



Vrije Universiteit Brussel

FACULTY OF SCIENCE
Department of Computer Science

Design Support for Session-based Adaptation in WSDM: Link Recommendations

Graduation thesis submitted to obtain a License Degree In Applied Computer Science

Stijn Coolbrandt

Academic Year 2004 – 2005

Promotor: Prof. Dr. Olga De Troyer
Guiding assistant: Sven Casteleyn





Vrije Universiteit Brussel

FACULTEIT VAN DE WETENSCHAPPEN
Departement Computerwetenschappen

Ontwerp Ondersteuning voor Sessie-gebaseerde Adaptatie in WSDM: Navigatie Suggesties

Verhandeling voorgedragen tot het behalen van de graad van Licentiaat in de Toegepaste Informatica

Stijn Coolbrandt

Academiejaar 2004 – 2005

Promotor: Prof. Dr. Olga De Troyer
Begeleider: Sven Casteleyn



Abstract

Nowadays, web sites often play a key role in the commercial plan of companies. It is therefore crucial, to design them in the best possible way in order to satisfy the needs of their visitors. One of the major causes of visitor dissatisfaction is the overwhelming amount of available information on large web sites. Users get lost in the maze of information, get annoyed and give up their search for information. To cope with this problem, web sites can adapt themselves to the needs of each user separately in order to guide them in their search. More specifically, link recommendations can be provided. These link recommendations are links to pages which could possibly interest the user and are based on information about the user (e.g. his goals, characteristics, preferences, ...). This information is mostly stored in a user model which is maintained for each visitor. This user model however has the disadvantage that it requires many efforts from both the designer and the visitor. The efforts of the designer include finding out which user information is needed, keeping the user model up to date and specifying how this information should be processed. The user, on the other hand, needs to supply feedback such as giving ratings or filling in long forms, and often needs to log in. In this dissertation we propose session-based link recommendations which do not require a user model and therefore avoid the efforts that arise when such a user model has to be constructed and maintained. The link recommendations techniques which generate the link recommendations take the session data of the active visitor as input and analyze the session data of previous visitors to make recommendations for this visitor. Providing personal link recommendations for each visitor therefore results in session-based adaptation which personalizes the web site for each user session.

The framework for this thesis is WSDM, a web site design method which already provides a language (ASL) to specify adaptation strategies. These strategies aim at improving the web site for all visitors and thus not specifically for individual visitors. To be able to specify link recommendation algorithms, WSDM and ASL are extended with design time support for session-based adaptation which aims at tailoring the web site separately for each visitor. Using these extensions, two session-based link recommendation techniques are proposed and specified in ASL. Both techniques are extensively discussed and validated using an experimental web site, designed and implemented using WSDM, on which both algorithms are tested.

Samenvatting

Tegenwoordig spelen web sites vaak een belangrijke rol in het commerciële plan van bedrijven. Daarom is het cruciaal om ze op de best mogelijke manier te ontwerpen met als doel te voldoen aan de noden van hun bezoekers. Een van de belangrijke redenen waarom bezoekers ontevreden raken is de overvloed aan informatie aanwezig op omvangrijke web sites. Gebruikers raken verloren in het web van informatie, raken geërgerd en geven hun zoektocht naar informatie op. Om dit probleem aan te gaan kunnen web sites zichzelf aanpassen aan de noden van elke gebruiker afzonderlijk met als doel hem te helpen in zijn zoektocht. Meer specifiek kunnen navigatie suggesties worden aangeboden. Deze navigatie suggesties zijn links naar pagina's die de gebruiker mogelijk kunnen interesseren en zijn gebaseerd op informatie over de bezoeker (bvb. zijn doelen, karakteristieken, voorkeuren, ...). Deze informatie is meestal opgeslagen in een model van de gebruiker dat wordt bijgehouden voor elke bezoeker. Dit model heeft echter het nadeel dat het veel inspanningen vraagt van de ontwerper en de bezoeker. De inspanningen van de ontwerper zijn onder meer het uitpluizen van welke informatie er moet worden bijgehouden, het model up-to-date houden en het specificeren hoe deze informatie verwerkt moet worden. De gebruiker moet van zijn kant voor feedback zorgen in de vorm van het geven van scores of het invullen van formulieren, en moet vaak ook inloggen. In dit proefschrift stellen we sessie-gebaseerde navigatie suggesties voor die geen model van de gebruiker nodig hebben en daardoor de inspanningen vermijden die ontstaan wanneer zo een model geconstrueerd en onderhouden moet worden. De navigatie suggestie technieken die de navigatie suggesties genereren nemen de sessie data van de actieve bezoeker als input en analyseren de sessie data van eerdere bezoekers om suggesties te maken voor deze bezoeker. Het verschaffen van deze persoonlijke navigatie suggesties voor elke bezoeker resulteert daardoor in sessie-gebaseerde adaptatie die de web site personaliseert voor elke gebruikerssessie.

Het framework voor deze thesis is WSDM, een ontwerp methode voor web sites die reeds een taal (ASL) heeft om adaptatie strategieën te specificeren. Deze strategieën zijn gericht op het verbeteren van de web site voor alle bezoekers en dus niet specifiek voor individuele bezoekers. Om in staat te zijn navigatie suggestie algoritmes te kunnen specificeren, moeten WSDM en ASL uitgebreid worden met ontwerp ondersteuning voor sessie-gebaseerde adaptatie met als doel de web site op elke gebruiker afzonderlijk af te stemmen. Met deze uitbreidingen worden twee sessie-gebaseerde navigatie suggestie technieken voorgesteld en gespecificeerd in ASL. Beide technieken worden uitgebreid besproken en gevalideerd door een experimentele web site, ontworpen en geïmplementeerd door middel van WSDM, waarop beide algoritmes getest worden.

Acknowledgements

I would like to thank my parents who supported me during all my study years. Because of them, I am able to write this dissertation. It would have been so much harder without their support. I am very grateful that they have been putting up with my lack of attention towards them because of all the time I had to spend on writing this dissertation. I would also like to thank my girlfriend Daphne for giving me moral support throughout my study and even more important, for coloring my life with the most beautiful colors.

I also owe many thanks to Sven Casteleyn who was my guiding assistant for this dissertation. He answered my countless questions and gave me lots of valuable advice which was of great help for me. Finally, I would like to thank Tom Vleminckx which whom I implemented the web site needed for the evaluation of my work.

Contents

| | | |
|----------|--|-----------|
| 1 | Introduction | 11 |
| 1.1 | Introduction | 11 |
| 1.2 | Problem Statement | 12 |
| 1.3 | Motivation | 12 |
| 1.4 | Objectives | 13 |
| 1.5 | Thesis Structure | 14 |
| | | |
| I | Background | 16 |
| | | |
| 2 | Web Site Design Method | 17 |
| 2.1 | Introduction | 17 |
| 2.2 | WSDM Overview | 17 |
| 2.2.1 | Mission Statement Specification | 18 |
| 2.2.2 | Audience Modelling | 18 |
| 2.2.3 | Conceptual Design | 19 |
| 2.2.4 | Implementation Design | 20 |
| 2.2.5 | Implementation | 20 |
| 2.3 | Adaptation Specification Language | 20 |
| 2.3.1 | The Navigation Model | 21 |
| 2.3.2 | Operations on the Navigation Model using ASL | 21 |
| 2.3.2.1 | Basic program structure | 22 |
| 2.3.2.2 | Control structures | 23 |
| 2.3.2.3 | Native operations and functions | 23 |
| 2.3.2.4 | Set expressions | 24 |
| 2.3.2.5 | Assignment operator | 24 |
| 2.3.2.6 | Tracking variables and monitors | 25 |
| 2.3.2.7 | Scripts | 26 |
| 2.3.3 | Specifying adaptive behaviour using ASL | 26 |
| | | |
| 3 | Machine Learning Techniques | 28 |
| 3.1 | Introduction | 28 |
| 3.2 | Bayes' Theorem | 29 |
| 3.3 | Naive Bayes Classifier | 30 |
| 3.4 | k-Nearest Neighbour Classifier | 31 |

| | | |
|-----------|---|-----------|
| 4 | Adaptive Behaviour for Web Sites | 33 |
| 4.1 | Introduction | 33 |
| 4.2 | Overview of Existing Adaptation Methods | 34 |
| 4.2.1 | Adaptation at The Presentation level | 36 |
| 4.2.1.1 | Adaptive multimedia presentation | 36 |
| 4.2.1.2 | Adaptive text presentation | 36 |
| 4.2.1.3 | Adaptation of modality | 38 |
| 4.2.2 | Adaptation at the Navigation Level | 38 |
| 4.2.2.1 | Direct guidance | 39 |
| 4.2.2.2 | Adaptive link sorting | 39 |
| 4.2.2.3 | Adaptive link hiding | 39 |
| 4.2.2.4 | Adaptive link annotation | 40 |
| 4.2.2.5 | Adaptive link generation | 40 |
| 4.2.2.6 | Map adaptation | 41 |
| 4.2.3 | Optimization Techniques | 41 |
| 4.2.3.1 | Promotion and demotion | 41 |
| 4.2.3.2 | Highlighting | 41 |
| 4.2.3.3 | Linking | 42 |
| 4.2.3.4 | Clustering | 42 |
| 4.3 | Recommendation Systems | 42 |
| 4.3.1 | Introduction | 42 |
| 4.3.2 | Characteristics of Recommendation Systems | 43 |
| 4.3.3 | Link Recommendations Techniques | 44 |
| 4.3.3.1 | Content-based recommendation techniques | 46 |
| 4.3.3.2 | Collaborative recommendation techniques | 49 |
| 4.3.3.3 | Hybrid recommendation techniques | 53 |
| 4.3.4 | Example of a Recommendation System: Amazon.com | 54 |
| 4.3.4.1 | Amazon.com's user models | 54 |
| 4.3.4.2 | How Amazon.com recommends items | 55 |
| 4.3.4.3 | Amazon.com's recommendation technique | 56 |
| II | Research | 58 |
| 5 | Integrating Link Recommendations into WSDM | 59 |
| 5.1 | Introduction | 59 |
| 5.2 | Extensions to WSDM | 61 |
| 5.2.1 | The Navigation Model | 61 |
| 5.2.2 | ASL | 62 |
| 5.2.2.1 | Extension 1: new native operations | 63 |
| 5.2.2.2 | Extension 2: new native functions | 64 |
| 5.2.2.3 | Extension 3: a new native set | 65 |
| 5.2.2.4 | Extension 4: a new set creator | 65 |
| 5.2.2.5 | Extension 5: revising the monitor statement | 65 |
| 5.3 | Proposed Recommendation Techniques | 66 |
| 5.3.1 | Link recommendations using a KNN algorithm | 66 |
| 5.3.1.1 | Introduction | 66 |
| 5.3.1.2 | Using the KNN algorithm to determine interest- ing pages | 67 |
| 5.3.1.3 | The algorithm expressed in ASL | 72 |

| | | |
|----------|--|------------|
| 5.3.1.4 | Analysis and extensions | 77 |
| 5.3.2 | Link Recommendations using Bayes' Theorem | 81 |
| 5.3.2.1 | Introduction | 81 |
| 5.3.2.2 | Using Bayes' Theorem to calculate page view probabilities | 82 |
| 5.3.2.3 | The algorithm expressed in ASL | 86 |
| 5.3.2.4 | Analysis and extensions | 90 |
| 6 | Implementation | 93 |
| 6.1 | Introduction | 93 |
| 6.2 | Designing a Web Site using WSDM | 94 |
| 6.2.1 | Introduction | 94 |
| 6.2.2 | Mission Statement Specification | 94 |
| 6.2.3 | Audience Modelling | 95 |
| 6.2.4 | Conceptual Design | 95 |
| 6.2.5 | Implementation Design | 97 |
| 6.2.6 | Implementation | 98 |
| 6.3 | Implementing the WSDM Web Site | 98 |
| 6.3.1 | Introduction | 98 |
| 6.3.2 | Annotation | 99 |
| 6.3.3 | Graph structure implementation | 100 |
| 6.3.4 | Running the web site | 103 |
| 6.4 | ASL Interpreter Implementation | 104 |
| 6.5 | Integration | 105 |
| 7 | Evaluation of the Algorithms | 107 |
| 7.1 | Introduction | 107 |
| 7.2 | Performed Experiments | 107 |
| 7.3 | Results of the Experiments | 109 |
| 7.4 | Interpretation of the Results | 110 |
| 8 | Conclusions | 113 |
| 8.1 | Introduction | 113 |
| 8.2 | Summary | 113 |
| 8.3 | Achievements | 114 |
| 8.4 | Future Work | 114 |
| A | ASL BNF Specification | 116 |

List of Tables

| | | |
|-----|---|-----|
| 2.1 | The native operations of ASL. | 23 |
| 2.2 | Some native functions of ASL. | 24 |
| 2.3 | The set creators and set operators in ASL. | 24 |
| 5.1 | The new native operations added to ASL. | 64 |
| 5.2 | An example of a simple session matrix. | 67 |
| 5.3 | The session data for the web site of the “The Recommenders” . . | 70 |
| 5.4 | The session distances to session 9. | 71 |
| 5.5 | The session data extended with page popularities for $k=4$ | 72 |
| 5.6 | The session data extended with page popularities for $k=3$ | 72 |
| 5.7 | The session data extended with posterior probabilities. | 86 |
| 7.1 | The number of clicks needed for each experiment. | 110 |
| 7.2 | The time needed for each experiment. | 110 |
| 7.3 | The results of the t -test for the number of clicks (exp. 1 vs. 2). . | 111 |
| 7.4 | The results of the t -test for the amount of time (exp. 1 vs. 2).. | 111 |
| 7.5 | The results of the t -test for the number of clicks (exp. 2 vs. 3). . | 112 |
| 7.6 | The results of the t -test for the amount of time (exp. 2 vs. 3). . | 112 |

List of Figures

| | | |
|-----|---|-----|
| 2.1 | An overview of WSDM. | 18 |
| 2.2 | Example of a simplified Navigation Model. | 22 |
| 4.1 | Taxonomy of adaptive hypermedia technologies. | 35 |
| 4.2 | A snapshot of the recommendation system of Amazon.com | 56 |
| 5.1 | An example of an extended Navigation Model. | 63 |
| 5.2 | Map of the web site of “The Recommenders”. | 70 |
| 6.1 | The task model for retrieving a publication. | 96 |
| 6.2 | The publication object chunk. | 97 |
| 6.3 | The member object chunk. | 100 |
| 6.4 | The annotation/transformation process for the member chunk. . | 101 |
| 6.5 | The design of the graph structure. | 102 |
| 6.6 | The homepage of the WISE web site. | 104 |
| 6.7 | The integration of all components. | 106 |
| 7.1 | A scatter plot of the experiments’ data. | 109 |

Chapter 1

Introduction

1.1 Introduction

Since the birth of the World Wide Web in 1990, its usage has increased from 0.4 % in 1995 to 13,9 % of the world population at the moment of writing¹. Along with this enormous growth of users, millions of web sites have been created which all serve information or services upon request of these users. The numbers mentioned above show that qualitative web design can be beneficial for almost a billion of people and can thus be considered as being a very hot topic in the world of hypermedia. Besides being a hot topic it is also a juvenile topic which is waiting to be further explored and enriched by researchers.

Many professional businesses nowadays depend heavily on their web sites. Users can buy products, hire services or simply find information via the World Wide Web. The goal of almost every web site is to keep the user satisfied and interested. This is, however, a very difficult task considering the information overload which is available on most web sites. One of the main reasons why users get annoyed and leave a web site, is because of not finding the information they are looking for. Even if they are satisfied, it can be advantageous to identify and recommend pages that could interest the user. For businesses exploiting professional web sites, for example, this can increase their revenues because users will buy recommended products which they would not have bought otherwise.

This recommendation of interesting pages or even products is already widely used on the Internet. When buying a book on Amazon.com, for example, customers get recommendations of other books or products which are commonly bought by customers who bought the same book. Whereas, in daily life, people tend to get recommendations by friends which are equally minded, the Internet gives the opportunity to provide recommendations by some kind of collective intelligence of unknown people which behave alike in some particular way. This is what makes the use of recommendations in web sites a very interesting study subject.

¹Internet world stats: <http://www.internetworldstats.com>

1.2 Problem Statement

As stated in the introduction, it is crucial to have some techniques which identify pages which are possibly interesting but unknown or unavailable for the user in his current browsing context. We will call the identification and recommendation of links to possible interesting pages for web site users *link recommendations*. These link recommendations are the main subject of this thesis. The kind of link recommendations that are considered in this dissertation are recommendations generated at runtime. Because of this runtime generation which is dependent on the current user, we can speak of user dependent web site adaptation. We will integrate this specific web site adaptation method into *WSDM*, which stands for Web Site Design Method. This is an approach for designing web sites developed by the Web & Information System Engineering (WISE) research group of the Vrije Universiteit Brussel.

In WSDM the focus of the design is not on the data of the organization but on the audience of the web site. This audience is split into different audience classes which each have their own characteristics and requirements. Using this information for every audience class, the needed data to fulfil these requirements is retrieved and a conceptual model is build. We will discuss WSDM in detail in the chapter 2.

The purpose of this thesis is to integrate link recommendations into WSDM so that web sites designed using WSDM can support this type of adaptation which is a valuable added feature for the web sites and its users. There are many different kinds of link recommendations. The one that will be integrated into WSDM will be based on the browsing behaviour of the users and will not require any additional tasks to be performed by the users. We will look a bit closer at the objectives and the motivation for the integration in the next sections.

1.3 Motivation

The importance of link recommendations was already mentioned in the introduction. They can be an important extra help for the users to find pages that they might not have found by themselves and can be a boost for electronic commerce web sites.

Note that there is a relation between the size of the web site and the need for link recommendations. When a web site is relatively small so that each piece of interesting information is only a few clicks away, link recommendations may seem unnecessary. For large data intensive web sites, where the web site seems to be never ending, link recommendations become much more appropriate. When there are thousands of pages of information about a specific subject, the user would not know where to start and would give up his quest for information. If, instead, the user would be offered some interesting recommendations as starting points he would be guided in the good direction.

It's obvious that the recommendation system stands or falls with the technique it uses to recommend links. When recommended links appear not to be as

interesting as was estimated for the user, they are totally needless and can even have a negative impact on the satisfaction of the users. Hence it's crucial to develop a system which imposes high demands and requirements on the quality of these recommended links.

Although providing link recommendations can be considered as an individual feature which is not dependent of a web site design method, any web design method supporting this adaptation method at design time certainly has an advantage over web design methods that do not support it. The integration of recommendations into WSDM will show the advantages of WSDM over other web design methods which do not provide the design support for link recommendations and other web site adaptation methods.

The biggest difference with existing link recommendations systems is that the link recommendations can be specified at design time. Existing link recommendation systems add the link recommendations at the implementation level and are not reusable as they are designed and implemented for a specific web site. The link recommendation algorithms that will be introduced in this dissertation can be reused by each web designer that uses WSDM and the algorithms to compute the link recommendations can be easily altered.

1.4 Objectives

The main objective of this thesis is to incorporate link recommendations into WSDM in such a way that they can be easily enabled or disabled by the web designer. We will explore the field of adaptive web sites and more specifically the already existing techniques for link recommendations. Furthermore, two techniques of computing link recommendations will be presented and evaluated. There are many ways of computing link recommendations, based on the contents, user ratings or just browsing behaviour of other users. The techniques that we will elaborate will be based on the browsing behaviour of the users and should not require a user model as this would be not consistent with the philosophy of adaptation in WSDM which is based on user access information.

The two techniques are each based on two well known techniques in the field of machine learning: Bayes' Theorem and the k-Nearest Neighbour algorithm. These techniques are already frequently used on the level of content-based link recommendations but rarely on the level of user-based link recommendations. We will implement these algorithms and evaluate their performance and impact in the last sections of this dissertation.

WSDM already supports adaptive behaviour to rearrange and re-organize the content and/or structure of the web site. This adaptive behaviour can be specified at design time using a dedicated language called *ASL* which is the abbreviation for Adaptation Specification Language. The current version of ASL already supports web site adaptation which aims at accommodating all visitors, also known as optimization. An example of this kind of adaptation is linking popular pages directly from the homepage of the web site. This kind of adaptation is thus visible to all visitors of the web site.

The adaptation supported by ASL is in contrast to the kind of adaptation which is the goal of this dissertation: generating visitor-specific links which are only visible to the visitor for which the links are generated. We will use ASL and adapt it to be able to allow dynamical adaptation at session level. This way information of past sessions can be used to recommend interesting links to new visitors. The session information will be used in the two link recommendation algorithms and can be used in all other link recommendation algorithms which are based on session data of previous visitors. Besides additions to the ASL language, WSDM will be extended at the design level so that link recommendations can be specified during the conceptual design phase. Before we delve into the world of WSDM and adaptive behaviour, we will first give an overview of the structure of this thesis.

1.5 Thesis Structure

This thesis is structured into 8 chapters including this short introduction. The following chapters are divided into a part covering all the background information (chapters 2, 3 and 4) and a research part (chapters 5, 6, 7 and 8). These chapters can be summarized as follows.

