://www.aaai.org/ocs/index.php/ICWSM/09/paper/view/154.

Bostock M, Ogievetsky V, Heer J (2011) D^3 data-driven documents. *IEEE Transactions on Visualization and Computer Graphics* 17(12):2301–2309.

Burke R (2002) Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction* 12(4):331–370, ISSN 1573-1391, URL http://dx.doi.org/10.1023/A:1021240730564.

De Domenico M, Porter MA, Arenas A (2015) Muxviz: A tool for multilayer analysis and visualization of networks. *Journal of Complex Networks* 3(2):159–176, URL http://dx.doi.org/10.1093/comnet/cnu038.

Gera R (2018) URL http://faculty.nps.edu/rgera/MA4404.html.

Herrington A, Herrington J (2008) What is an authentic learning environment? *Online and Distance Learning: Concepts, Methodologies, Tools, and Applications*, 68–77 (Hershey, PA, USA: IGI Global), ISBN 978-1-59904-935-9, URL http://dx.doi.org/10.4018/978-1-59904-935-9.ch008.

Keßler C, d'Aquin M, Dietze S (2013) Linked data for science and education. *Semantic Web* 4(1):1–2, ISSN 1570-0844, URL http://dx.doi.org/10.3233/SW-120091.

Khan Academy (2018) Knowledge map | Khan Academy. https://www.khanacademy.org/exercisedashboard.

Kivelä M, Arenas A, Barthelemy M, Gleeson JP, Moreno Y, Porter MA (2014) Multilayer networks. *Journal of Complex Networks* 2(3):203–271.

Kogleck L (2016) A network of knowledge. *Nature* 531:S102.

Lazarinis F, Green S, Pearson E, eds. (2011) *Handbook of Research on E-Learning Standards and Interoperability: Frameworks and Issues* (IGI Global), ISBN 978-1-61692-789-9 978-1-61692-790-5, URL http://dx.doi.org/10.4018/978-1-61692-789-9.

Metadata (2018) Merriam-Webster. Accessed October 19, 2018. https://www.merriam-webster.com/dictionary/metadata.

Miller SJ (2011) *Metadata for Digital Collections : A How-to-Do-It Manual*. How-to-do-it manuals ; no. 179 (New York: Neal-Schuman Publishers), ISBN 978-1-55570-746-0.

Mueller A (2012) wordcloud python package. https://github.com/amueller/word_cloud.

Naval Postgraduate School (2018) Naval Postgraduate School—SmartCatalog www.academiccatalog.com. http://nps.smartcatalogiq.com/en/Current/Academic-Catalog.

Nesshöver C, Vandewalle M, Wittmer H, Balian EV, Carmen E, Geijzendorffer IR, Görg C, Jongman R, Livoreil B, Santamaria L, Schindler S, Settele J, Sousa Pinto I, Török K, van Dijk J, Watt AD, Young JC, Zulka KP, the KNEU Project Team (2016) The network of knowledge approach: Improving the science and society dialogue on biodiversity and ecosystem services in Europe. *Biodiversity and Conservation* 25(7):1215–1233, ISSN 1572-9710, URL http://dx.doi.org/10.1007/s10531-016-1127-5.

Newman M (2010) *Networks: An introduction* (Oxford, England: Oxford University Press).

Operations Research Department at the Naval Postgraduate School (2016) Academic program review of the Operations Research Department from 2015-2017.

Pane JF, Steiner ED, Baird MD, Hamilton LS, Pane JD (2017) How does personalized learning affect student achievement? https://www.rand.org/pubs/research_briefs/RB9994.html.

PyPDF2 (2016) The pdffilereader class. https://pythonhosted.org/PyPDF2/PdfFileReader.html.

Ramos JE (2003) Using tf-idf to determine word relevance in document queries. *Proceedings of the 1st Instructional Conference on Machine Learning*, volume 242, 133–142.

Russell JS (2018) Personalized learning is key to meeting students' needs in continuing education. URL https://evolllution.com/attracting-students/todays_learner/personalized-learning-is-key-to-meeting-students-needs-in-continuing-education/.

Sanchez SM (2011) NOLHdesigns spreadsheet version 6. *Seed Center for Data Farming. https://harvest. nps. edu* .

Sargent RG (2013) Verification and validation of simulation models. *Journal of Simulation* 7(1):12–24, ISSN 1747-7786, URL http://dx.doi.org/10.1057/jos.2012.20.

Stiles MJ (2000) Effective learning and the virtual learning environment. *Proceedings: EUNIS 2000—Towards Virtual Universities, Instytut Informatyki Politechniki Poznan-skiej*.

US Department of Education (2017) National Education Technology Plan. https://tech.ed.gov/netp/.

Wu D, Lu J, Zhang G (2015) A fuzzy tree matching-based personalized e-learning recommender system. *IEEE Transactions on Fuzzy Systems* 23(6):2412–2426, ISSN 1063-6706, 1941-0034, URL http://dx.doi.org/10.1109/TFUZZ.2015.2426201.

THIS PAGE INTENTIONALLY LEFT BLANK

Initial Distribution List

(1) Defense Technical Information Center
Ft. Belvoir, Virginia

(2) Dudley Knox Library
Naval Postgraduate School
Monterey, California