Chapter 2 covers the necessary background information about WSDM, the design method in which the link recommendation techniques will be integrated. A brief overview of all the design phases of WSDM is given and the syntax and semantics of ASL is introduced. It also shows how ASL can be used to specify at design time, possible adaptive behaviour at runtime. This chapter should be read by readers which are not familiar with WSDM and its adaptation language.

Chapter 3 introduces three machine learning techniques needed in the subsequent chapters: Bayes' Theorem, the Naive Bayes Classifier and the k-Nearest Neighbour Classifier. Readers which already know these techniques can skip this chapter.

Chapter 4 gives an overview of the existing web site adaptation methods with a special focus on the existing recommendation systems. The first part of this chapter discusses other adaptation methods and situates link recommendations in the context of web site adaptation methods. The reader can skip this part as all of these adaptation methods, except for the link recommendations, do not return in the subsequent chapters of this dissertation. The remainder of this section explores the existing recommendation systems and their link recommendation techniques. It's advisable to read this part in order to get familiar with all aspects of link recommendation techniques.

Chapter 5 is the core chapter of this thesis as it discusses the link recommendation techniques and their integration into WSDM. First, the extensions to WSDM needed for supporting the link recommendation techniques are discussed. Using these extensions, two link recommendation techniques are proposed and described in ASL.

Chapter 6 takes the theoretical discussion of chapter 5 to the practical level and describes the implementation of a WSDM web site. This includes an overview of the design process for the target web site, an overview of the actual implementation and a discussion of the implementation of an ASL interpreter.

Chapter 7 evaluates the proposed link recommendation algorithms, more specifically their impact on the browsing behaviour of the visitors. This evaluation is based on tests with a WSDM web site which provides link recommendations.

Chapter 8 makes a round-up of the achievements and discusses further extensions.

Part I

Background

Chapter 2

Web Site Design Method

2.1 Introduction

In this chapter, an overview of WSDM [15, 18, 26, 27, 28, 29, 48] is given. This is an audience driven web site design method which was already briefly mentioned in chapter 1. WSDM is still under development at the WISE research group of the Vrije Universiteit Brussel. The last years, support for adaptive behaviour has been added to WSDM. For the incorporation of this adaptive behaviour into WSDM ASL was created. ASL is a language that enables the designer to formulate adaptive behaviour at design time [16, 17, 18]. In the second part of this chapter we will let our light shine on this language and see how it can be used to specify adaptive behaviour.

2.2 WSDM Overview

The Web Site Design Method is audience driven which means that it takes the potential visitors of the web site as a starting point for the design. This in contrast to data driven design methods such as WebML [19]. The weak point of these methods is that they tend not to pay enough attention to the requirements of the visitors. WSDM tries to overcome the shortcomings of data driven design methods by dividing the visitors into audience classes in order to better tailor the website to the particular needs of the different kind of visitors, and thus increase the overall usability of the web site.

WSDM consists of 5 phases which will be described in detail below. Some of these phases contain sub-phases as can be seen on figure 2.1 which visualizes the overview of the different phases of WSDM. The 5 main phases are the Mission Statement Specification phase, the Audience Modeling phase, the Conceptual Design phase, the Implementation Design phase and the Implementation phase. Let's take a look at each of these phases and their sub-phases in the following sections.

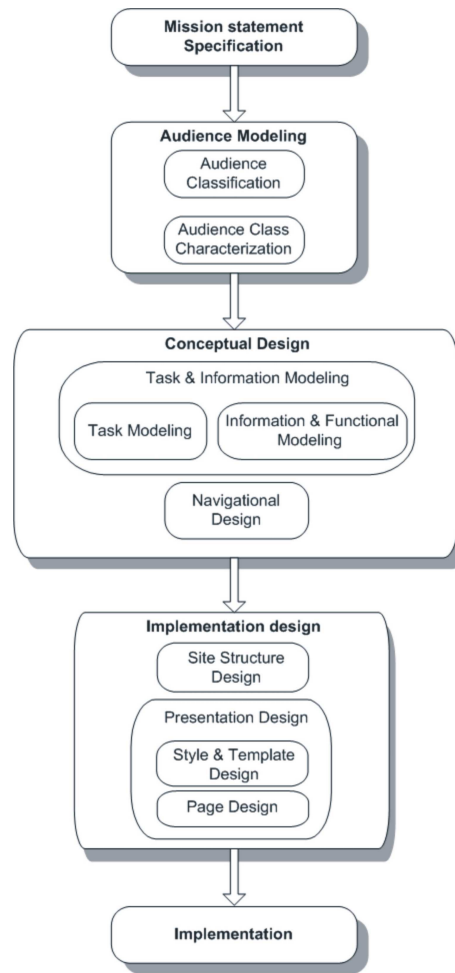


Figure 2.1: An overview of WSDM.

2.2.1 Mission Statement Specification

The *Mission Statement Specification* phase is a very short but nonetheless crucial phase that should be well-thought out. In the Mission Statement Specification phase the subject and the purpose of the web site should be identified. The last step of this phase is to declare the target audience. The output of this phase will serve as input for the Audience Modelling phase.

2.2.2 Audience Modelling

The second phase is the *Audience Modelling* phase which is totally focused on the potential visitors. This phase has two sub-phases: the Audience Classification and the Audience Class Characterization phase. The *Audience Classification* phase identifies and classifies the different types of visitors of the web site. This results in audience classes where each user belonging to the same audience class has the same informational and functional requirements. These audience classes

are then characterized in the *Audience Class Characterization* phase. Examples of such characteristics are the language, knowledge level and age of the audience class.

The *Audience Modelling* phase therefore returns a set of audience classes for which their functional- and informational requirements, their navigational- and usability requirements and their characteristics are provided. Note that audience classes can be structured as a hierarchy with one super audience class called visitor which is then subclassed. These subclasses inherit the same requirements and extend them with extra requirements. The characteristics are however not inherited by the subclasses. Subclasses can on their turn be subclassed too if they are not specific enough.

2.2.3 Conceptual Design

When the audience classes and their requirements and characteristics are identified, we can start with the conceptual design which is free from any implementation details. The *Conceptual Design* phase also has two sub-phases: the Task & Information Modeling phase and the Navigation Design phase.

The *Task & Information Modeling* phase can be divided into two phases. First, during the *Task Modeling* phase, a *Task Model* is defined for each requirement of each audience class. Every Task model is then further elaborated into elementary tasks. The goal of the second sub-phase, the *Information and Functional Modeling* phase, is then to define a data model or *chunk* for each of these elementary tasks. This chunk models the information and functionalities required to fulfil the elementary task. To conclude this phase, the chunks need to be integrated into a single model called the *Business Information Model*.

The second sub-phase is the *Navigation Design* phase in which we design the conceptual structure of the web site. This structure defines how the users of the different audience classes will navigate through the web site. It consists out of nodes that group chunks, which were identified in the previous sub-phase, together. These nodes are then connected by means of *links* which form the navigation paths between the nodes. WSDM distinguishes four types of links:

- **Structural links:** links which provide the actual conceptual structure of the information and functionality available the web site.
- **Semantic links:** links which represent existing semantical relationships between concepts represented by the nodes. An example of a semantical link is the semantical relationship between a movie and its director, which results in a semantic link between both concepts.
- **Navigation aid links:** links which are added on top of the conceptual structure to ease the navigation through the web site and to enhance the usability of the web site. The home link which appears on most web sites is a good example of a navigation aid link.
- **Process logic links:** links which express the workflow between two or more nodes. Buying products online often requires multiple steps such

as choosing the products, filling in personal information and selecting the payment method. The nodes contain this information are connected together by process logic links.

Using links between nodes, a navigation track for each audience class is created. All these navigation tracks together form the *Navigation Model*. The notion of having a navigation track for each audience class is in fact exactly that what makes WSDM audience driven. This in contrast to the Navigation Models of the data driven design methods which are purely based on the available data.

2.2.4 Implementation Design

The next phase is the *Implementation Design* phase which adds implementation related aspects to the conceptual design. It consists of two sub-phases: the Site Structure Design phase and the Presentation Design phase which also consists of two sub-phases.

In the *Site Structure Design* phase we decide how the nodes which are defined in the Navigation Design phase are grouped into pages. It is thus possible to have more than one node per physical page, nodes are not the same as pages. It's up to the designer to decide whether a page contains one node or multiple ones.

The *Presentation Design* phase defines the layout for all the pages on the web site. It consists of two sub-phases: the Page Design phase and the Template & Style phase. The goal in the *Template & Style* sub-phase is to create templates which specify the layout of the pages for all the types of pages in the web site. During the *Page Design* sub-phase, we describe how the chunks on a page should be positioned and how the page layout should be.

2.2.5 Implementation

The *Implementation* phase is the last phase of WSDM in which the actual implementation of the web site takes place. This implementation is based on the previous phases and the implementation environment is totally in the hands of the designer. The web site can be generated automatically using a pipeline transformation process which takes the chunks, the Navigation Model, the site structure design, the template design and the page design as input and outputs the web site in the desired implementation language.

In chapter 6, we will view an example of a web site which is designed using WSDM. This example will show how WSDM is used in practice.

2.3 Adaptation Specification Language

The Adaptation Specification Language (ASL) is designed to let the designer specify at design time which adaptive behaviour will be allowed at runtime. ASL is a high level rule specification language which provides the designer with an easy formalism to specify complex adaptation strategies on a WSDM web site at design time. We will discuss ASL in the context of WSDM for which it

is developed.

The language is event-based: it specifies which conditions should be satisfied and which actions should be performed when these conditions are met. The actions which are then performed can only influence the Navigation Model of the web site which was already discussed in section 2.2.3. Some examples of ASL operations are adding a link between two nodes or disconnecting a chunk from a node.

First, we will discuss the Navigation Model in section 2.3.1 because it is essential to fully grasp it in order to understand the semantics of ASL. We will then take a closer look at the syntax and semantics of ASL in section 2.3.2 and give some examples of how it can be used in practice in section 2.3.3.

2.3.1 The Navigation Model

Recall from section 2.2.3 that the Navigation Model is composed of a set of nodes which are connected by links. Information and/or functionality can be added to nodes by connecting chunks to the nodes. Note that a connection between a node and a chunk is called a *connection* and is not the same as a link. Hence, users can browse through the web site using the links between nodes and can view the information encapsulated in the chunks connected to these nodes.

Figure 2.2 shows an example of a simplified Navigation Model for the WISE web site which is designed using WSDM¹. The ellipses denote chunks, the rectangles nodes. The arrows between the nodes are the (structural) links, the lines between the nodes and the chunks are the connections. The rectangles which are double lined are the root nodes of their audience track (the navigation track of an audience class).

In this example there is a visitor audience track which represents the audience track for the super audience class *Visitor*. For this audience class, there are two nodes available which means that the information (in the form of chunks) belonging to the nodes is considered as being relevant for all visitors. There are also two subclasses of the *Visitor* audience class: the *Researcher* audience class and the *Student* audience class. They each have their own navigation track which was identified by the designer during the Conceptual Design phase. Not to overload the figure, the elaboration of the *Researcher* and *Student* audience track has been omitted. Now that we have a more profound view of the Navigation Model we can go to the main issue of this section which is ASL.

2.3.2 Operations on the Navigation Model using ASL

In this section we discuss the syntax and semantics of ASL. ASL gives the designer the chance to perform runtime operations on the Navigation Model which is defined at the conceptual level. We will give a summary of the operations, functions and control structures that are included in ASL². This overview is

¹The web site of the WISE research group of the Vrije Universiteit Brussel is available at <http://wise.vub.ac.be>.

²Note that ASL is currently still under development and is therefore still subject to changes.

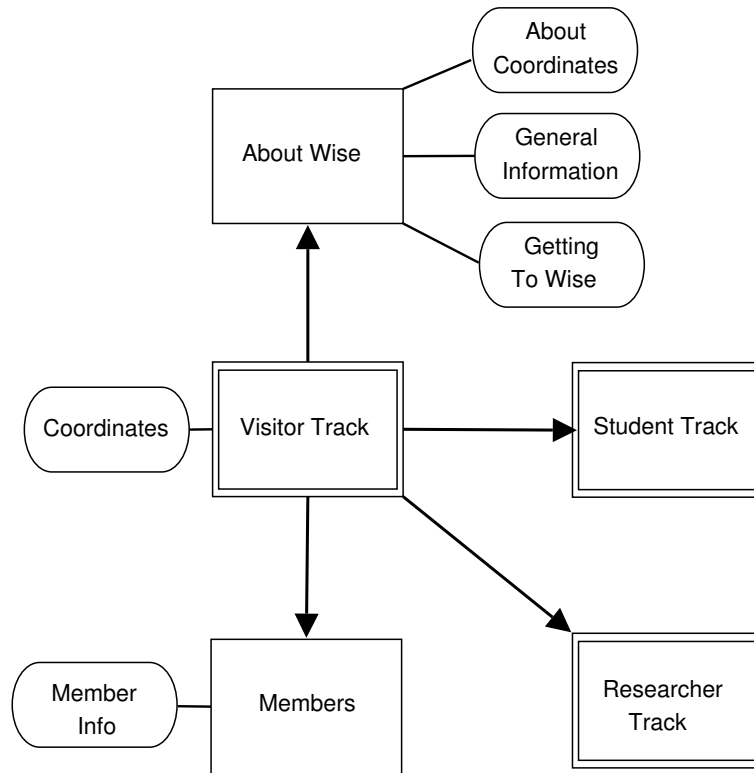


Figure 2.2: Example of a simplified Navigation Model.

however not exhaustive but only includes the most relevant operations which show how adaptive behaviour can be achieved and which we will need in chapter 5 of this dissertation. The operations that are omitted include logical, arithmetic and some statistical operators which are included in most programming languages but which are not relevant for this dissertation. Appendix A provides a complete overview of the syntax ASL in BNF notation which stands for Backus Naur Form [4].

2.3.2.1 Basic program structure

Each ASL program is considered as an adaptation policy. Such a policy consists of one or more script specifications (see section 2.3.2.7) followed by one or more adaptation specifications. An adaptation specification specifies an adaptation strategy which should be executed whenever it is triggered. This trigger is based on some kind of condition which should be met (e.g. a certain time or a user event). An adaptation specification can be specified as follows:

when <trigger> **do** <adaptationStrategy>

The adaptation strategy can be specified by either a rule or a rule sequence. A rule is either a simple rule or a *forEach* construct (see section 2.3.2.2). A simple rule is exactly what it says and is only one rule (such as an operation, a

monitor statement or an *if* statement). A rule sequence is a sequence of rules and should be enclosed with a *begin* and *end* statement as follows:

```
begin
<rule>*
end
```

2.3.2.2 Control structures

ASL has two control structures: the *forEach* construct and the *if* statement. The *forEach* control structure can be solely used to iterate over a set. The syntax of the *forEach* construct is as follows

```
forEach e in set:
  <apply adaptation strategy>
```

The *if* statement is probably the most used control structure ever and can not be missed in any programming language. The syntax of the *if* statement is

```
if condition
then <apply adaptation strategy>
```

The condition should return a truth value which triggers the body of the *if* statement when true. Now that we've seen the basic program and control structures of ASL, we can take a look at its more complex features.

2.3.2.3 Native operations and functions

ASL provides several native operations and functions. The native operations are elementary operations which means that they only exist out of one single operation. These operations manipulate either nodes, chunks or links. Nodes can be added and deleted, chunks can be connected and disconnected from nodes, and links can be also added and deleted. Aside from the two add operations which create a new node and link respectively, all other operations take existing nodes, chunks or links as input. Note also that there are no operations to add or delete chunks, this is because chunks contain bits of information or functionality created by the designer during the Conceptual Design phase. Deleting chunks at runtime would throw away this information or functionality. This is not the case for nodes, which serve to group information, and can thus be added or deleted in order to re-group information. An overview of the native operations is given in table 2.1.

| Operation Semantics | Syntax |
|---|--|
| Adding a node n | <code>addNode(n)</code> |
| Deleting a node n | <code>deleteNode(n)</code> |
| Connecting a chunk c to a node n | <code>connectChunk(h, n)</code> |
| Disconnecting a chunk c from a node n | <code>disconnectChunk(h, n)</code> |
| Adding a link of type t from node n_1 to node n_2 | <code>addLink(t, n_1, n_2)</code> |
| Deleting a link of type t from node n_1 to node n_2 | <code>deleteLink(t, n_1, n_2)</code> |

Table 2.1: The native operations of ASL.

The native functions provided by ASL return values, in contrast to the native operations which only cause a side-effect. Some of these native functions of ASL can be seen in table 2.2.

| Operation Semantics | Syntax |
|--|-------------------------------------|
| Returns the audience class of a node n | <code>audienceClass(n)</code> |
| Returns the audience class of the current user | <code>currentAudienceClass()</code> |
| Returns the source node of link l | <code>source(l)</code> |
| Returns the target node of link l | <code>target(l)</code> |

Table 2.2: Some native functions of ASL.

2.3.2.4 Set expressions

ASL makes use of sets to collect elements like nodes and chunks. To iterate over these sets, ASL provides a set iterator. There are also set comparators and statistical operators which we will not discuss here as they will not be used in this dissertation. Table 2.3 contains the basic native sets and set creators which all return a set.

| Operation Semantics | Syntax |
|---|--|
| Returns all nodes in the web site | <code>Nodes</code> |
| Returns all chunks in the web site | <code>Chunks</code> |
| Returns all links in the web site | <code>Links</code> |
| Returns all connections in the web site | <code>Connections</code> |
| Returns all pages in the web site | <code>Pages</code> |
| Returns all audience classes in the web site | <code>AudienceClasses</code> |
| Returns all nodes in the audience track ac | <code>nodesInAudienceTrack(ac)</code> |
| Returns all chunks in the audience track ac | <code>chunksInAudienceTrack(ac)</code> |
| Returns all chunks connected to node n | <code>chunksFromNode(n)</code> |
| Returns all nodes linked to node n | <code>nodesLinkedTo(n)</code> |
| Returns all nodes linked from node n | <code>nodesLinkedFrom(n)</code> |
| Returns all nodes connected from chunk c | <code>nodesFromChunk(c)</code> |

Table 2.3: The set creators and set operators in ASL.

To iterate over a set, the *forEach* loop introduced in section 2.3.2.2 should be used. In section 2.3.3 which shows some simple applications of ASL, we will see how this construct can be used.

2.3.2.5 Assignment operator

An assignment operator is used in almost every programming language and makes it possible to store values into variables and to retrieve these values later on. The syntax of the assignment operator is straightforward:

`variable := expression`

ASL also provides the possibility to store new added elements into a variable using the following syntax:


```
addLink(linkType, node1, node2) as newLink
```

This puts the newly created link into the *newLink* variable and therefore has a similar effect as an assignment operation.

2.3.2.6 Tracking variables and monitors

ASL also provides tracking variables. A tracking variable is a variable that can be attached to elements of the design like nodes, chunks and links. To attach a tracking variable to each node in the web site for example, the following code should be executed:

```
forEach node in Nodes:
    addTrackingVariable node.amountOfAccesses
```

The lifetime of a tracking variable is the same as the lifetime of the web site. Once a monitor is constructed, it can not be deleted unless the element to which it is attached is deleted. Now that we have defined a tracking variable for each node of a web site, let's see how can it be used to store the number of accesses to the node.

Tracking variables can be updated by monitor statements. Monitors are used to specify which events on which design elements require tracking of their use. This tracking then results in updating tracking variables whenever the events occur. Optionally, a condition can be specified which triggers the update process when it is met. There are four kind of events integrated in ASL:

1. A *click* event which occurs when a link is activated, i.e. when it is followed by a visitor.
2. A *load* event which occurs when a node is loaded, i.e. when the page containing the node is loaded.
3. A *sessionStart* event which occurs at the start of a new user session.
4. A *sessionEnd* event which occurs at the end of a new user session.

Instead of just listing the syntax like in the previous sections, an example of a monitor statement is given below. In this example, the tracking variable *amountOfAccesses* defined above is increased whenever a node is loaded.

```
forEach node in Nodes:
    begin
        monitor load on node
        do node.amountOfAccesses := node.amountOfAccesses + 1
    end
```

ASL also provides a mechanism to sort sets based on the value of a monitor. Using this sorting construct we can, for example, sort the *Nodes* set according to the number of accesses as follows:

```
Nodes[SORT on element : element.amountOfAccesses];
```

This sorting is standard in ascending order, i.e. from low to high values.

2.3.2.7 Scripts

Scripts are a mechanism to define an operation once and to reuse it several times. This resembles much to macros available in high level programming languages. A script has a name and an arbitrary number of parameters. The following code is an example of a script:

```
script addTwoWayLink(linkType, node1, node2):
  begin
    addLink(linkType, node1, node2);
    addLink(linkType, node2, node1);
  end
```

This script makes *node1* reachable from *node2* and vice versa. The script can easily be called as follows:

```
call addTwoWayLink(linkType, node1, node2)
```

2.3.3 Specifying adaptive behaviour using ASL

Now that we have seen the syntax and semantics of ASL, a few simple applications which show how ASL can be used to specify adaptive behaviour at design time will be given. An example application is to discard nodes which are not visited at all. We will use the *amountOfAccesses* tracking variable that was defined in the previous section. The code that should be included to discard unvisited pages can be seen below:

```
forEach node in Nodes:
  if node.amountOfAccesses == 0
    then deleteNode(node)
```

Note that this particular adaptive behaviour introduces a problem. When a script containing this ASL code is executed each hour after launching the web site, many nodes will be deleted because initially no nodes are visited. An easy solution is to manually run the script. This way the designer can wait until the site has been visited enough. A better solution is to let the designer specify at design time when it should be executed. This can be done by tracking the total number of sessions and then setting a threshold for the activation of the code. We will use a monitor which tracks the total number of sessions on the web site as follows:

```
begin
  addTrackingVariable website.amountOfSessions;
  monitor load on website
  do website.amountOfSessions := website.amountOfSessions + 1
end
```

Notice how a new element, called *website*, is introduced. First of all, a tracking variable is attached to the *website* element. The monitor then updates the tracking variable whenever the web site is loaded, i.e. whenever a new visitor session starts. We can now use this new tracking variable and compare it with a threshold value to check if the code can be executed.

```
if website.amountOfSessions > threshold
  forEach node in Nodes:
    if node.amountOfAccesses == 0
      then deleteNode(node)
```

It is clear that it is easy to specify this form of adaptive behaviour as it takes only a few lines of ASL code. Instead of deleting a node we can also decide to link nodes which are visited “often” to the root node of the audience track where the nodes are part of. In the code below, some new syntax is introduced. The *audienceTrack* operation takes a node or chunk as input and returns the audience track to which it belongs. The *root* operation then returns the root node of that audience track.

```
forEach node in Nodes:
  if node.amountOfAccesses > 100000
    then addLink(structural, root(audienceTrack(node)), node)
```

Of course much more complex operations can be specified in ASL. This will be shown chapter 5 where 2 complex algorithms will be specified in ASL.

Chapter 3

Machine Learning Techniques

3.1 Introduction

The background information in this chapter is situated in the field of statistics and machine learning. We will introduce some well known concepts and algorithms used in these two fields which we will need for the rest of this dissertation. It's not the intention to cover the whole field but only the techniques which are relevant for this thesis.

We will take a look at two easy to implement and frequently used machine learning algorithms which are mainly employed for classification: the *Naive Bayes Classifier* and the *k-Nearest Neighbour Classifier*. These algorithms are often used by link recommendation techniques to classify pages or other web items like products (either based on text or on access logs as we will see in chapter 4) as being interesting or not for the user. To be able to classify new pages, a training set of already classified pages is needed. Based on this training set new unseen pages can be classified using a classification algorithm. It's evident that the representability of the training set is crucial and therefore a solid training set is needed. We will show in detail how these two classification algorithms can be used in practice to classify web pages or documents into pre-defined classes in chapter 4.

There are many other classification methods such as decision trees [41], Support Vector Machines [23] and neural networks [41] which we will not discuss extensively here as they fall beyond the scope of this thesis. Some of these will however be briefly listed in section 4.3.3 of chapter 4. Note also that we will only discuss classification methods for classifying instances to discrete classes and ignore continuous valued target functions.

Before we start with explaining how the Naive Bayes Classifier works we will introduce *Bayes' Theorem* which is named after the eighteenth-century British mathematician Thomas Bayes.

3.2 Bayes' Theorem

Bayes' Theorem is a simple but powerful theorem which is well known in the world of probability theory, statistics and machine learning [41]. Bayes' Theorem calculates the posterior probability $P(A/B)$. This denotes the probability that A occurs given the fact that B occurs. This probability can be calculated as follows:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \quad (3.1)$$

The theorem uses the prior probability $P(A)$ (the probability that A occurs) together with the conditional probability $P(B|A)$ and the prior probability $P(B)$ which serves as a normalization factor so that the probabilities sum to 1. These probabilities should thus be known in order to be able to compute the posterior probability $P(A/B)$.

The best way to illustrate Bayes' Theorem is by means of an example. The example of Tom Mitchell in [41] will be used which shows how important Bayes' Theorem can be for a wide range of fields. The example is situated in the medical world and addresses a diagnoses problem with two hypotheses: the patient is diagnosed with cancer or the patient is not diagnosed with cancer. It's obvious that a solid diagnosis is crucial as live or dead are depending on it.

Given that a test of a patient results as positive, should we diagnose the patient as being positive? At first sight almost all people would say yes because they forget a very important aspect of tests: most of the tests are not 100% certain. To diagnose the patient we have to know how good the test is and how large the chance is of having cancer. If we are supplied with these values we can calculate the probability $P(\text{cancer}/\text{positive})$. This is the probability that a patient has cancer given the fact that the test gave a positive result. Suppose the following probabilities are given:

- $P(\text{cancer}) = 0.008$
- $P(\neg\text{cancer}) = 0.992$
- $P(\text{positive}/\text{cancer}) = 0.98$
- $P(\text{negative}/\text{cancer}) = 0.02$
- $P(\text{positive}/\neg\text{cancer}) = 0.03$
- $P(\text{negative}/\neg\text{cancer}) = 0.97$

Knowing these probabilities we can calculate $P(\text{cancer}/\text{positive})$ using Bayes' Theorem. This gives:

$$P(\text{cancer}/\text{positive}) = \frac{P(\text{positive}/\text{cancer})P(\text{cancer})}{P(\text{positive})}$$

It might seem as if $P(\text{positive})$ is missing, but this probability can be easily calculated by summing up $P(\text{positive} \wedge \text{cancer})$ and $P(\text{positive} \wedge \neg\text{cancer})$. These two probabilities can be calculated using the product rule:

$$\begin{aligned}
P(\text{positive} \wedge \text{cancer}) &= P(\text{positive} | \text{cancer})P(\text{cancer}) = 0.0078 \\
P(\text{positive} \wedge \neg \text{cancer}) &= P(\text{positive} | \neg \text{cancer})P(\neg \text{cancer}) = 0.0298
\end{aligned}$$

These values can now be substituted into the formula and calculate $P(\text{cancer} | \text{positive})$:

$$\begin{aligned}
P(\text{cancer} | \text{positive}) &= \frac{P(\text{positive} | \text{cancer})P(\text{cancer})}{P(\text{positive} \wedge \text{cancer}) + P(\text{positive} \wedge \neg \text{cancer})} \\
&= \frac{0.98 * 0.008}{0.0078 + 0.0298} \\
&= \frac{0.0078}{0.0376} \\
&= 0.21
\end{aligned}$$

This leads to the conclusion that although the test was positive, the probability that the patient has cancer is “only” 21%. This example shows that Bayes’ Theorem can be very useful for calculating unknown probabilities when the prior and class conditional probabilities are given. We will now introduce an algorithm based on Bayes’ Theorem that is frequently used for classification: the Naive Bayes Classifier.

3.3 Naive Bayes Classifier

In this section we explain how the Naive Bayes Classifier works [41, 51]. This is a statistical technique that can be used to classify documents into predefined classes using the words in the document as features. As the name indicates, this classifier uses Bayes’ Theorem to perform this classification. The reason why it is called naive is because it makes the simplifying assumption that all features are conditionally independent given the class. Even though this assumption of independence is often not met, it has been shown that the Naive Bayes Classifier gives very good results in practice [31]. Especially compared to much more complex classification methods it performs remarkably well.

The formalization of the simplifying assumption which is made for computing the Naive Bayes classifier is as follows:

$$P(X|C) = \prod_{i=1}^n P(X_i|C) \quad (3.2)$$

In this formula, X is a feature vector containing the features (X_1, \dots, X_n) and C is a class. It states that the probability of observing the conjunction of features X_1, \dots, X_n is the product of the probabilities of the individual class conditional probabilities of these features. This assumption makes it much easier to compute $P(X|C)$. We can now use this assumption to calculate the probability that given a feature vector, an instance belongs to a certain class. This can be done using Bayes’ Theorem:

$$\begin{aligned}
P(C|X) &= \frac{P(X|C)P(C)}{P(X)} \\
&= \frac{P(C) \prod_{i=1}^n P(X_i|C)}{P(X)} \\
&= P(C) \prod_{i=1}^n P(X_i|C) \tag{3.3}
\end{aligned}$$

Note that in the last line, the denominator $P(X)$ is dropped. This is because $P(X)$ is invariant across all classes. To assign an instance to a class we have to compute the probability $P(X/C_j)$ for each predefined class C_j . We can write this as

$$NBC = \underset{C_j \in C}{\operatorname{argmax}} P(C) \prod_{i=1}^n P(X_i|C) \tag{3.4}$$

This formula is the Naive Bayes Classifier. The instance is assigned to the class with the maximum probability. We will see how this classifier can be applied in practice in chapter 4. We now turn our attention to the second classification algorithm which is the k-Nearest Neighbour Classifier.

3.4 k-Nearest Neighbour Classifier

After introducing the Naive Bayes Classifier, this section discusses the k-Nearest Neighbour Classifier (KNN Classifier) [41]. This algorithm bases its classification on the most common class of the k nearest neighbours in the training set. It is an instance-based classification approach which means that classification is performed by measuring the similarity of new instances to training instances. There are three main aspects which characterize a k-Nearest Neighbour Classifier:

1. A distance metric needed to define the concept "nearest".
2. The number of neighbours which is the k value.
3. A strategy to decide the class of the new instance.

Choosing a good distance metric is crucial for the algorithm as it will define which training instances will be selected as neighbours. We will show an example of choosing a metric for text classification in the next chapter. Another important aspect is choosing the value for k . If the value of k is too low, not enough neighbours will be considered and important information can be lost. On the other side, a large value for k will result in including instances which are further away and thus not similar enough. The value of k should therefore be fine-tuned and is totally application dependent. The third aspect is the strategy which is used to decide the class using the classes of the k nearest neighbours. The standard way to do this is to let the neighbours vote (each neighbour votes for his own class) and classify the instance to the class which receives the most

votes. Another more advanced way of determining the class is called distance weighting. This strategy uses the inverse of the distance as a vote weight so that the neighbours which are near to the instance have greater influence on the classification than neighbours which are further away from the instance.

Once these three aspects have been identified, the algorithm can be executed as follows:

1. Measure the distance of the training instances to the new instance with the distance metric.
2. Retrieve the k nearest training instances based on the measured distances.
3. Apply the strategy to classify the new instance given these k nearest neighbours.

In the next chapter we will see how these 2 algorithms are being used in the context of adaptive behaviour. In the research part of this dissertation, the foundations of these two algorithms will be used as a basis for two similar algorithms. They will however not be used for classifying pages but for determining the interestingness of pages.

Chapter 4

Adaptive Behaviour for Web Sites

4.1 Introduction

In this chapter an overview will be given of the existing adaptive behaviour methods for web sites and, in a broader context, hypermedia. *Hypermedia* is a style of building systems for organising, structuring and accessing information around a network of multimedia nodes connected together by links [21]. The term hypermedia is actually derived from hypertext which is text that contains links to other texts. It extends the notion of hypertext to also include links to any kind of multimedia such as graphics, sound and video. The World Wide Web is the most well known and biggest hypermedia system containing structured information in the form of pages which are connected together with hyperlinks. When we talk about adaptivity in this dissertation, it will be in the context of web sites although both concepts of web sites and hypermedia are interchangeable in this chapter.

Adaptive behaviour is the process of adapting the content and/or structure of a web site in order to increase the usability of the web site and the satisfaction of the visitors. This adaptation is mostly based on some kind of user profile or browsing behaviour of the visitors and comes in many different flavours.

There are many user characteristics to which a web site can adapt itself to. These are characteristics like knowledge, interests, goals, preferences and behaviour which are all user dependent. Almost all adaptation is based on these kinds of user or group dependent characteristics. To adapt to the users' knowledge for example, a web site can "choose" to only show some kind of information if the user has the background knowledge to grasp this information. Adaptation to the users' interests can be done by maintaining a user profile and giving recommendations based on this user profile. When the goals of the users are known, the web site can adapt itself in such a way that it shows only the information relevant to these goals. Users can also have preferences to which the web site can adapt itself. An example of this kind of adaptation is when a user on a news site likes to read the sport headlines before the cultural headlines. A web site

can adapt itself to this preference by showing the cultural headlines beneath the sport headlines. Another kind of adaptation is behaviour adaptation. By storing the browsing behaviour of the user, the web site can, for example, provide links to frequently accessed pages on the home page.

The overview that follows will discuss existing adaptation methods. Note that there is a difference between adaptation methods and adaptation techniques. Adaptation methods are more conceptual abstract adaptation ideas whereas adaptation techniques are specific adaptation algorithms for a particular adaptation method. Instead of using characteristics like the ones identified above as a guide throughout this chapter, we will follow the taxonomy of adaptive hypermedia technologies which is proposed by Peter Brusilovsky [10, 11, 12]. Brusilovsky defines adaptive hypermedia systems as follows:

By adaptive hypermedia systems we mean all hypertext and hypermedia systems which reflect some features of the user in the user model and apply this model to adapt various visible aspects of the system to the user.

A consequence of this definition is that this taxonomy is only applicable to web sites which have some kind of user model of its visitors. This user model doesn't need to be some accurate description of the user or the group of users it belongs to and be can be, for example, solely based on the browsing history of the user. As can be seen on figure 4.1 taken from [12], the taxonomy is based on adaptation at two levels: content or presentation based adaptation and navigation or link based adaptation. We will take a look at each of these methods in the following section and in particular link recommendations which can be classified as an adaptive link generation technique.

Besides the classification by Brusilovsky, another classification by Perkowitz [46] identifies two kinds of web site adaptation: customization and optimization. *Customization* or *personalization* adapts the presentation of the web site in order to give personalized views. Note that this presentation adaptation not only includes layout changes but also the generation of interesting links. This in contrast to the classification of Brusilovsky where the generation of links is classified as an adaptation method at the navigational level. In fact almost all navigation and presentation level adaptation methods proposed by Brusilovsky are customization methods as they provide personalized views on the web site and do not change the web site structure. Methods that do change an aspect (e.g. structure or presentation) of the site permanently so that these changes are visible for each visitor are classified as *optimization* methods by Perkowitz. We will discuss some of these optimization techniques described by Perkowitz separately in section 4.2.3. The last section (section 4.3) of this chapter generally describes recommendation systems and discusses some existing link recommendation techniques.

4.2 Overview of Existing Adaptation Methods

We will break this section down into three subsections: adaptation on the presentation level, adaptation on the navigation level and web site optimization.

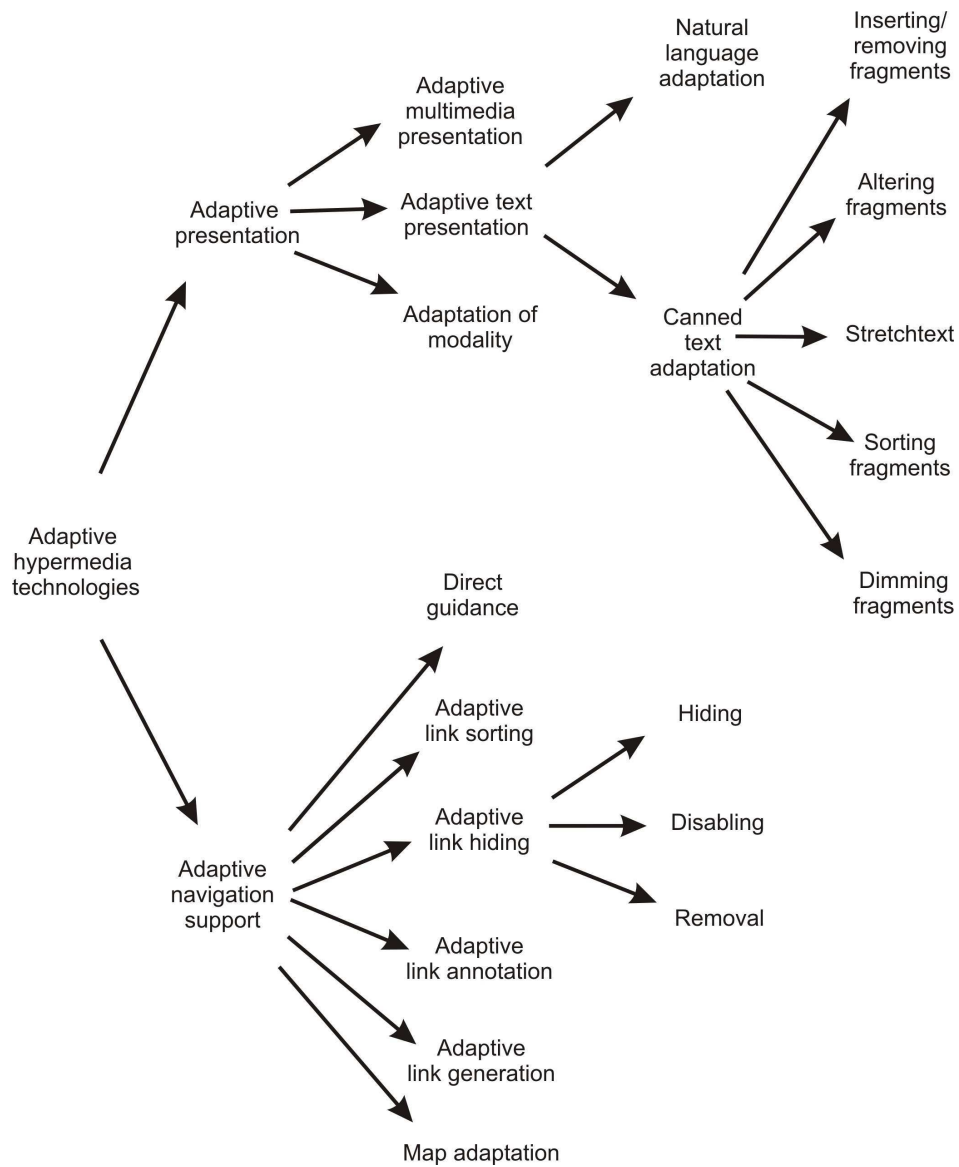


Figure 4.1: Taxonomy of adaptive hypermedia technologies.

The first two subsections (sections 4.2.1 and 4.2.2) give an overview of the adaptation methods that are well known in the field without going too much into detail. The structure of these two sections follows the classification of adaptation methods by Brusilovsky. The third subsection (4.2.3) discusses some web site optimization techniques which are not included in the classification of Brusilovsky.

4.2.1 Adaptation at The Presentation level

In this section we will take a look at the various methods available to adapt the content of a web site based on user characteristics. These methods often take in account the knowledge of the users and give personalized views of the available information based on this knowledge. It's obvious that someone who is familiar with a certain subject wants more detailed information about the subject than someone who isn't. The person who isn't familiar with that subject only wants to read the relevant and trivial information because it is new ground for him. This is an example of the adaptation which occurs at the presentation level and which can be used in instructional web sites or any other web sites which use some kind of knowledge level representation for its users. We will now take a closer look at the three categories of content based adaptation methods defined by Brusilovsky.

4.2.1.1 Adaptive multimedia presentation

In *adaptive multimedia presentation* only the content in the form of multimedia is adapted to the characteristics of the user. This is one of the least used and studied methods in the field of adaptive behaviour. The reason why is because it is much easier to adapt textual information than multimedia information such as video and sound. On the other hand, an image or video can say more than 1000 words. This is the power of multimedia and the reason why this adaptation method will be used more and more in the future when new techniques to adapt multimedia arise.

Most of the multimedia adaptation nowadays is based on technical device details and user preferences and not on the knowledge of the user. Multimedia can for example be adapted to the screen size, Internet connection type and language of the client device. It's obvious that a user who visits a web site using an old computer with a 486 processor and a 56K modem doesn't want to wait for large multimedia information to load. On the other hand, a user with a Pentium 4 processor and a cable modem doesn't want mediocre multimedia quality. For these kinds of problems multimedia adaptation can make sure that the multimedia is adapted to the users' needs. A well known language to achieve this kind of adaptation is SMIL (Synchronized Multimedia Integration Language) [3]. More information about adaptive multimedia presentation can be found in [30, 49].

4.2.1.2 Adaptive text presentation

The most explored and used adaptation method is without any doubt *adaptive text presentation*. In adaptive text presentation users get, based on their user model, different text fragments for the same pages. Adaptive text presentation can be decomposed in natural language adaptation and canned text adaptation.

Natural language adaptation

Natural language adaptation uses natural language processing and generation techniques to provide some form of textual adaptation. These techniques are mostly situated in the field of artificial intelligence and are at the moment rarely

used in web sites. Natural language adaptation is however promising because it can be used to analyze and construct text at runtime. This way virtual pages can be constructed and tailored to the user models of the visitors [25]. More information about these natural language techniques can be found in the field of artificial intelligence .

Canned text adaptation

The most used form of adaptive text presentation is *canned text presentation*. Canned text adaptation uses predefined text fragments which are used to generate a page at runtime based on the user model. In contrast to the previous method where text is generated, the text fragments are static. The key issues are the inclusion or exclusion and the place of the fragment. An example of this kind of adaptation is when different text fragments are stored for some piece of information, and each text fragment is in another language. By retrieving the language of the user, the correct fragment can be shown. We will discuss the following five methods of canned text adaptation as defined by Brusilovsky:

- Inserting/removing fragments
- Altering fragments
- Stretchtext
- Sorting fragments
- Dimming fragments

The first method of canned text adaptation is *inserting and removing fragments*. The insertion or removal of fragments is conditionally which means that some conditions have to be satisfied in order that the fragment is being removed or inserted. These conditions are mostly based on the user model but can also be based on some other context (e.g. time based conditions which only show fragments at a certain time [7]). This adaptation method can be used to give extra explanation at runtime to users who need it. A drawback of this method is that identifying the user model is crucial as it determines which fragments are shown and which are not. If the user model is not correctly identified, the user can get a page containing fragments which don't suit his needs.

The next canned text adaptation method up in the list is *altering fragments* and is also known as *fragment variants* [37, 36]. This method selects a fragment out of a set of alternative fragments at runtime. It is, like the previous method, also based on conditions which determine which fragment is selected. A good example of this method is selecting a fragment based on the expertise of the user. Hence, expert users can get a more in-depth fragment than novice users. This method suffers from the same drawback as the previous one, i.e. identifying the user model is extremely vital.

Another method, which is called *stretchtext*, uses hot words which can be used in the same fashion as hyperlinks. The only difference is that instead of directing the user to a new page when clicked, the clicked hot word is being replaced by a fragment. It is also possible to roll back this operation so that

only the hot word is showed. By taking in account the user model and especially the user's knowledge, unknown hot words can be collapsed and known words uncollapsed when a page is accessed. A very interesting feature of stretchtext is that the user can still collapse words that were uncollapsed. When he performs this action, he actually unconsciously shows that the user model is incorrect because it estimated that he knew the concept behind the hot word. This is a major benefit as the user model can be updated accordingly and hence be optimized at runtime. This method also gives control to the user in contrast to the previous methods where the user just had to sit back and be complacent with whatever information was offered to him.

Sorting fragments is the following method that we will discuss. Just like the name says, fragments are sorted according to the knowledge, interests or preferences of the user. Fragments which are most related to these characteristics are placed above those which are not. There is however some weak point for this method. This method can only be applied to a set fragments which are semantically unrelated. When one concept is introduced in a fragment and is repeated in another fragment, it is not possible to reverse their order. Aside from this restraint, it can be a helpful method for the user in his pursuit of information.

The last canned text adaptation method is *dimming fragments*. This method dims fragments which are estimated as being not relevant for the user. This can be done by changing the font color or decreasing the font size for example. The advantage of this method is that the fragment is not totally hidden and that the user can still read it if he is interested.

4.2.1.3 Adaptation of modality

The third adaptation method at the presentation level is called *adaptation of modality*. This adaptation is based on the fact that some single piece of information can be represented by different types of media. Based on the user model, it can be for example better to represent information with a video (children tend to grasp information better by means of video than by plain text for example). Notice also the similarity with the altering fragments adaptation methods. The only difference is that the altering fragments method only considers text fragments whereas this method considers all types of media.

4.2.2 Adaptation at the Navigation Level

After the discussion of adaptation at the presentation level, we now focus on adaptation at the navigation level which manipulates links on the web site at runtime. There are 6 methods of this kind of adaptation as can be seen on figure 4.1: direct guidance, adaptive link sorting, adaptive link hiding, adaptive link annotation, adaptive link generation and map adaptation. We will discuss each of them in the following sections.

4.2.2.1 Direct guidance

We begin this overview of adaptation methods at the navigation level with direct guidance. In *direct guidance*, the user is offered a dynamical link to the next best page. This next best page is based on the user model and/or browsing behaviour of the user. This method is good practice for tutoring web sites or web sites which know the goals of their users. It's however not a good idea to use this method solely without any other navigation options because most of the users want to explore a web site and don't like it when they are given only one option.

4.2.2.2 Adaptive link sorting

A method which is similar to the fragments sorting method but on the link level is *adaptive link sorting*. Just as with sorting fragments, links are sorted according to the user model (at least according to a characteristic embedded in the user model). Using this method, the most relevant links will be placed on top and the less relevant on the bottom of the page. This method is also used in link recommendation lists such as a recommendation list of movies that relate to another particular movie. It's obvious that this method can only be used for non-contextual links as it would otherwise remove links out of their context. Another drawback for this method is that it may disorientate users which frequently visit a web site because links can have a different place on every visit.

4.2.2.3 Adaptive link hiding

Another method which resembles to one of the canned text adaptation methods is the *adaptive link hiding* method. All links that lead to non-relevant information for a given user are hidden by this method. This method has three variants:

- Adaptive link hiding
- Adaptive link disabling
- Adaptive link removal

Adaptive link hiding is already defined above but some additional explanation to the "hiding" principle is necessary. Hiding links does not mean that the links are physically unreachable. The purpose of link hiding is to hide the fact that a certain word is actually a link. This is mostly done by making the linked word look exactly the same as the text surrounding the word. Users will see the word and if they understand the linked word, they will not notice anything. Other users which ask themselves questions when encountering that word, can still discover that this word is actually a link and follow the link. This method can be used for both contextual and non-contextual links as no page information is removed and links can still be followed.

Adaptive link disabling goes a step further and totally disables the link so that it can not be followed. Although the link is disabled, its visual representation does not. This is a bit awkward as it is in contradiction with the user's

expectation which expects that the link can be followed. This method can also be used for both contextual and non-contextual links but unlike the previous method, the links can not be followed anymore. This method should be used together with the previous one because links that can not be followed is against all laws of usability.

The last method is the least used one and can only be used for non-contextual links. *Adaptive link removal* totally removes the link and can thus not be used in running text. This method is most sensitive to faults in the user model because all non-relevant links are removed. This means that even if some of the removed links turn out to be relevant after all, the information is gone and the user may miss some interesting information. This method can be beneficiary for link lists in cooperation with a solid user model as it shrinks down the navigation space to the links which are most relevant.

4.2.2.4 Adaptive link annotation

The fourth method of adaptation methods on the navigation level is *adaptive link annotation*. This methods adds some kind of annotation to the links in order to show the relevance of the links. This can be done by giving the links different colours where each colour stands for a different kind of relevance. As an example, Wu [61] identifies three kinds of annotations: relevance, knowledge and understanding. They are based respectively on the relevance of the links, whether the user already knows the concepts described in the pages the links point to and whether the user is ready to understand the information on the linked page. Of course other criteria can be used and they can also be combined together. This method can be used for any kind of links without losing contextual information.

4.2.2.5 Adaptive link generation

This is the method which is most interesting for this thesis because it includes the link recommendations. *Adaptive link generation* generates new links for a page and includes three cases:

- Discovering new links between pages and adding them permanently to the set of existing links.
- Generating links for similarity-based navigation between pages.
- Dynamic recommendation of relevant links.

The first case of link generation is based on visitor patterns of all users and creates new links between pages which were not related at design time. For a large web site, it is not exceptional that the designer fails to relate pages which the users see as related. To overcome this design problem, the adaptive link generation method can be used to detect and solve these kinds of problems. This method changes the structure of the web site permanently and thus influences the browsing behaviour of all future users.

The second case is similarity-based navigation which uses a similarity metric and links each page to a set of most similar pages [59]. This can be a very

helpful guide when navigating through a web site if the similarity metric is well-thought-of.

The last case is the dynamic recommendation of relevant links or, as we called it in the first chapter, link recommendations. This is either based on the content of the pages (*content-based recommendations*), on the web site access patterns of the users (*collaborative recommendations*) or on a combination of both content and web site access patterns. We will discuss recommendation systems and the existing techniques of this method in section 4.3.

4.2.2.6 Map adaptation

The last method of adaptation at the navigation level is *map adaptation*. Some web sites offer a site map to the users which give a view on the link structure of the web sites. The idea of map adaptation is to personalize this map for each user. Some of the other methods such as link removal, link annotation and link sorting can be used on this map to adapt it.

4.2.3 Optimization Techniques

All methods discussed in sections 4.2.1 and 4.2.2 are used to customize the web site to the personal needs of its visitors. The type of adaptation offered by these methods is temporal, i.e. it does not affect the structure or presentation of the web site permanently. There is only one method in the previous sections that does affect the structure permanently: the discovery of new links between pages and adding them permanently to the set of existing links (section 4.2.2.5). Perkowski defines some other optimization techniques which alter the web site structure in [47]. These techniques, which are transformations on the site, are promotion and demotion, highlighting, linking and clustering. We will discuss each of these transformations briefly in the next sections.

4.2.3.1 Promotion and demotion

Promotion and *demotion* of web pages and links is based on the popularity of the pages and links. It uses user access logs to calculate the popularity of pages and links. The promotion of a page is the process of moving a page closer to the root page of a web site. This way, the page is easier to find for the visitors. Promotion of a popular link occurs when a link is moved closer to the top of the page.

Demotion is the opposite of promotion as it moves nodes which are not accessed frequently further away from the root node of the web site. For the demotion of links, links are placed closer to the bottom of the page as they are not popular.

4.2.3.2 Highlighting

Highlighting occurs when popular links are highlighted for each visitor in the same way. This seems to be the same as the adaptive link annotation in section 4.2.2.4 but it isn't. The difference is that here, the highlighting is permanent

and the same for every visitor whereas the adaptive link annotation is dynamic and user dependent.

4.2.3.3 Linking

Large web sites sometimes have the problem that pages which are frequently accessed together in one visitor session, are unrelated in the site structure. To solve this problem, two pages can be linked when the access logs reveal that these pages are considered as being related by the visitors. The designer should therefore write an algorithm which discovers unlinked related pages at runtime and links them together when this is detected. This *linking* can result in much less time to find and visit both the related pages and thus increases the users' satisfaction. *Unlinking*, on the other hand, deletes links which are almost never used by the visitors.

4.2.3.4 Clustering

Clustering is the grouping of related pages on a single page so that these related pages are accessible from the same pages. When a group of related pages is detected at runtime, an index page for these related pages is created. This can drastically reduce the time to browse through related pages which are not grouped together by the designer.

We have now seen many different ways of incorporating adaptive behaviour into web sites. In the following section we will take a closer look at the most relevant adaptation method for this dissertation: link recommendations.

4.3 Recommendation Systems

4.3.1 Introduction

This section discusses recommendation systems (also known as recommender systems) and gives an overview of the existing link recommendation techniques. To avoid unclearness about what a recommendation system precisely is we will use the following definition of Brusilovsky[12]:

Adaptive recommendation systems attempt to deduce the user's goals and interests from his or her browsing activity, and build a list of suggested links to nodes that usually can not be reached directly from the current page, but are most relevant to that user.

This definition states that the recommendations are only based on the browsing activity but it can also be based on the user model and therefore we will extend the definition as follows:

Adaptive recommendation systems attempt to deduce the user's goals and interests from his or her browsing activity and/or user model, and build a list of suggested links to nodes that usually can not be reached directly from the current page, but are most relevant to that user.

All systems corresponding to this definition are recommendation systems. Most recommendation systems differ in the way that they deduce the goals and interests and in how relevance is computed. The techniques to compute this relevance can be divided into three main types: content-based recommendations, collaborative recommendations and a hybrid approach which is a combination of these two types [5]. We will take a closer look at these recommendation types separately in the section 4.3.3. There we will see how the algorithms which we saw in chapter 3 are used in recommendation systems to generate link recommendations. To conclude the chapter, an example of how recommendation systems are used in practice is given. This example is the recommendation system of the well known online megastore Amazon.com¹. Let's first take a look at some of the characteristics of recommendation systems.

4.3.2 Characteristics of Recommendation Systems

In this section, some of the main characteristics which define recommendation systems are given. This is not an all-encompassing list and is based on some of the features described in existing recommendation system literature.

- **Open or closed corpus:** open corpus recommendation systems are not bound to one web site or a group of web sites and recommend links to pages in the seemingly unbounded WWW. Closed corpus recommendation systems on the other hand, recommend only links to pages in a certain area, typically a single web site or a group of web sites. In fact, recommendation systems are still the only adaptive systems which achieve some success in open corpus[13].
- **Content-based and/or collaborative recommendations:** as was already mentioned, the recommendations are either based on the content of the pages or on the browsing behaviour of similar users. There are also some hybrid techniques which use both recommendations techniques together [5] and which we will see in section 4.3.3.3.
- **Implicit feedback or explicit feedback:** many recommendation systems use explicit feedback such as user ratings to recommend items to users. The disadvantages of this approach are that it requires users to spend additional effort and that they may stop providing feedback when they don't notice any benefits. Implicit feedback methods don't have these disadvantages because users are not aware of providing feedback just by browsing the web site. A simple way of inferring implicit feedback is by viewing the visited pages as positive ratings. Other ways to do this is by examining the duration of page views and observing the retention of pages (i.e. observing which pages the visitor has bookmarked or saved) [43].
- **Exploitation or exploration:** does the recommendation system strictly generates pages which are similar to pages the users liked in the past or does it also recommends other pages which are less similar [58]? Using exploration, users can be introduced to other popular information which is not heavily related with their interests but which can be interesting

¹The web site of Amazon.com is available at <http://www.amazon.com>.

once they have detected this information. When a recommendation system only exploits its knowledge, users can get stuck in the same limited area of the web site. The best solution is to use a mix of both approaches: starting with exploitation for example, and gradually switching to exploration when the user gets recommendations to the same information over and over again.

- **Detecting drifting user interests:** some recommendation systems are able to detect changes in user interests. When user interests change quickly, systems which detect this change obviously have an advantage over those that don't. In [38], Koychev and Schwab present a method for dealing with drifting interests by introducing the notion of gradual forgetting.
- **Editable user model:** enabling users to edit their user model can lead to better recommendations. By editing their user model users can, when a model is based on a set of weighted keywords for example, change the weights of the keywords and add keywords which should be taken into account when recommendations are generated [6]. An editable user model is also a solution for the drifting user interests problem. Note that a user model is not compulsory for recommendation systems although a user model can certainly lead to more personalized recommendations.

We will see some examples of existing recommendation systems in section 4.3.4 and show which characteristics they exploit.

4.3.3 Link Recommendations Techniques

This section describes the three main types of link recommendations: content-based recommendations, collaborative recommendations and a hybrid approach based on a mix of both types. Once these three types have been described, the existing link recommendation techniques for each type will be discussed in sections 4.3.3.1, 4.3.3.2 and 4.3.3.3.

Content-based recommendations are based on content level similarities between pages. These recommendations generate links to pages which are in terms of content similar to the pages that the specific visitor liked in the past. A common way to do this is by extracting keywords out of the pages which are visited by the users and maintaining these keywords in their user model. When a user likes a page and gives it a good rating, the weights of the keywords related to the page are increased in his user model. This way pages which contain multiple occurrences of the keywords which have a high weight value in his user model can be recommended. Other ways of generating content-based links, including the ones based on the algorithms described in chapter 3, will be discussed in the next section.

The second type of recommendation techniques are collaborative recommendations and are also known as collaborative filtering techniques. These are based on the browsing behaviour of similar users and don't take into account the content of web pages. They draw on the idea that people who agreed in their subjective evaluation of past items are likely to agree again in the future [50].

This similarity between the users is mostly based on ratings or purchases of items. Users which give similar ratings for the same items are considered to be similar. Note that we are talking about items instead of pages because ratings are mostly given to items on the web pages (e.g. movies like on the Internet Movie Database ²) instead of on the pages themselves.

Both types have their pros and cons [5]. Content-based recommendation techniques are good at representing the interests of a single user. A shortcoming is that the textual feature extraction procedure is not capable of detecting all aspects of web pages like for example the available non-textual multimedia information. It is also bound to the restriction that it will only recommend pages similar to the ones the user liked before. Furthermore, the need of feedback from the user is crucial. A user has to specify his interests and/or give ratings in order to have a set of pages to which new pages can be compared for similarity.

Almost all of these shortcomings are solved by collaborative recommendation techniques which are good at representing the common interests of a group of similar users. These techniques don't use textual representation of pages and can therefore deal with any kind of information. Because collaborative recommendations are based on the likings of a group of similar users, this should not restrict the recommended pages to being strictly similar to the ones the user liked before. The third problem of content-based recommendation techniques is partially solved because the recommendations are depending on the ratings of a group of users and therefore fewer ratings per individual would be sufficient in order to give recommendations.

On the other hand, there are also some cons when using collaborative recommendation techniques. The first problem is known as sparsity: if the number of users is small relative to the amount of available information, collaborative recommendations tend to cover only a small area of the web site. The second problem is the first-rater problem: pages can not be recommended until a certain amount of users have visited them, if they ever discover the new pages at all. Another problem is that group similarities are rather general and can not lead to user specific recommendations. As you probably noticed, content-based recommendation systems don't have these shortcomings. This observation has lead to new hybrid approaches which combine both ways of recommending interesting pages.

Hybrid recommendation systems combine the advantages of the content-based and collaborative recommendation methods. By combining both methods, recommendations can represent common interests and can at the same time deal with personalized interests. This hybrid approach is a rather new approach which is being used more and more recently. We will see some techniques of combining both methods in section 4.3.3.3.

²IMDb is the world biggest online movie database which uses a specific recommendation system itself. It is available at <http://www.imdb.com>.

4.3.3.1 Content-based recommendation techniques

There are many techniques available for content-based recommendations. Almost all of them are machine learning techniques for classifying pages into categories. We have already seen two of the most frequently used machine learning techniques in the previous chapter: the Naive Bayes Classifier and the k-Nearest Neighbour Classifier. We will now show how these two techniques are used and briefly discuss some other frequently used techniques.

Content-based recommendations using the Naive Bayes Classifier

In chapter 3, we introduced the Naive Bayes Classifier. Here we will show how the Naive Bayes Classifier can be used in practice to classify pages as being interesting or not for a particular user. We will use the word “page” throughout this section for easy comprehension but it could also be any other web item having some textual description (e.g. meta data).

When a user visits pages and rates them as interesting they are classified to the class of interesting pages. Pages which get low rates are classified to the class of uninteresting pages. The most standard way to classify new pages using the Naive Bayes Classifier is by using the words on these pages as features (another way is to use meta data containing descriptive keywords). This is known as document or text classification. Using the words on the pages which are classified as being interesting or not, the Naive Bayes Classifier can be used to predict the classes of not yet visited pages. Note that the representative set of training pages are in this case the pages which are rated by the user. The simplifying assumption which is made by the classifier means that, for this example, words are considered as being independent of each other. This is of course not true as a page is not a bag of words where order is of no importance. Even though this assumption is illegal, the algorithm gives good results [31].

To classify a page, we use the occurrences of the words (of course omitting trivial words like “and”, “in”, “the”, etc) on these pages as features. Let’s, for example, say that a user has rated a page containing the words “PHP”, “HTML”, “browser” and “design” as being interesting. This will increase the probability that not yet visited pages with the same words will be classified as being interesting. To compute the probability that a page belongs to a certain class, we have to calculate the probabilities that each word in the page belongs to this class. Following equation 3.2 from chapter 3, the probability that a page which contains the bag of words W belongs to the class C can be computed as follows:

$$P(W|C) = \prod_{i=1}^m P(w_i|C)$$

But how can the probability $P(w_i|C)$ be computed? There are many ways to estimate this probability. In [41], this is done by maintaining a vocabulary which contains all words that occur in the pages rated by a specific user. $P(w_i|C_j)$ can then be estimated as follows:

$$P(w_i|C_j) = \frac{n_i + 1}{n + |Vocabulary|}$$

In this formula, n is the number of words in the pages of class C_j , n_i the number of times w_i occurs in the pages with class C_j and $|Vocabulary|$ the size of the vocabulary [41]. The reason why 1 is added to the numerator³ is because otherwise unseen words will lead to a zero probability of $P(w_i|C_j)$ and therefore also a to zero probability of $P(C/X)$. To determine the class to which the page belongs, the formula of the Naive Bayes Classifier should be used:

$$NBC = \operatorname{argmax}_{C_j \text{ in } C} P(C) \prod_{i=1}^m P(w_i|C)$$

For $P(C_j)$, the percentage of pages belonging to this class is a good estimator. Note that these class conditional and posterior probabilities are all generated during the training phase of the algorithm.

Using these estimators, pages can be classified by the Naive Bayes Classifier in a fairly easy way. This is the main advantage of the algorithm. When one can find a reliable estimator which is easy to compute, the algorithm requires much less computational power than other more complex classification algorithms. A disadvantage of this classifier is that it can give distorted results for features which are heavily interrelated and thus dependent on each other.

Content-based recommendation using the KNN Classifier

The second content-based recommendation technique which we discuss uses the KNN Classifier. Recall from section 3 that this classifier needs a distance metric, a number of neighbours and a classification strategy. This technique determines the class of an unseen page based on the most common class of the k most similar pages which have already been classified. The only thing which is needed specifically for this content-based recommendation technique is a distance metric for computing the similarity between pages. Once we have this metric, the k most similar pages can be used as the input for a classification strategy such as one of those mentioned in section 3. Let's choose the standard strategy where the vote of each page is equally weighted as the strategy that we will apply in this example.

An example of a well-known and frequently used distance metric to compute the similarity between pages is the *Term Frequency-Inverse Document Frequency* (TF-IDF) algorithm [33, 36, 52]. This algorithm is one of the most successful weighting schemes for documents. The main reason why is because it considers words that appear in one page and rarely in others as being more relevant to the class of the page than words which appear frequently. This way prepositions, adjectives and conjunctions don't need to be filtered out of a document.

³This is called Laplace smoothing.

TF-IDF uses a *Vector Space Model* to represent a page: a vector relative to some dictionary vector where each element in the vector is a weight representing each word in the dictionary. These weight elements are used to express the importance of the words in the page. They are calculated by multiplying the *Term Frequency* (TF) with the *Inverse Document Frequency* (IDF). The Term Frequency is the number of times the word represented by the element occurs in the page. The Inverse Document Frequency is the inverse of the number of pages in which the word occurs at least once. The formula to compute the element e which represents the weight of word w in page p is:

$$e_i = TF(w_i, p) \cdot IDF(w_i)$$

The Inverse Document Frequency can be computed as follows:

$$IDF(w_i) = \log\left(\frac{|P|}{DF(w_i)}\right)$$

The DF stands for Document Frequency which is the number of pages in which word w_i occurs. The $|P|$ denotes the number of pages compared and is in the case of measuring the distance between two pages obviously always 2. Using the above formulas, a vector can be constructed for both pages. The distance between both vectors then can be measured using the *cosine similarity* which is frequently used in combination with the TF-IDF algorithm⁴. The cosine similarity is calculated as follows:

$$Similarity(p_1, p_2) = \frac{\sum_{w \in W} (e_{p_1, w} \cdot e_{p_2, w})}{\sqrt{\sum_{w \in W} e_{p_1, w}^2} \cdot \sqrt{\sum_{w \in W} e_{p_2, w}^2}}$$

The greater the value of the cosine similarity, the more similar the pages are. To be able to use this as a distance metric the inverse should be taken so that similar pages have a small distance instead of a large distance. Having defined this distance metric, the k most similar pages can be retrieved. Using the proposed classification strategy the class can be easily determined. An advantage of the KNN algorithm is that no additional data structures are needed. This is in contrast to the Naive Bayes Classifier where a vocabulary of all words is constructed for each user which results in higher memory requirements. Another advantage is that there is no training time required for this algorithm. A disadvantage is the categorization time being linear to the amount of training pages because each training page should be compared with a new unseen page. We will now briefly discuss some other techniques which are also used in content-based recommendation.

Other techniques for content-based recommendations

Some other frequently used techniques used for generating content-based recommendations are:

⁴This is called cosine similarity because it computes the cosine of the angle between two vectors.

- **Support Vector Machines:** this technique separates positive from negative examples in the best possible way. It does this by searching the hyperplane which separates the positive from the negative training examples by the widest margin. When this hyperplane is constructed at training time, new items can be classified according to which side of the hyperplane they fall onto. Support Vector Machines can handle high dimensional input which is certainly needed for text classification as it uses the words on the pages as features. They are considered to be the most accurate classification method but are very slow to train [34].
- **Decision Trees:** a decision tree classifier uses a tree where the nodes are terms (i.e. the words in the classified pages), the branches are tests on the weight of the terms and the leafs are classes. Classification is done by recursively testing the weight that a term of a node has in the feature vector of the page until a leaf node is reached. The page is then assigned to the class which labels the leaf node. The advantage of this technique is that it is easy interpretable in contrast to other techniques such as the Naive Bayes Classifier. It is a popular symbolic classification technique used in text classification.
- **Neural Networks:** a neural network classifier is a network of units, where the input units usually represent words, the output units represent classes, and the weights on the edges between the units represent conditional dependence relations. For classifying a page, its word weights are first assigned to the input units. Then the activation of these units is propagated forward through the network and the resulting output units' value determines the class of the page. Neural networks work very well in complex domains and classification of new pages can be done fast. The drawbacks are that the training phase is slow, and that results are difficult to understand.

For a more comprehensive review of these techniques we refer to [54]. We will now take a look at the existing collaborative recommendation techniques.

4.3.3.2 Collaborative recommendation techniques

Like in the previous section where we described some techniques used for content-based recommendations, we will now describe some techniques used for collaborative recommendations. These collaborative recommendation techniques can be divided into two main types [9]: memory-based collaborative recommendations and model-based collaborative recommendations.

The *memory-based collaborative recommendation* techniques are the classic collaborative techniques which acts over the whole user database to predict which items a user will like. These techniques are characterized by the fact that they have no learning phase. Memory-based collaborative recommendation techniques are based on the k-Nearest Neighbour algorithm which we saw in chapter 3: they select a set of k most similar users, which is called a *neighbourhood*, and recommend items which these users liked. Another way of selecting a neighbourhood is by using a correlation threshold but we will not consider this approach here as it is just another way of computing the neighbourhood. In chapter 3 we

mentioned the importance of the distance metric for the KNN algorithm which will be used here to determine the similarity between users. We will therefore explore some distance metrics used to select the nearest users. These distance metrics determine whether a user will like a certain item (*prediction problem*) or recommend the N most interesting items (*top- N recommendation problem*) [35]. Memory-based collaborative recommendation techniques have the advantage that they are very simple, incorporate up-to-date information because the database is used for every new recommendation, and that they are relatively accurate [9]. The drawbacks of these techniques are the poor scalability when there are many users. These users have to be compared every time when a new prediction is made which results in a slow recommendation process.

On the other side we have *model-based collaborative recommendation* techniques which use the user database to learn a model offline which is then used for prediction. These techniques are mostly probabilistic based techniques such as Bayesian Networks which make a guess of the expected value. Other model-based techniques include clustering and rule-based approaches which we will describe later. Model-based techniques are much faster because they use a pre-computed model and are as accurate as memory-based techniques [9]. They are however not suited for quickly changing databases as this leads to an incorrect model which is not up-to-date.

Model-based collaborative recommendation techniques are also called *item-based collaborative recommendation techniques* [35] because the item similarities are rather static and can thus be computed offline [53]. This is not true for user similarities which are dynamic and require that the user has rated or purchased several items before a recommendation can be made. Hence, these similarities can not be computed offline but have to be computed online. Therefore memory-based collaborative recommendation techniques are also called *user-based collaborative recommendation techniques* [35]. We will give a brief overview of these techniques shortly but will show first how memory-based techniques work in practice.

Memory-based collaborative recommendation techniques

Memory-based collaborative recommendation techniques employ user-user similarity to construct a set of similar users. We will now describe some of these techniques based on user-user similarity. Note that although the memory-based techniques are called user-based techniques because they use user-user similarity, it is also possible to use item-item similarity. By using item-item similarities, we seek the items which are similar to the items the active user has liked, based upon the ratings of other users on these items. In [53], an item-item similarity is used but they call the technique model-based because these similarities are computed offline. If these similarities are computed online, the technique would be memory-based. The techniques that we will introduce here are therefore also applicable for item-item similarities.

There are two important aspects in the KNN approach of memory-based techniques:

1. **A distance metric algorithm:** this is needed for computing similarities between users. This class of algorithms takes two users as input (represented by vectors containing their ratings for items for example) and outputs a value which determines how similar the two users are.
2. **A prediction strategy:** once we have calculated the neighbourhood of most similar users, we need a prediction strategy to make predictions upon this neighbourhood.

We will now list some metrics and strategies which are frequently used in collaborative recommendation systems. Many algorithms have been used in practice to compute similarities; we will describe some of the ones which are frequently used. Any mathematical formulas will be omitted because they lie beyond the scope of this dissertation and therefore we refer to [53, 55, 56] for a more in dept discussion of these algorithms.

- **Correlation using the Pearson Algorithm:** this algorithm considers only the items which are rated by both the users, and calculates a weighted average of the deviations from the neighbours' mean. This is probably the most used algorithm to compute similarity and it has been shown that it performs better than the other algorithms listed below [9, 56].
- **Mean Squared Differences Algorithm:** the Mean Squared Differences Algorithm does exactly what its name says and computes the mean squared difference between two rating vectors of the users. This is an obviously simple and fast technique but it is not as accurate as the Pearson Algorithms.
- **Cosine-based vector similarity:** we already encountered this technique in section 4.3.3.1. It can be used in exactly the same as we showed there but this time by using the user rating vectors as the input.

When we have constructed a neighbourhood using a distance metric algorithm, we need to make a prediction and thus choose for a strategy to make this prediction. We will now list some prediction strategies which are commonly used among recommendation systems [42].

- **Most-frequent item recommendation:** this strategy scans through all the users in the neighbourhood and retrieves the most frequently selected items. When the most frequent items are retrieved, they are sorted by frequency and the most frequent items not yet selected by the active user are recommended. This strategy does not consider ratings and can only be used on binary valued choices such as purchases or like/dislike votes.
- **Weighted-average recommendation:** this is the standard strategy which just uses the weighted average of the user ratings in the neighbourhood. It is called weighted average because the average of the ratings is weighted using the similarity value of the users.
- **Average-deviation recommendation:** this strategy computes the average deviation of a neighbour's rating from that neighbour's mean rating over all items the neighbour has rated. This is then converted to a predictive distribution by adding it to the active user's mean rating. This

algorithm is based on the fact that users center their ratings at different points. On a scale from one to ten, for example, some users may find 6 already a high vote while others find it rather low and find 10 a high vote.

Now that we have seen how the memory-based collaborative recommendation techniques work, we will take a look at some of the existing model-based collaborative recommendation techniques.

Model-based collaborative recommendation techniques

We will describe 4 model-based algorithms which learn a model which is then used for prediction. The four algorithms are Bayesian Networks, association rules, clustering and horting. We will give short overview of these algorithms because they are too complex to go too much into detail.

A *Bayesian Network* [9] is a graphical model for probabilistic relationships among a set of variables. Bayesian Networks are also known as *Belief Networks* and also have their roots in the field of machine learning. A Bayesian Network for collaborative recommendations exists out of nodes which represent all the possible items that can be recommended. The edges in the graph are direct edges between the nodes which are the items. The network defines probabilistic relationships between the nodes: to each node conditional probabilities tables are attached encoded in the form of a decision tree which predicts the rating of the item given the ratings of its parent items. An edge from item I_1 to I_2 in a Bayesian Network means that I_1 causes I_2 which is equal to saying that I_2 is conditional dependent on I_1 . The drawbacks of this technique are that the building phase of the model can take a few hours or days and that it is not suitable when relationships among items change quickly. On the other hand, it is very small, fast and as accurate as memory-based methods.

The second model-based technique that we will describe is a rule-based approach and is called association rules [2]. *Association rules* construct a set of rules which describe the relationships or associations between a number of items, i.e. the likelihood that a set of things will happen at the same time. Association rules are used to analyze the co-occurrence of transactions within a session and make recommendations based on the strength of the associations. An example of an association rule for a movie recommendation system is the following: if a user likes the movies “The Godfather” and “Scarface” he will like the movie “Goodfellas”. The pros of this technique are that it is fast, easy to implement, does not require much storage and is not user-specific. It however suffers the same drawback as the previous technique and is not suitable when relationships change quickly.

The next technique which has a model-based approach is clustering. *Clustering* constructs groups of users which have similar interests and assign a class to each group. Determining to which cluster a user belongs is therefore a classification problem [9]. The class can be determined by estimating the probability that a particular user belongs to a particular class given its item ratings, and from there on predictions can be made by averaging the ratings of other users in that class or cluster. Clustering techniques usually produce less personal rec-

ommendations than other methods, and in some cases, the clusters have worse accuracy than nearest neighbour algorithms [9]. Just like the other model-based techniques, performance is usually very good. This is because the size of the group that must be analyzed is reduced heavily by using clusters. Some new clustering techniques [57, 60] have been proposed which try to solve the problem of sparsity where all collaborative recommendation systems deal with, i.e. the number of ratings already obtained is very small compared to the number of ratings that need to be predicted. These techniques cluster both users and items to handle this problem.

To conclude this overview of model-based techniques we will take a look at horting. *Horting* [1] is a graph-based technique in which nodes are users, and edges between nodes indicate the degree of similarity between two users. Predictions are made by walking the graph and combining opinions of similar users which are connected to each other by the nodes. The advantage of horting is that it is not a requirement that two users rate the same item to be used in predictions.

Now that we have described both memory-based and model-based collaborative recommendation techniques we will look at the third form of recommendation techniques which are hybrid recommendation techniques.

4.3.3.3 Hybrid recommendation techniques

Hybrid recommendation techniques have emerged during the years based on the fact that both content-based and collaborative recommendation techniques have methods have complementary strengths and weaknesses. Hence, combining their strengths and discarding their weaknesses leads to a new and better hybrid approach. In this section, some of these hybrid techniques which combine both content and collaborative recommendation techniques will be addressed.

The first approach we will discuss is *collaboration via content* [5, 45]. This method maintains user profiles based on content analysis. These user profiles are closely compared to determine users with similar preferences for collaborative recommendation. This way items are recommended to a user when they score highly against their own profile, and when they are rated highly by a user with a similar profile. The consequence of this approach is that we make use of the experiences of other users and are also able to recommend items which are unseen by these users.

Another approach is called content-boosted collaborative filtering [40]. In *content-boosted collaborative filtering* a *pseudo user-ratings vector* is created for every user u in the database. This vector contains the ratings of the user for all items. When a user has not rated an item, the rating is predicted using a content-based recommendation technique. Putting all pseudo user-ratings vectors of all users together gives a pseudo ratings matrix which can then be used as input for a collaborative recommendation technique. The main idea in this in this approach is that it predicts the ratings of unrated items using a content-based recommendation technique and then uses these ratings to perform collaborative filtering. It has been shown that this technique performs better

than pure content or collaboration approaches and that it overcomes problems like the sparsity and the first-rater problem[40].

In [14], Burke lists some other hybrid recommendation techniques which basically combine the results of both recommendation types in some way. A *weighted hybrid recommender* computes the predictions of both approaches separately and then uses some weighted approach to mix the predictions. Using weighted mixing, one approach can be given a higher weight and thus higher impact, or each approach can be equally weighted so that the prediction is just a linear combination of both predictions. Another combination of both recommendation types is the *switching hybrid recommender*. Based on some condition, the recommendation system switches between both recommendation types and therefore they are never used together. One way to switch is by using some confidence value for both types and use the type which has the highest confidence value. The last approach that we will describe is the *mixed hybrid approach*. This approach presents the recommended items of both types together. This approach can only be used to recommend items and not to make predictions about ratings because only one rating can be offered for one item and combining both weights would result in a weighted approach.

4.3.4 Example of a Recommendation System: Amazon.com

In this last section of our overview of recommendation systems we show how the famous online store Amazon.com makes use of a recommendation system to increase their revenues and the satisfaction of their customers. Amazon.com is the largest online retailer with more than 30 million customers and has thus all the power to build the best possible recommendation system for their business. That's the reason why we will take a look at how Amazon.com brings the theory of recommendation system into practice.

4.3.4.1 Amazon.com's user models

Amazon.com enables users to create an account so that their feedback can be used to construct a user model. This user model is then used by Amazon.com to generate personal recommendations. The user model is based on both implicit and explicit feedback from the user. The three principal forms of feedback are the following ones:

- The items purchased by the user (implicit feedback).
- The items rated by the user (explicit feedback).
- The items the user viewed (implicit feedback).

Using all this feedback, Amazon.com manages to build a solid user model that doesn't require much or any explicit feedback. Of course it's evident that explicit feedback from the user will lead to a better, more precise user model and therefore also better recommendations. But even if a user does not provide ratings for items, Amazon.com's recommendation system still succeeds to recommend interesting items to the user by using the implicit feedback that "fed" his user model.

4.3.4.2 How Amazon.com recommends items

Now that we've seen how the user model of Amazon.com's customers is constructed we will take a look at how this user model is used to recommend items to the customers. Amazon.com uses recommendations all over their web site. When a registered user visits the home page of the Amazon.com web site, the user already encounters some items which are similar to the ones he purchased, viewed or gave high ratings. The home page of the Amazon.com web site is thus tailored to each single registered user: each user has its own personalized online shop. On the top of the page, there is also a link to a personal "recommendation shop":

Hello, Stijn Coolbrandt. We have recommendations for you.

This personal recommendation shop lists all kinds of recommendations for the user and is browsable per area. Hence users can view recommendations in areas like DVD's, books and music. Amazon.com not only explicitly tells the user which items are recommended, but even offers the user the possibility to discover why these items are recommended. When a user "asks" for the reason why an item is recommended, a popup window opens with the following information:

Recommended for you ... because you were interested in: ...

This window shows the user a list of similar items which he found interesting and which form the basis of the current recommendation. It gives the user the opportunity to tell the recommendation system that he is not interested in the product or that he already owns it. These are two other kinds of feedback we didn't mention before. When a user indicates that the product doesn't interest him, it will not be recommended anymore. If he tells the system that he owns it, it will get the same recommendation value as a purchased item and will also not be recommended anymore.


Another form of recommending items used by Amazon.com is the list of similar items which can be viewed on an item page. For each item, a list of items which are commonly bought by the users who bought the item is recommended. This can be seen on figure 4.2 which is a snapshot of such a list of similar items.

A user also gets the chance to explicitly improve his recommendations. Amazon.com offers three ways to do this:

- By editing his history. This includes items he owns, items he rated and items he didn't find interesting.
- By selecting favorite areas and items in these areas.
- By rating a list of products generated by the system.

The difference with the feedback described earlier is that in this case the user takes the initiative to give feedback. Editing his history or selecting his favorite areas and items takes much more time than giving a single rating while browsing through the web site. When a system can convince a user of the importance of recommendations, it takes control of a powerful position which can be favorable for both sides.

Customers who bought this item...



Machine Learning
by Tom M. Mitchell
Average Customer Review: ★★★★★
Usually ships in 24 hours

This item ships for **FREE** with **Super Saver Shipping**. [See details.](#)

From Book News, Inc.
An introductory text on primary approaches to machine learning and the study of computer algorithms that improve automatically through experience. Introduce basics concepts from statistics, artificial intelligence, information theory, and other disciplines as need arises, with balanced coverage of... [Read more](#)


Also bought these items...

Show items from:

- ▶ **All Products**
- [Books \(20\)](#)
- [DVD \(3\)](#)



Artificial Intelligence
by Stuart J. Russell, Peter Norvig
[More like this](#)



Pattern Classification (2nd Edition)
by Richard O. Duda, et al
[More like this](#)

Figure 4.2: A snapshot of the recommendation system of Amazon.com

4.3.4.3 Amazon.com's recommendation technique

To conclude our view at the Amazon.com recommendation system we will describe which technique is used to generate recommendations. Because Amazon.com has millions of customers and items in their product catalog, scalability is a very important issue. Most of the collaborative techniques have a computing time which scales with the number of customers and items. Using these techniques requires heavy computational power and long processing time to compute online recommendations for such a huge store like Amazon.com. Therefore Amazon.com developed an own technique which scales much better than the techniques mentioned in the previous sections. This technique is called item-to-item collaborative filtering and is described in [39]. It is a model-based collaborative recommendation technique and uses item-item similarity instead of user-user similarity. This means that rather than comparing users for similarities, items which the user purchased and rated are compared to other items for similarities and the most similar items are then used for making recommendations.

For computing similarities between items, a similar-items table is constructed [39]. This table is constructed as follows:

```

For each item in product catalog  $I_1$ 
  For each customer  $C$  who purchased product  $I_1$ 
    For each item  $I_2$  purchased by customer  $C$ 
      Record that a customer purchased  $I_1$  and  $I_2$ 
  For each item  $I_2$ 
    Compute the similarity between  $I_1$  and  $I_2$ 

```

How the similarity between items is computed by Amazon.com is unknown. In [39], they suggest to use the cosine measure but they do not explicitly state that

it is this distance metric which Amazon.com uses. The dimensions of the vector which represents an item correspond to the users who purchased that specific item. Computing the similar-items table is very time consuming but because it considers the relationships between items, this can be done offline (that's why this technique is model-based). Recall that this is due to the fact that the relationships between items is relatively static in contrast to the relationship between users. Once this table is constructed offline, looking up items similar to the items purchased and rated by the users can be done very fast. This lookup process is furthermore only dependent on the number of items the user purchased and rated which makes it more scalable than any other technique we have seen.

To underline the force of the technique developed by Amazon.com we list the advantages of this collaborative recommendation technique:

- Similarities can be computed offline. This is really a must when a large amount of data is available.
- Because of computing similarities offline, its performance is extremely well. This is because the online process only needs to lookup items similar to the ones the user purchased and rated.
- It scales very well because the online recommendations process is only dependent on the number of items the user purchased and rated.
- The recommendation quality is excellent because it only recommends highly similar items.
- The technique also works very well when limited user data is available.

The description of this technique concludes this chapter. We will now turn theory into practice in the research part of this dissertation.

Part II

Research

Chapter 5

Integrating Link Recommendations into WSDM

5.1 Introduction

This chapter describes how link recommendations will be integrated into WSDM. We will propose extensions to WSDM and ASL such that they are able to support dynamic adaptation at the user session level. Once that we have proposed these extensions we will use these extensions to build a recommendation system using ASL.

The current version of ASL already supports web site optimization, i.e. adaptation which aims at accommodating all visitors. This optimization adapts the structure of the web site based on its usage in order to improve the web site for all users. We have already seen examples of such kind of adaptation in section 4.2.3 of chapter 4. These examples include promotion, demotion, linking and clustering of nodes and can already be easily implemented using ASL. In contrast to the optimization techniques which adapt the structure of the web site permanently for all users, we want to adapt the structure of the web site temporally and separately for each user. The extensions to both WSDM and ASL needed for this adaptation at the user session level will be described in section 5.2.

A big advantage of the audience driven design of WSDM is that no user model is needed. Because the web site is designed with the target audience in mind, a visitor should be able to find all the necessary information in the audience track of the audience class to which he belongs. When the web site is designed in a proper way, there is thus no need for a user model. A user model is often used by web sites which are designed in a data driven way. These web sites don't take in account the target users and don't structure their web site according to the needs of the target users. Because of this data driven design, user models are frequently used to determine the preferences of the users and to

personalize the web site to the users' needs, something which is done in WSDM at design time in the Audience Modelling phase.

Undoubtedly, using a user model for each visitor often leads to more specific characteristics for each visitor than the division of users into audience classes which leads to general characteristics for the users in each audience class. This minor drawback is however compensated by the fact that WSDM avoids the problems which arise when building and updating a user model. Maintaining a user model is often a demanding task for both the designer and the user of the web site.

The designer first has to design a user model which is capable of representing the user properly. This gives rise to many questions. Should the user's interests be stored in the user model? What about his specific preferences? How will all information such as preferences and interests be retrieved? Even when a good user model is constructed, the designer also has to specify how this information will be used. This can be by providing links to pages which are interesting according to his user model, or by altering text fragments based on the knowledge level of the visitor. There is a wide range of personalization methods as we have seen in chapter 4. This troublesome task of designing and processing a user model is nonexistent for the WSDM designer which obviously saves him a lot of work.

What is even more important is that visitor is not saddled with the user model. The utilization of a user model often implies that visitors need to log in to the web site and need to provide information and feedback. Users often don't want to provide feedback and get dissatisfied when they don't notice any benefits. Filling in large forms with personal information is not something all user like to do, either because of the effort it requires or because of privacy issues. When WSDM is used, users don't have to spend all these additional efforts like logging in or providing feedback.

The fact that there is no real user model which can be used across sessions poses an interesting challenge: how can we generate useful link recommendations at the session level? To be able to generate personal link recommendations for each visitor, we will extend ASL with the ability to track node accesses separately for each session. We can then use this session information to generate link recommendations based on the nodes a visitor accessed in his current session and relate this information to the information of previous sessions.

Both link recommendation techniques which we will present are collaborative link recommendation techniques because they are based on the browsing history of other similar users. Pages are recommended which are commonly visited by visitors with a similar browsing history. Because the recommendations are only based on the pages the visitors viewed, no explicit feedback is required which is a big advantage. The link recommendation techniques also act in a closed corpus and therefore recommend only links to pages within the web site. These recommendations will be generated separately for each visitor by taking his session information as input and comparing it to the available session data of the other sessions. When a visitor accesses a page on the web

site, link recommendations are generated and presented to the visitor. These link recommendations are generated by computing an interestingness value for each unvisited page. Whenever this value is larger than a certain threshold, the page is recommended.

The two techniques which we will present, generate new link recommendations whenever a page in the web site is requested by the user. This can be seen as a refreshment of the link recommendations which is justified by the change in the session information of the current user, i.e. a new page view is added to the session information. To clarify this refreshment strategy, imagine browsing through a news site. When we are viewing a news page in the world news section, links will be recommended which are related to this page. When a new page is accessed, for example in the sports section, many new link recommendations related to sports will be generated. Every new page which is accessed can therefore lead to new different link recommendations. That's why link recommendations are generated every time a new page is visited. Both of the link recommendations techniques which we will describe, follow this refreshment strategy. The only thing that distinguishes them is in how they determine which pages are interesting.

We will first propose the needed extensions to WSDM and ASL in section 5.2 such that the link recommendation techniques can be implemented in ASL. In section 5.3, we will propose our two collaborative link recommendation techniques based on Bayes' Theorem and the k-Nearest Neighbour algorithm.

5.2 Extensions to WSDM

To be able to specify at design time which dynamic adaptation should occur at the user session level we need make some extensions to both WSDM and ASL. These are not major extensions but nonetheless crucial in order to implement link recommendation techniques. The extensions to WSDM in section 5.2.1 are situated on the Conceptual Design level or are more specific, extensions to the Navigation Model. ASL will be mainly extended with operations that track and access session specific information. These extensions will be proposed in section 5.2.2.

5.2.1 The Navigation Model

We now return to the Navigation Model which was already discussed in section 2.3.1 of chapter 2.1. The Navigation Model is constructed during the Navigation Design phase which is the second sub-phase of the Conceptual Design phase. It is a graph which defines the conceptual structure of the web site and which consists of nodes and chunks. Recall that the edges between the nodes in the graphs are called links, and the edges between the nodes and chunks in the graph are called connections. All these links and connections are static and the links are traversable by each visitor. When we want to generate personal recommendation links for each visitor, we can not do this by just adding a new link because then all other visitors would see this new link too. This is the reason why we need to extend the Navigation Model so that we are able to add

links which are only visible for one visitor.

We will extend the Navigation Model with a new kind of links which are temporal and bound to a specific user session. The lifetime of these links is the same as the lifetime of the session to which they are bound, unless they are deleted explicitly. When the session ends, all links belonging to this session will be deleted. We will therefore call these links *session links*. The session links are almost identical to the normal permanent links, the only difference is that they are only visible within the session to which they are bound. Using the extended version of ASL which will be proposed in the next section, the designer will be able to specify at design time which session links should be constructed at runtime during each session. Note that the introduction of these session links does not alter the Navigation Model at design time but only temporally at runtime for each visitor separately.

On figure 5.1, we can see an example of an extended Navigation Model. It extends the simplified Navigation Model of the WISE web site we saw earlier in chapter 2.1 with some session links. The dotted lines with the hollow arrows are the session links. They are labeled by the sessions to which they are bound to. We can see that for *session 1*, there are two session links from the root node of the *Researcher Track* to respectively the *Members* and the *About Wise* node. Notice that there are two session links from the root node of the *Researcher Track* to the *Members* node, each bound to a different session.

By extending the Navigation Model with session links, we are now able to present personalized links to each visitor. Session links can be added and deleted during a browsing session of a visitor. The explicit deletion of session links is necessary for the refreshment strategy that we will employ in our link recommendation techniques. When these links would be permanent across a session it would be only possible to generate link recommendations once for each node, or to add link recommendations incrementally.

Just as for the normal permanent links, there will be four types of session links. For providing link recommendations, we will add navigation aid session links which is obvious as the recommendations are meant as an additional aid for the users. Now that we have proposed the extensions to the Navigation Model we will show how these session links can be added and deleted using ASL.

5.2.2 ASL

As we have seen in section 2.3.3 of chapter 2.1, ASL makes it possible to perform operations on the Navigation Model at runtime. We have seen how the structure of the site can be changed using operations like deleting and adding links. Because we have extended the Navigation Model with the notion of session links, we have to extend ASL too so that it provides the necessary operations to add and remove session links. Another extension which we will propose include operators which serve to access stored session information.

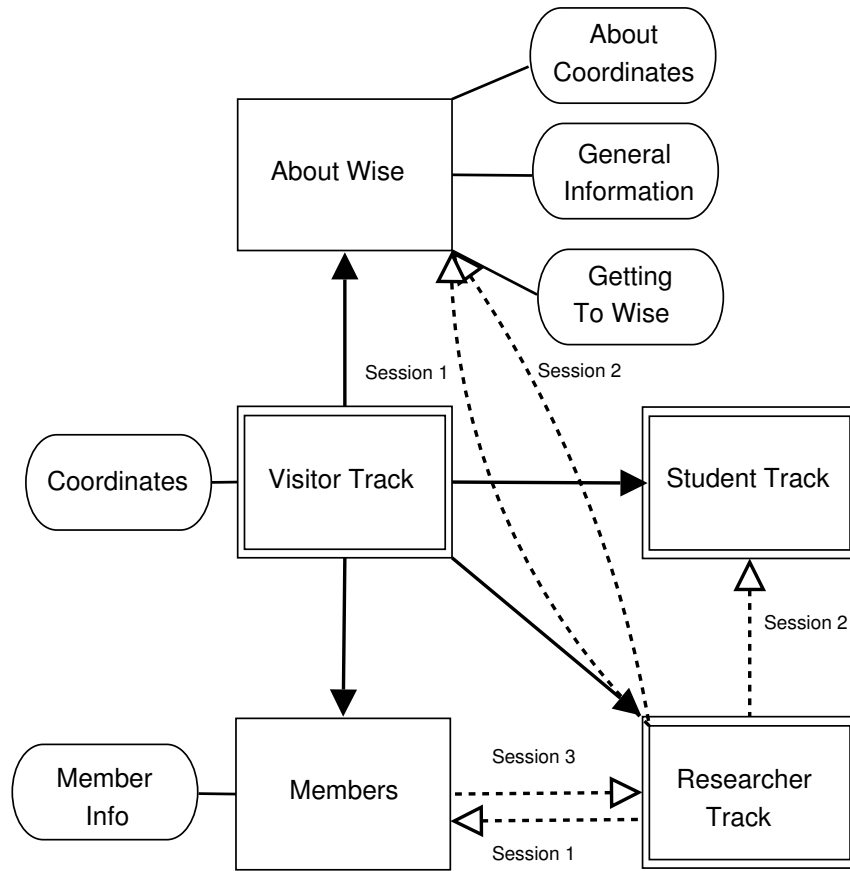


Figure 5.1: An example of an extended Navigation Model.

5.2.2.1 Extension 1: new native operations

We will start with describing the new native operations which will act upon session links. The existing native operations are extended with three new ones: adding a session link to a session, deleting a session link from a session and clearing all session links of a given session. The addition of session information to ASL is a new extension that will be introduced in section 5.2.2.3. For now, the only thing we need to know is that referring to a session can be done in the same way as referring to nodes or chunks. In section 5.2.2.3, we will see how the session argument of these new elementary operations can be retrieved.

The first operation which adds a session link to a session has four arguments: the type of link, the target session to which the link should be added and the source and target node of the link. The second new native operation deletes a session link and requires four arguments: the type of link, the target session and the source and target node of the link. To be able to clear all the session links of a given type and belonging to a certain session with only one operation we have also added an operation which takes the given type and session as input.

A summary of all these new native operations can be seen in table 2.1.

| Operation Semantics | Syntax |
|---|--|
| Add a session link of type t from node n_1 to node n_2 , for session s . | <code>addSessionLink(t, s, n₁, n₂)</code> |
| Delete a session link of type t from node n_1 to node n_2 , for session s . | <code>deleteSessionLink(t, s, n₁, n₂)</code> |
| Clear all session links of type t for the given session s with the given ID . | <code>clearSessionLinks(t, s)</code> |

Table 5.1: The new native operations added to ASL.

5.2.2.2 Extension 2: new native functions

We have already seen the *currentAudienceClass()* native function provided by ASL. We will add another native function called *currentSession()* which returns the session of the current user session. This new function returns a session object holding session information. Using this new function we can, for example, attach session dependent tracking variables to nodes. We will now demonstrate how this new function can be used to track all node accesses within a session by using a tracking variable.

To track node accesses within a session we add a tracking variable *accesses* to each node as follows:

```

forEach node in Nodes:
  begin
    currentSession := currentSession();
    addTrackingVariable node.accesses;
    monitor load on node
    do node[currentSession].accesses := node[currentSession()].accesses + 1
  end

```

This piece of code attaches a tracking variable called *accesses* to each node in the web site. Instead of maintaining the node accesses or clicks of all visitors like we did for the *amountOfAccesses* tracking variable in chapter 2.1, we will track node accesses for each visitor session separately. Each time a node is loaded (accessed by a visitor), the body of the monitor statement is executed. Whenever a node is loaded by a visitor, the *accesses* variable relative to the session of the visitor is increased. This can be interpreted as follows: the number of accesses by session s to node n is equal to $n[s].accesses$. Notice that the *currentSession()* function can only have a valid value when it is used within the control flow of a monitor statement which specifies an action on a certain event. When the *currentSession()* is used outside a monitor statement, it is not possible to resolve to which session it is bound as no event is specified. Using the above tracking variable we are now able to check how many times a node is accessed in a session.

Another native function which is added is *currentNode()*. This function returns the currently loaded node in an active session. This function is however only available when a script is attached to a node and is executed when that

node is loaded. When this script is loaded, the node to which the script is attached is available within the script using the *currentNode()* function.

5.2.2.3 Extension 3: a new native set

The next extension which we will propose is a new native set called *Sessions* which contains all the sessions of the web site. This set contains all the sessions which have been tracked during the lifetime of the web site. By iterating over these sets we can access session dependent monitors.

The following code gathers the total amount of accesses by using the *accesses* monitor defined above:

```
begin
totalAmountOfAccesses := 0;
forEach node in Nodes:
    foreach session in Sessions:
        totalAmountOfAccesses :=
            totalAmountOfAccesses + node[session].accesses;
end
```

This example shows that we use the elements in the *Sessions* set to retrieve session dependent monitor values such as the *accesses* monitor. Note that the sessions in the *Sessions* set are ordered sequentially which means that they are ordered according to the time on which they started.

Along with this new native set we extend the *audienceClass* function so that it can also take a session as input and return the audience class of this session.

5.2.2.4 Extension 4: a new set creator

Another extension to ASL is a set creator called *subset* which takes a set as an argument and returns a subset of the set. The *subset* operator needs two numbers which indicate which subset of elements should be selected out of the set. We will again demonstrate this new set creator with the following example:

```
begin
sortedNodes := Nodes[SORT on element : element.amountOfAccesses];
fiveMostPopularNodes := subset(sortedNodes, 1, 5);
end
```

The example constructs a set with the 5 most popular nodes by first sorting the nodes on the amount of accesses and then taking the 5 first elements of this set.

5.2.2.5 Extension 5: revising the monitor statement

Our last extension involves the monitor statement. The current version of ASL only allows to update tracking variables. Because we want to be able to execute more complex operations whenever some event occurs, we will allow any adaptation strategy to be specified.

All these new extensions will be used in the next section in which two link recommendation techniques will be proposed. They can also be found in the

appendix A which contains the ASL BNF specification. These extensions are emphasized in italics.

5.3 Proposed Recommendation Techniques

5.3.1 Link recommendations using a KNN algorithm

5.3.1.1 Introduction

The first link recommendation algorithm is a KNN like algorithm and is similar to the memory-based recommendation techniques which is described in section 4.3.3.2 of chapter 4. This *k nearest neighbour link recommendation* algorithm will determine the *k* most similar visitors using some distance metric and will then recommend the pages which are most commonly visited by these *k* nearest visitors.

As an introduction to the algorithm, we will briefly discuss the main structure of the algorithm using pseudo-code. The algorithm can be summed up as follows:

```

FETCH_INTERESTING_LINKS(WSP, t, S, k, DM)
  VP = get visited pages of current session using S
  UP = {WSP / VP}
  KNearestSessions = Get the k nearest sessions to
                      current session using DM and S
  For each up in UP do:
    p = Calculate percentage of sessions in
        KNearestSessions which visited up
    If p > t
      Recommend up

```

There are five input arguments for this algorithm. The first one, *WSP*, contains the set of all pages in the web site. The second argument, *t*, is the threshold that should be reached to recommend a page to the visitor and the third argument, *S*, contains the session data. A simple session matrix like the one illustrated in table 5.2 is sufficient for the algorithm as we only need to know which sessions accessed which pages. In this table, visited and unvisited pages are marked in their corresponding cells by a 1 and a 0 respectively. The fourth argument, *k*, is obviously the *k* value which determines how many neighbours will be considered for recommending nodes. The last argument, *DM*, is the distance metric that will be used to determine the *k* nearest visitor sessions. By taking this metric as an input argument, we can easily change the used metric in a plug and play kind of way. We will see shortly how this is done in practice.

Let's take a look at the pseudo-code. First, we need to know which pages were accessed in the current session. This information can be easily retrieved from the session matrix *S*. On the second line, the pages which are not yet visited are retrieved and stored into *UP*. The next line retrieves the *k* nearest visitor sessions using the distance metric *DM*. The *forEach* loop which follows calculates how many of the *k* nearest sessions visited the unvisited page and

| | Web site pages | | | | | |
|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| Sessions | P ₁ | P ₂ | P ₃ | P ₄ | P ₅ | P ₆ |
| S ₁ | 1 | 0 | 0 | 1 | 1 | 1 |
| S ₂ | 1 | 1 | 1 | 0 | 0 | 0 |
| S ₃ | 1 | 0 | 1 | 1 | 0 | 0 |
| S ₄ | 1 | 0 | 0 | 0 | 1 | 1 |

Table 5.2: An example of a simple session matrix.

recommends the page to the current visitor if that percentage is above the pre-defined threshold. This is done for each page which is not yet visited by the given visitor.

The value which determines the interestingness of the page is easy to calculate as it is the percentage of sessions which accessed the page, i.e. the number of sessions which accessed the page divided by the total number of sessions which is in this case k as only k sessions are considered. This interestingness is the popularity of the page among the k nearest sessions. Note that the way of selecting the most interesting page resembles much to the most frequent-item recommendation techniques which we saw in section 4.3.3.2 of chapter 4. Both techniques select the pages or items which are frequented the most by the k nearest sessions.

The recommendation process only considers pages which are not yet visited by the user in the current session. This is an obvious choice as we would otherwise recommend the same visited pages multiple times throughout one session.

Notice also that we are talking about pages and not about nodes. When we will discuss the algorithm expressed in ASL however, we will talk about recommending nodes because links in WSDM are between nodes and not between pages. Because we don't want to narrow this technique down to the context of WSDM we will, for now, talk about recommending pages which is more general.

5.3.1.2 Using the KNN algorithm to determine interesting pages

This section explains some aspects of the algorithm more in detail and shows how it works using an example. First we need to clarify an important aspect: which metric is used to determine the distance between two sessions?

The distance metric is the most important aspect of the algorithm as it resolves which sessions will be used to make recommendations. The goal is to identify sessions which are most similar to the current session. Because we want to keep the session data simple and small, the same simple session matrix as the one that was shown earlier on in table 5.2 will be used. The fact that we use such a simple session data format, diminishes the number of ways of comparing sessions to each other somewhat. We can not, for example, use the time visitors spent on pages to compare sessions against each other. We can only use the information that a page is visited or unvisited within a session. This leads us to

an uncomplicated distance metric which places visitors which have visited the same pages close together. The following pseudo-code shows how the chosen distance metric works:

```

MEASURE_DISTANCE(S1, S2)
  VP = get pages visited by S1
  d = 0
  For each vp in VP do:
    If vp is not visited by S2
      d = d + 1
  Return d / LENGTH(VP)

```

This function is the *DM* argument of the *k* nearest neighbour link recommendation algorithm and computes the distance by taking two sessions as input arguments and outputting the distance between these two sessions. These session arguments can be seen as the rows of a session matrix like the one we saw earlier on. Note that the first argument needs to be the target session, the one for which we want to recommend pages. This is important because, for the computation of the distance, we are only interested in the pages which the current visitor has visited. If this current visitor has visited 3 pages so far and its session is compared to an old session in which 15 pages were visited, we are only interested in whether the old session visited these same 3 pages too. In the other case, when the old session would be the first argument, we would check if these 15 pages have been visited by the target session. This can only lead to a low distance because only 3 of these 15 pages can possibly have been visited. Therefore, the order of the arguments is very important for this function.

On the first line of the body of the function, all the pages visited by the first session are retrieved. Then, the distance is initialized using the variable *d* to store this value. On line three to five, a *forEach* construct computes the distance between the sessions. For each page that is visited during the first session, we test if it is also visited in the second session. If this is not the case, the distance is increased. When the *forEach* construct ends, the distance is returned and divided by the number of pages which are compared in the *forEach* construct. This way the distance will always be between 0 and 1. Sessions which visited exactly the same pages as the first session will get 0 as distance and sessions which have no visited pages in common will have 1 as session distance.

When the proposed distance metric is used, it is easy to retrieve the *k* nearest sessions by selecting the *k* sessions with the smallest distance to the target session. These *k* nearest sessions are then used to determine the page popularity (or interestingness) for each page which is not yet visited in the current session. We will again use pseudo-code to show how this popularity is determined. The function which calculates this popularity is as follows:

```

MEASURE_POPULARITY(p, KNS)
  pop = 0
  For each s in KNS do:
    If p is visited by s
      pop = pop + 1
  Return pop / LENGTH(KNS)

```

The two input arguments of this function are the page p and the session data matrix KNS containing the k nearest sessions. What we are interested in, is the popularity of this page among these k sessions. We measure this popularity simply by counting the number of sessions which visited p and dividing this number by the number of sessions, which is the k value. The first line of the function initializes the pop variable which is then incremented each time a session visited the page in the next three lines. The last statement returns the popularity but divides it by the total number of sessions first. A page which is visited by all sessions will get a popularity of 1 and a page which is visited by none of the sessions will get a popularity of 0.

We now have all ingredients for this algorithm. During the design of the algorithm, two important choices were made:

1. How are the k nearest sessions selected, or which distance metric is used?
2. How is the popularity of pages in these k nearest sessions determined?

We have chosen for very simple answers to these questions. There are many other plausible answers which will give good results. The goal of this dissertation is however not to find the best answer because there is simply not enough time and space to test all possible combinations of answers. That's why we opted for two simple answers which are solid enough to achieve the real goal of this dissertation: incorporating link recommendation techniques into WSDM.

Example of a popularity computation

We will now show how this algorithm works in practice using an example. The purpose of the example is not to evaluate the k nearest neighbour link recommendation algorithm but only to give the reader a chance to better grasp the technical side of the algorithm. This is why we opted for an imaginary web site and imaginary visitor sessions. Because less is more in this case (less data to absorb will result in a higher comprehensibility), the size of the web site is very small (i.e. there are few pages in the web site) and the number of sessions is also very low. A map of this non-existing web site can be seen on figure 5.2. This is a simple web site of an imaginary musical trio called "The Recommenders". The web site of "The Recommenders", for simplicity, only contains typical basic information about their musical career. Note that this web site is not constructed using WSDM because the design method used to design this web site is not relevant for clarifying this algorithm.

As can be seen on figure 5.2, there are 12 pages represented by rectangles. The edges between the pages are the links between pages. To have a shorter reference to each of the pages, they are labelled by the diamonds in the top left corner of each rectangle. The labels of the pages will now be used in the session data table. Table 5.3 shows some session data for 9 visitor sessions. For each session, we can see which pages are visited and which are not. Visited pages are checked, unvisited pages are obviously unchecked. The last session, session 9, will be used to demonstrate the popularities of the unvisited pages are computed.

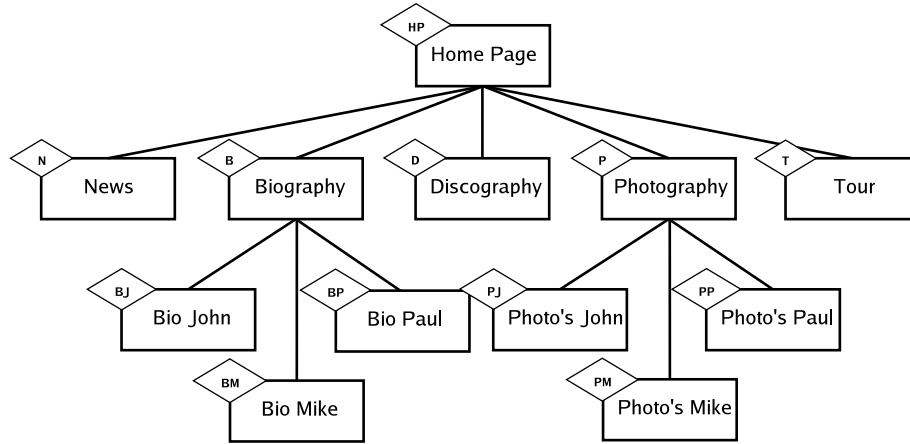


Figure 5.2: Map of the web site of “The Recommenders”.

| | Web site pages | | | | | | | | | | | |
|----------|----------------|---|---|---|---|---|----|----|----|----|----|----|
| Sessions | HP | N | B | D | P | T | BJ | BM | BP | PJ | PM | PP |
| 1 | ✓ | ✓ | ✓ | ✓ | | ✓ | | ✓ | | | | |
| 2 | ✓ | | | | ✓ | | | | | | | ✓ |
| 3 | ✓ | ✓ | | ✓ | ✓ | | | | | | | ✓ |
| 4 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | ✓ | ✓ | | ✓ | ✓ |
| 5 | ✓ | | | ✓ | | | | | | | | |
| 6 | ✓ | ✓ | ✓ | | ✓ | ✓ | ✓ | | ✓ | ✓ | | ✓ |
| 7 | ✓ | ✓ | | | ✓ | ✓ | | | | ✓ | ✓ | ✓ |
| 8 | ✓ | ✓ | ✓ | | ✓ | | | ✓ | ✓ | | ✓ | ✓ |
| 9 | ✓ | ✓ | ✓ | ? | ✓ | ? | ? | ? | ? | ? | ? | ? |

Table 5.3: The session data for the web site of the “The Recommenders”

We will demonstrate the algorithm by computing the page popularity of the “Tour” page for session 9. For the other pages, we will just give the result of this computation and replace the question marks in the table below by their respective values. Note that these sessions all entered the web site via the “Home Page” page and that the table thus not includes sessions which entered the web site using search engines or bookmarked pages for example. This is a minor simplification that doesn’t have any effects on the computation whatsoever.

The value of k is the first thing we need to choose. Because the number of available sessions is small, we will choose a small k value: 4. Now that we have set this value it’s time to measure the session distance of session 9 to each of the other 8 sessions. Four pages are already visited in the ninth session: the “Home Page”, “News”, “Biography” and the “Photography” page. To measure the session distance of session 9 to the others, we have to test which of these other sessions visited these pages too and compute the distance using the metric we have seen. The session distance to session 9 for all other sessions can be seen in the last

| | Web site pages | | | | Distance |
|----------|----------------|---|---|---|--------------|
| Sessions | HP | N | B | P | To session 9 |
| 1 | ✓ | ✓ | ✓ | | 0.25 |
| 2 | ✓ | | | ✓ | 0.50 |
| 3 | ✓ | ✓ | | ✓ | 0.25 |
| 4 | ✓ | ✓ | ✓ | ✓ | 0 |
| 5 | ✓ | | | | 0.75 |
| 6 | ✓ | ✓ | ✓ | ✓ | 0 |
| 7 | ✓ | ✓ | | ✓ | 0.25 |
| 8 | ✓ | ✓ | ✓ | ✓ | 0 |

Table 5.4: The session distances to session 9.

column of table 5.4. Only the columns of the relevant pages are shown in this table so that it's not overloaded with the redundant information of the other pages. The distance of session 1 to session 9, for example, can be computed as follows: only the "Photography" page is not visited in this session and thus is the session distance 1 divided by the number of pages which is 4. This results in a distance of 0.25.

Selecting the 4 nearest neighbour sessions is now quite easy using table 5.4. Sessions 4, 6 and 8 have a session distance of 0 and have thus each visited all the pages which are visited by session 9. For the fourth nearest session we have three sessions with the same session distance: session 1, 3 or 7. The distance metric does not really provide a solution for choosing a session out of these three sessions. The algorithm simply sorts the sessions using the session distance from 0 to 1 into a list and then takes the first k sessions. When there are multiple sessions left with same session distance, the choice among these sessions is at random. Keeping in mind that the we have to select one of the three sessions at random we will select session 7 at random. The four nearest sessions which will be used in this example are therefore session 4, 6, 7 and 8.

Now that the 4 nearest sessions have been selected, our next task is to compute the popularity of the "Tour" page. To compute this popularity we only have to look up in table 5.3 which sessions visited the "Tour" page and divide this number by the total number of sessions. In table 5.3, we can see that the "Tour" page has been visited during session 4, 6 and 7 but not during session 8. This leads to a popularity of 75% as 3 out of the 4 sessions visited the "Tour" page. Doing this for all other pages which are not yet visited by the ninth session we get the page popularities listed in table 5.5. This table only shows the 4 nearest sessions which have been used to determine the popularity of the pages. The popularity of each page can be easily verified by counting in each column the number of cells that are checked and by dividing this number by 4.

To emphasize the impact of the value of k , table 5.6 shows the page popularities when we only consider the three nearest neighbours for determining the page popularity. This gives slightly different values for the page popularities. In this case, the difference is not really big but when thousands of sessions are available, the choice between a k value of 20 and a k value of 100 can lead to

| | Web site pages | | | | | | | | | | | |
|----------|----------------|---|---|----|---|----|----|----|----|----|----|-----|
| Sessions | HP | N | B | D | P | T | BJ | BM | BP | PJ | PM | PP |
| 4 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | ✓ | ✓ | | ✓ | ✓ |
| 6 | ✓ | ✓ | ✓ | | ✓ | ✓ | ✓ | | ✓ | ✓ | | ✓ |
| 7 | ✓ | ✓ | | | ✓ | ✓ | | | | ✓ | ✓ | ✓ |
| 8 | ✓ | ✓ | ✓ | | ✓ | | | ✓ | ✓ | | ✓ | ✓ |
| 9 | ✓ | ✓ | ✓ | 25 | ✓ | 75 | 25 | 50 | 75 | 50 | 75 | 100 |

Table 5.5: The session data extended with page popularities for $k=4$.

| | Web site pages | | | | | | | | | | | |
|----------|----------------|---|---|----|---|----|----|----|-----|----|----|-----|
| Sessions | HP | N | B | D | P | T | BJ | BM | BP | PJ | PM | PP |
| 4 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | ✓ | ✓ | | ✓ | ✓ |
| 6 | ✓ | ✓ | ✓ | | ✓ | ✓ | ✓ | | ✓ | ✓ | | ✓ |
| 8 | ✓ | ✓ | ✓ | | ✓ | | | ✓ | ✓ | | ✓ | ✓ |
| 9 | ✓ | ✓ | ✓ | 33 | ✓ | 66 | 33 | 66 | 100 | 33 | 66 | 100 |

Table 5.6: The session data extended with page popularities for $k=3$.

very distinctive page popularity values. Determining a suitable k value is a very difficult task. We will return to this subject in section 5.3.1.4.

5.3.1.3 The algorithm expressed in ASL

In this section we will express the k nearest neighbour link recommendation algorithm in ASL. The algorithm itself will be implemented as a script which can then be used in an adaptation policy. Recall from the pseudo-code that the algorithm takes the distance metric as an input argument. An advantage of taking the distance metric as an input argument is that it can be changed independently of the algorithm. A minor drawback is that the distance metric needs to be specified outside of the algorithm to make the algorithm work.

Before we can discuss the algorithm, we have to set up a tracking variable which monitors the accesses of a session to a node. We will use the *accesses* tracking variable which we already defined in section 5.2.2:

```

forEach node in Nodes:
  begin
    addTrackingVariable node.accesses;
    monitor load on node
    do begin
      currentSession := currentSession();
      node[currentSession].accesses :=
        node[currentSession].accesses + 1
    end
  end

```

The problem with ASL is that it is not possible to write a function which measures the distance between sessions. A script could be used to measure the

distance between sessions but scripts can not return values. A solution for this problem is the use of a session tracking variable which holds the distance between a session and the other sessions. Every time that a session loads a new node, the session distance from that session to all other sessions is computed. By computing the distance only when nodes are loaded, the distance is only updated for active sessions. This session distance tracking variable is not needed for terminated sessions because these sessions obviously do not require any recommendations anymore. The ASL code shown in algorithm 1 creates a tracking variable and monitor which updates the session distance for the active sessions.

The script shown in algorithm 1 can be explained as follows:

- **Line 1:** the algorithm is declared with as name *trackSessionDistance*.
- **Line 3 to 7:** a tracking variable called *sessionDistance* is added for each new session which starts. The *currentSession* function returns the started session as it is bound to the monitored session.
- **Line 9 to 29:** these lines specify what happens when a node is loaded within a session. Whenever a node is loaded within a session (line 8), this code is executed.
 - **Line 10:** the output of the *currentSession* function is assigned to a variable in order to avoid calling the *currentSession* function more then once.
 - **Line 11 to 16:** all the nodes which are visited in the session which loaded the node are collected. The browsing history of this visitor is thus constructed and maintained in the list of visited nodes. It is quite straightforward how this happens: a node which is accessed once or multiple times is added to this list.
 - **Line 17:** initializes a variable that holds of the number of visited nodes by the given session.
 - **Line 18 to 28:** this *forEach* construct computes the session distance to all other sessions.
 - ◊ **Line 20:** the variable *countPages* is initialized. This variable will hold the number of pages that are also visited by the session to which the current one is compared.
 - ◊ **Line 21 to 25:** the computation of the value of the *countPages* variable happens in this *forEach* construct. These lines increase the value of the *countPages* variable for each node which isn't accessed (lines 23-24).
 - ◊ **Line 26:** the *countPages* variable is divided by the number of visited nodes and is assigned to the *distance* variable. This variable now holds the distance between the two compared sessions.
 - ◊ **Line 27:** assigns the value of the *distance* variable to the *sessionDistance* tracking variable from session *currentSession* to session *s*.

This distance metric script should be loaded in the main ASL script which will also load the k nearest neighbour link recommendation algorithm. The following code enables the execution of the distance metric algorithm:

Algorithm 1 The distance metric script.

```

1: script trackSessionDistance():
2: begin
3:   monitor sessionStart
4:   do begin
5:     currentSession := currentSession();
6:     addTrackingVariable currentSession.sessionDistance;
7:   end;
8:   monitor load on node
9:   do begin
10:    currentSession := currentSession();
11:    visitedNodes := {};
12:    forEach n in Nodes:
13:      begin
14:        if n[currentSession].accesses > 0
15:        then visitedNodes := visitedNodes + {n}
16:      end;
17:    numberOfVisitedNodes := length(visitedNodes);
18:    forEach s in Sessions:
19:      begin
20:        countPages := 0;
21:        forEach visitedNode in VisitedNodes:
22:          begin
23:            if visitedNode[s].accesses = 0
24:            then countPages := countPages + 1
25:          end;
26:        distance := countPages / numberOfVisitedNodes;
27:        currentSession[s].sessionDistance := distance;
28:      end
29:    end
30: end

```

call trackSessionDistance()

We can change the body of algorithm 1 easily without changing the k nearest neighbour link recommendation algorithm. This is the strength of separating the distance metric from the rest of the code of the algorithm. Another advantage is that the distance metric can also be reused by any other algorithm which needs this distance between sessions.

Now it's time to move to the k nearest neighbour link recommendation algorithm itself which can be seen on algorithm 2. The algorithm is implemented as a script with as name *KNNLinkRecommendations*. The script will be executed each time a node is loaded so that link recommendations are generated for each node visit. This is done as follows:

```

forEach node in Nodes:
  monitor load on node
  do call KNNLinkRecommendations(10, 0.50)

```

The code of the k nearest neighbour link recommendation algorithm can be explained as follows:

Algorithm 2 The k nearest neighbour link recommendation algorithm.

```

1: script KNNLinkRecommendations( $k$ , threshold):
2: begin
3:   currentSession := currentSession();
4:   clearSessionLinks(navigationAid, currentSession);
5:   visitedNodes := {};
6:   forEach  $n$  in Nodes:
7:     begin
8:       if  $n$ [currentSession].accesses > 0
9:       then visitedNodes := visitedNodes + { $n$ }
10:    end;
11:   numberOfVisitedNodes := length(visitedNodes);
12:   sortedSessions := Sessions[SORT on element :
13:     currentSession[element].sessionDistance];
14:   KNearestSessions := subset(sortedSessions, 1,  $k$ );
15:   unvisitedNodes := Nodes DIFFERENCE visitedNodes;
16:   forEach  $u$  in unvisitedNodes:
17:     begin
18:       popularity := 0;
19:       forEach  $s$  in KNearestSessions:
20:         begin
21:           if  $u$ [ $s$ ].accesses > 0
22:           then popularity := popularity + 1
23:         end;
24:       popularity := popularity /  $k$ ;
25:       if popularity > threshold
26:       then addSessionLink( $nA$ , currentSession, currentNode(),  $u$ )
27:     end
28: end

```

- **Line 1:** the algorithm is defined as a script which takes the value of k and the threshold value as input arguments. This threshold value determines when a node should be recommended: a session link to this node will be created when the popularity of the node is larger than this threshold value. Note that the pseudo-code required five input arguments in contrast to the two input arguments of the ASL version. The reason why is because the distance metric, session data and nodes in the web site are already available in ASL.
- **Line 3:** stores the output of the *currentSession* function into a variable.
- **Line 4:** indicates that all navigation aid session links (which are the link recommendations) belonging to the visitor session which invoked the node load should be cleared. To clarify why this operation is necessary consider the following example scenario of a visitor session on the web site of “The Recommenders”:
 1. A visitor enters the web site via the “Home Page” page.
 2. He accesses the “Biography” page and gets “Bio Paul” page as recommendation.
 3. He loads that “Bio Paul” page.

4. He returns to the “Biography” page.

When the recommendations which the visitor received at the second step at the “Biography” page would not be cleared, he would get again the “Bio Paul” page as a recommendation. Therefore this recommendation or session link should be cleared, which goes for all other session links too. Even when there is a session link to a page which is not yet visited, the probability would not match the current probability as the browsing history has changed due to the fact that the “Bio Paul” page was visited. This explains the necessity of the *clearSessionLinks* operation.

- **Line 5 to 11:** these lines are exactly the same as lines 11 to 17 of algorithm 1, the distance metric script. During these lines, all the visited nodes are fetched.
- **Line 12 and 13:** all the available sessions are sorted according to the distance of the current session to these sessions. This *sort* operation relies on the *sessionDistance* tracking variable which has to be initialized before the execution of this code. The result of this *sort* operation is the same set of sessions but this time sorted ascending according to the value of the *sessionDistance* tracking variable. This result is then stored into the *sortedSessions* variable on the same line.
- **Line 14:** now that we have sorted the sessions according to their distance to the current session, we have to select the k sessions with the smallest distance. Retrieving these k sessions can be done by using the *subset* set creator which will take the first k elements of the *sortedSessions* set. This set is then stored into *KNearestSession* variable which we will use to calculate the page popularities in the remainder of the algorithm.
- **Line 15:** all the nodes which are not yet visited in the current session are retrieved and stored in the *unvisitedNodes* variable.
- **Line 16 to 27:** this *forEach* construct computes the page popularities for each of the unvisited nodes retrieved on line 15.
 - **Line 18 to 24:** these lines are the ASL version of the pseudo-code function *MEASURE_POPULARITY* which we saw earlier on. The popularity is calculated by testing for each session (line 19) if it has visited the node (line 21) and by increasing the popularity (line 22) when the test is passed. When this is done for all sessions, the popularity is divided by the number of sessions which is k (line 24).
 - **Line 25 and 26:** the last thing which happens is checking whether the page popularity is larger than the threshold (line 25). When this condition is met, a navigation aid session link is added to the unvisited node u and is therefore recommended (line 26). For esthetical reasons we have abbreviated *navigationAid* as *nA* on line 26, otherwise the line would be spread over 2 lines of code. Note that the *currentNode* function returns the loaded node that evoked the execution of the script.

Algorithm 3 The main adaptation specification calling both scripts.

```

1: when initialization do
2:   begin
3:     forEach node in Nodes:
4:       begin
5:         addTrackingVariable node.accesses;
6:         monitor load on node
7:         do begin
8:           currentSession := currentSession();
9:           node[currentSession].accesses :=
10:            node[currentSession].accesses + 1
11:         end
12:       end;
13:     call trackSessionDistance();
14:     forEach node in Nodes:
15:       monitor load on node
16:       do call KNNLinkRecommendationScript(10, 0.50)
17:   end

```

Algorithm 3 shows how both algorithms should be called in the main ASL script which is a script on the web site server that can be run by the administrator. Note that this is not a real script but an adaptation policy which triggers the execution of the other algorithms. Both scripts should be inserted before this adaptation specification. This adaptation specification first sets up the tracking variable for the node accesses, then calls the *trackSessionDistance* script and finally monitors every node and calls the *KNNLinkRecommendations* script when a node is loaded.

5.3.1.4 Analysis and extensions

Having described the k nearest neighbour link recommendation algorithm in detail, its nature will be discussed in this section. This includes an analysis of its complexity and a description of possible changes and extensions to the algorithm.

Our analysis starts with determining the time complexity of the algorithm. Time complexity is the amount of computer time or number of steps a program needs to run. This complexity is usually described using *big-O notation* which expresses a proportional upper bound on the running time. More detailed information about complexity can be found in [22].

The complexity of the *trackSessionDistance* script is $O(N) + O(S \cdot VN)$ where N is the number of nodes in the web site, S is the number of sessions and VN the number of visited nodes. $O(N)$ is the complexity of the *forEach* loop from lines 12 to 16 and $O(S \cdot VN)$ is the complexity of the *forEach* loop from lines 18 to 28. Taking in account the additive property of complexity which says that the complexity of two statements is the larger complexity makes that the total complexity is $\max(O(N), O(S \cdot VN))$. Because the number of sessions will be many times larger than the number of nodes in the session, the total complexity is $O(S \cdot VN)$. The worst case upper bound for this complexity is thus

$O(S \cdot N)$ as a user can at most visit N nodes. When the number of available sessions is high, this can be a very time consuming operation.

A way to reduce the complexity of the script is by computing the session distance to the last x^1 sessions. Because the *Sessions* set is sorted sequentially, this can be done as follows:

```
numberOfSessions := length(Sessions);
lastXSessions := subset(Sessions, numberOfSessions - x,
                        numberOfSessions);
```

These lines can be inserted at the begin of the script and x can be added as an input argument of the script. The *Sessions* set on line 18 should also be replaced with the *lastXSessions* set. Note that we have to include the same lines and use the same x value for the *KNNLinkRecommendations* script. Otherwise the script would try to retrieve the session distance to sessions for which this session distance has not been computed. The complexity is now reduced to $O(X \cdot N)$. It's obvious that the value of x should be still high enough, for example a tenth of the number of total sessions. A result of this new approach is that drifting user interests can be detected faster. The reason why is because it only considers the last x sessions and discards all older sessions. When visitors tend to visit certain nodes more during the last x sessions than in the past, these nodes will get a higher popularity when only these last x sessions are considered.

Another way to reduce the number of sessions is by only tracking the session distance to sessions with the same audience class. It can not only reduce the computational time but also the quality of the link recommendations as it considers only sessions with the same audience class as the session which gave rise to the execution of the script. This is an improvement of the quality because in WSDM, visitors with the same audience class are considered to be alike. The ASL code for this extension is as follows:

```
forEach session in Sessions:
  begin
    if audienceClass(session) = currentAudienceClass()
      then audienceClassSessions := audienceClassSessions + session
    end;
```

These lines should be added to both scripts and the *audienceClassSessions* set should be used instead of the *Sessions* set. Another possible change is to take into account the number of nodes a session visited for the computation of the session distance. This way, sessions which visited more nodes can be placed closer to the target session. This can be done by multiplying the session distance by the number of nodes accessed by the session to which the target session is compared to.

Let us now analyze the complexity of the *KNNLinkRecommendations* script. This complexity is $O(N) + O(S \cdot \log S) + O(K \cdot UN)$. $O(N)$ is the complexity of

¹Note that x should always be larger than k . Otherwise it is not possible to select k sessions.

the first *forEach* loop from lines 6 to 10. $O(S \cdot \log S)$ is the worst case complexity of the *sort* operation which can be achieved using a merge sort algorithm. $O(K \cdot UN)$ is the complexity of the *forEach* loop from line 16 to line 28, where K is the k value and UN denotes the number of unvisited nodes. A worst case upper bound for this complexity is $O(K \cdot N)$. All other statements and operations have a complexity of $O(1)$ and therefore don't need to be considered. We can also eliminate $O(N)$ which is certainly smaller than the other complexities. The largest complexity is dependent on the size of the web site. $O(S \cdot \log S)$ will be the largest complexity for a web site with thousands of visitors each day. When the value of k is large, for example half of the number of sessions, and there are many nodes in the web site, $O(K \cdot N)$ will be the largest complexity. We can now use this knowledge to improve the performance of the algorithm.

As we've already discussed for the *trackSessionDistance* script, the complexity of the algorithm can be improved by only considering the last x sessions or by only considering sessions with the same audience class as the given visitor. Instead of cutting down the size of the *Sessions* set, we can also shrink the size of the unvisited nodes set by only considering nodes which belong to the audience class of the visitor. The following code shows how this can be done:

```
unvisitedNodesInAC := {};
forEach u in unvisitedNodes:
  begin
    if audienceClass(u) = currentAudienceClass()
      then unvisitedNodesInAC := unvisitedNodesInAC + u
  end;
```

This code should be inserted after line 15 and the *unvisitedNodes* set should be replaced by the *unvisitedNodesInAC* set on line 16. Using this code, only nodes within the audience class of the user will be recommended. This narrows down his navigation space and can be seen as an exploitation strategy which exploits the fact that the user belongs to an audience class, i.e. no nodes belonging to an other audience class can be recommended.

So far, the main drive for the modifications was to reduce the computation of the algorithm. In addition to these computational improvements, we can also take the quality of the link recommendations as a starting point for improvements. A possible change which will improve the quality of link recommendations is to only start the recommendation process when a certain number of nodes have been visited. By doing this, we avoid generating the general recommendations which a visitor receives during the first few nodes he accesses. These recommendations at the begin of a session are mostly the most popular nodes on the web site and could lead the visitor to the information for which he is not looking for. To implement this, lines 13 to 15 in algorithm 3 should be replaced by the following code:

```
forEach node in Nodes:
  monitor load on node
  do begin
    accesses := 0;
    currentSession := currentSession();
```

```

forEach node in Nodes:
  begin
    if node[currentSession].accesses > 1
      then accesses := accesses + 1
    end;
    if accesses > 5
      then call KNNLinkRecommendationScript(10, 0.50)
    end

```

When a node is loaded, the above code first tests if the number of nodes accessed in the session which loaded the node is greater than 5, and only calls the *KNNLinkRecommendations* script when this test is passed.

Another extension which will be discussed is a change in the popularity computation. Until now, we have ignored the distance from the nodes to each other. With distance, we mean the minimal number of links which need to be traversed in order to reach a node. Do we want link recommendations to nodes which are located far from the current node or which are just a few clicks away? There is no real standard answer to this question, it all depends on what kind of recommendations we want to offer. Nodes which are located further will probably be less related to the current node than the nodes which lie closer to the current node. It's a difficult choice for the designer to answer this question for all visitors. Some visitors like to discover new subjects while others are more conservative and are only interested in closely related nodes. The best choice would therefore be to let the visitor decide individually about the kind of link recommendations he wants.

In practice, we need a new ASL native function which measures the distance between nodes. This distance can then be multiplied by the popularity to give a higher popularity to more distant nodes. By multiplying the popularity with the inverse of the distance, the opposite can be achieved (i.e. neighbour nodes will get a higher popularity than nodes which lie far away). Note that this distance has to be between 0 and 1, otherwise we will get popularities which are larger than 1. We will not include any ASL code as it is clear that the popularity only needs to be multiplied with the distance or the inverse of the distance between the two nodes.

A pure visual enhancement is to add a tracking variable to a session link holding the popularity of the target node of that session link. This tracking variable can then be used to adaptively sort the session links according to the popularity. This can be done by replacing lines 25 and 26 by the following code:

```

if popularity > threshold
then begin
  addSessionLink(navigationAid, currentSession, currentNode(), u)
  as newSessionLink
  addTrackingVariable newSessionLink.popularity
  initialized on popularity
end

```

To conclude the analysis of the algorithm, we discuss how the values of both input arguments influence the generated link recommendations. The value of the

first input argument, which is k , heavily depends on the web site traffic. When there are thousands of visitors each day, k should be set larger than when there are only hundred visitors each day. A good guideline to set the value of k is to fix it to some percentage of the average number of visitors each day. When doing so, we still have to decide to which percentage it should be set. A large percentage will lead to more general recommendation whereas a small percentage will lead to more specific recommendations. This is because, for a small percentage, only the visitors which are the most similar are considered. When lots of visitors are considered, there will be many visitors which are not at all similar to the current visitor. The value of k should therefore be chosen carefully depending on what kind of recommendations the designer wants to offer.

The second input argument, which is threshold value, determines how popular a node should be in order that it will be recommended. A low threshold value results in many link recommendations to less popular nodes while a high threshold value results in recommending less but more popular nodes. This is another choice the designer should make. It's advisable to set this threshold value not too low as this would lead to many link recommendations including links to less relevant nodes. Many link recommendations can also overwhelm the visitor with too many choices which is not what we want.

5.3.2 Link Recommendations using Bayes' Theorem

5.3.2.1 Introduction

The second link recommendation technique which will be proposed makes use of Bayes' Theorem. In chapter 3, we already discussed Bayes' Theorem and showed that it can be used to calculate the probability that a certain hypothesis holds.

Instead of computing the popularity of pages like in the previous algorithm, we will calculate the probability that a certain page is interesting for a visitor. For each page in the web site which the visitor has not yet viewed, we will try to determine its interestingness by taking into account the pages the visitor already visited. Bayes' Theorem will be used to determine this interestingness by calculating the posterior probability $P(up/vp)$ that a visitor will view an unseen page up given the pages vp he has already visited. Note that vp is a set of pages and is thus a conjunction of all the pages which the current visitor already visited. In the next section, we will see how $P(up/vp)$ is computed.

The following pseudo-code briefly sums up the algorithm which we will call the *Bayesian link recommendation algorithm*:

```

FETCH_INTERESTING_LINKS(WSP, t, S)
  VP = get visited pages of current session using S
  UP = {WSP / VP}
  For each  $up$  in UP do:
    Calculate  $p = P(up | VP)$  using Bayes' Theorem
    If  $p > t$ 
      Recommend  $up$ 

```

The three input arguments are the same as the first three arguments of the previous algorithm: WSP contains the set of all pages in the web site, t is the threshold that should be reached to recommend a page to the visitor and S contains all session information. The format of this session information is the same as for the previous algorithm. The first two lines are the same as for the k nearest neighbour link recommendation algorithm: they compute the set of visited pages. This retrieved set of visited pages is stored in VP which is then used for the computing the interestingness of all pages the visitor has not yet visited. This set of unvisited pages is stored in UP . The next two lines calculate the posterior probability p for each unseen page up in the set of unseen pages using Bayes' Theorem. When the probability of an unseen page is larger then the threshold, the page is recommended to the visitor. Note that the session data S is also used for the calculation of the posterior probability by Bayes' Theorem as we will see in a short while.

The above pseudo-code is an over-simplified version of the algorithm as it doesn't specify how the probabilities are calculated and merely shows the base structure of the algorithm. This pseudo-code shows that this base structure is very simple and that all of the computation lies within the calculation of the probabilities using Bayes' Theorem. Hence, all our concentration will be focused on the calculation of the page view probabilities using Bayes' Theorem in the next section.

5.3.2.2 Using Bayes' Theorem to calculate page view probabilities

In this section we will show how Bayes' Theorem can be used to calculate page view probabilities which will be used as an indication of the interestingness of pages. The technique which we will describe is a memory-based collaborative recommendation technique in the sense that it uses the session data of previous visitors stored in memory to calculate the interestingness of pages.

In the introduction, we already mentioned that we will calculate the posterior probability $P(up/vp)$ using Bayes' Theorem. This posterior probability can be written more precisely as follows:

$$P(up/vp) = P(up|vp_1 \wedge vp_2 \wedge \dots \wedge vp_n) \quad (5.1)$$

This expresses the probability that the unvisited page up will be visited given the set of visited pages $(vp_1 \wedge vp_2 \wedge \dots \wedge vp_n)$. Notice that we use vp as a shorthand notation for this set of visited pages. Using the formula of Bayes' Theorem, the posterior probability can be calculated in the following manner:

$$\begin{aligned} P(up/vp) &= P(up|vp_1 \wedge vp_2 \wedge \dots \wedge vp_n) \\ &= \frac{P(vp_1 \wedge vp_2 \wedge \dots \wedge vp_n|up) \cdot P(up)}{P(vp_1 \wedge vp_2 \wedge \dots \wedge vp_n)} \end{aligned} \quad (5.2)$$

To compute of the posterior probability $P(up/vp)$, we have to estimate the following three probabilities:

1. $P(vp_1 \wedge vp_2 \wedge \dots \wedge vp_n|up)$ which is the probability that the pages $vp_1 \wedge vp_2 \wedge \dots \wedge vp_n$ will be visited given the fact that page up is visited.

2. $P(up)$ which is the prior probability that page up will be visited.
3. $P(vp_1 \wedge vp_2 \wedge \dots \wedge vp_n)$ which is the prior probability that the pages $vp_1 \wedge vp_2 \wedge \dots \wedge vp_n$ are visited together.

Once we know these three probabilities, we know the posterior probability. The last two probabilities are fairly easy to compute using the available session data and therefore we will start with explaining how they are computed before we take on the first probability.

The prior probability $P(up)$ expresses the probability that this page will be visited by a user who visits the web site. For the home page of a web site, this probability will be very high as almost all users enter a web site via its home page. Other pages which are buried deep in the web site will be visited rarely and have an accordingly low probability. How can this probability be computed? The answer is simple: count by how many visitor sessions the page was accessed and divided this number by the total number of visitor sessions. This gives the percentage of sessions in which the given page was accessed. If a page is accessed in all previous sessions, we may conclude that the probability that the page will be accessed in the new session is 100%. More formally, we get:

$$P(up) = \frac{\text{count}(\text{sessions which visited } up)}{\text{count}(\text{sessions})} \quad (5.3)$$

The next prior probability is a little bit harder to compute as several pages are involved in the computation. The computation is however similar to the previous one. To estimate this probability, we count the number of sessions in which these pages are accessed together and divide this number again by the total number of visitor sessions. This will result in lower probabilities and serves as the normalization factor for the posterior probability. This leads to the following “formula”:

$$\begin{aligned} P(vp) &= P(vp_1 \wedge vp_2 \wedge \dots \wedge vp_n) \\ &= \frac{\text{count}(\text{sessions which visited all pages in } vp)}{\text{count}(\text{sessions})} \end{aligned} \quad (5.4)$$

The first probability in the above list is the hardest one to compute. To compute the probability $P(vp_1 \wedge vp_2 \wedge \dots \wedge vp_n|up)$, which is also called the likelihood, we will assume that page visits are independent of each other. This is the same naive assumption which we used for the Naive Bayesian Classifier in chapter 3. It’s clear that this is a false assumption as page visits are definitely related to each other. Each page which a user visits is dependent on the current page as he can only visit the pages which are reachable from the current page (i.e. for which a link exists from the current page). We already mentioned in chapter 3 that although this naive assumption often doesn’t hold, it gives good results. Using this naive assumption, calculating the probability $P(vp_1 \wedge vp_2 \wedge \dots \wedge vp_n|up)$ can now be written as:

$$P(vp_1 \wedge vp_2 \wedge \dots \wedge vp_n|up) = P(vp_1|up) \cdot P(vp_2|up) \cdot \dots \cdot P(vp_n|up) \quad (5.5)$$

The calculation is therefore reduced to finding a good estimator of $P(vp_x|up)$ or the probability that page vp_x will be visited given that page up is visited. We can use the available session data to estimate this value as follows:

$$P(vp_x|up) = \frac{\text{count}(\text{sessions which visited both } vp_x \text{ and } up)}{\text{count}(\text{sessions which visited } up)} \quad (5.6)$$

We have now gathered all probabilities needed to compute $P(up/vp)$. They are all based on evidence, the available session data, which is then used to make a prediction about an unvisited page in the web site.

Example of a posterior probability computation

To give a deeper insight in how the posterior probability is computed in practice we will now give a simple example. We will again use the web site of “The Recommenders” and its session data to illustrate the computation of the posterior probability. The posterior probability will be computed in detail for the “Tour” page of the web site.

For the computation of the posterior probability of the “Tour” page, we need to take in account the pages which have already been visited by the user in the current session. For the visitor of session 9 this are the “Home Page”, “News”, “Biography” and the “Photography” pages. We therefore need to compute the probability that the “Tour” page will be visited given his browsing history (i.e. the 4 notes already visited in this session). By substituting these pages into equation 5.2 we get:

$$P(T|HP \wedge N \wedge B \wedge P) = \frac{P(HP \wedge N \wedge B \wedge P|T) \cdot P(T)}{P(HP \wedge N \wedge B \wedge P)}$$

To compute the posterior probability, we will first calculate the three probabilities in the formula separately. We will start with the easiest one: $P(T)$ or the probability that the “Tour” page will be visited. Recall equation 5.3 which calculates this probability:

$$P(T) = \frac{\text{count}(\text{sessions which visited } T)}{\text{count}(\text{sessions})}$$

When we substitute the available information we find in table 5.2, we can compute the value of $P(T)$ easily:

$$\begin{aligned} P(T) &= \frac{\text{count}(\text{sessions which visited } T)}{\text{count}(\text{sessions})} \\ &= \frac{4}{8} \\ &= 0.5 \end{aligned}$$

The next probability which we will compute is $P(HP \wedge N \wedge B \wedge P)$ or the probability that these 4 pages will be visited together in one session. The formula

to compute this probability is as simple as the previous one. Substituting the session information into equation 5.4 gives:

$$\begin{aligned} P(HP \wedge N \wedge B \wedge P) &= P(vp) \\ &= \frac{3}{8} \\ &= 0.375 \end{aligned}$$

The last missing piece is the probability that these 4 pages will be visited given the fact that the “Tour” page is already visited. Using the formula we have seen to compute the probability $P(HP \wedge N \wedge B \wedge P|T)$, which is equation 5.5, we get:

$$P(HP \wedge N \wedge B \wedge P|T) = P(HP|T) \cdot P(N|T) \cdot P(B|T) \cdot P(P|T)$$

We can now use the formula $P(vp_x|up)$ to compute each of the 5 remaining probabilities. We will only show how it is computed for the first one of the list because it is always the same process. Computing $P(HP|T)$ using equation 5.6 gives:

$$\begin{aligned} P(HP|T) &= \frac{\text{count}(\text{sessions which visited both HP and T})}{\text{count}(\text{sessions which visited T})} \\ &= \frac{4}{4} \\ &= 1 \end{aligned}$$

Retrieving this info is simple: table 5.3 shows that there are 4 sessions which have visited both the “Home Page” page and the “Tour” page. There are also 4 sessions which visited the “Tour” page and therefore every session that visited the “Tour” page also visited the “Home Page” page. This make sense as we stated earlier on that each visitor enters the web site via the “Home Page” page. Doing the same for all other 4 probabilities we get the following values:

$$\begin{aligned} P(N|T) &= \frac{4}{4} \\ &= 1 \\ P(B|T) &= \frac{3}{4} \\ &= 0.75 \\ P(P|T) &= \frac{3}{4} \\ &= 0.75 \end{aligned}$$

Computing $P(HP \wedge N \wedge B \wedge P|T)$ is now easy:

$$P(HP \wedge N \wedge B \wedge P|T) = 1 \cdot 1 \cdot 0.75 \cdot 0.75 = 0.5625$$

We now have the values of all three probabilities needed to compute the posterior probability, all we need to do is to substitute these values into Bayes’ Theorem. That results in:

| | Web site pages | | | | | | | | | | | |
|----------|----------------|---|---|----|---|----|----|----|-----|----|----|----|
| Sessions | HP | N | B | D | P | T | BJ | BM | BP | PJ | PM | PP |
| 1 | ✓ | ✓ | ✓ | ✓ | | ✓ | | ✓ | | | | |
| 2 | ✓ | | | | ✓ | | | | | | | ✓ |
| 3 | ✓ | ✓ | | ✓ | ✓ | | | | | | | ✓ |
| 4 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | ✓ | ✓ | | ✓ | ✓ |
| 5 | ✓ | | | ✓ | | | | | | | | |
| 6 | ✓ | ✓ | ✓ | | ✓ | ✓ | ✓ | | ✓ | ✓ | | ✓ |
| 7 | ✓ | ✓ | | | ✓ | ✓ | | | | ✓ | ✓ | ✓ |
| 8 | ✓ | ✓ | ✓ | | ✓ | | | ✓ | ✓ | | ✓ | ✓ |
| 9 | ✓ | ✓ | ✓ | 25 | ✓ | 75 | 33 | 67 | 100 | 33 | 67 | 83 |

Table 5.7: The session data extended with posterior probabilities.

$$\begin{aligned}
P(T|HP \wedge N \wedge B \wedge P) &= \frac{P(HP \wedge N \wedge B \wedge P|T) \cdot P(T)}{P(HP \wedge N \wedge B \wedge P)} \\
&= \frac{0.5625 \cdot 0.5}{0.375} \\
&= 0.75
\end{aligned}$$

The probability that the “Tour” page will be visited given the 4 visited pages is thus 75%. When we do the same for all other unvisited pages, we can replace the question marks in table 5.3 with the posterior probabilities that these pages will be visited. The session data for the web site of the “The Recommenders” extended with the posterior probabilities expressed in percentages for each unvisited page in the ninth session can be seen in table 5.7.

This table shows that the “Bio Paul” page gets a probability of 100% percent. This is because the visitor sessions which also visited the same pages as the ninth visitor session, all visited the “Bio Paul” page. We will examine the nature of this algorithm more profoundly in section 5.3.2.4.

5.3.2.3 The algorithm expressed in ASL

In this section we will express the Bayesian link recommendation algorithm in ASL. Because, in ASL, the algorithm operates on nodes and not on pages we will talk about recommending nodes instead of pages just as for the previous algorithm. We will adjust the formula of Bayes’ Theorem which we have seen before because we consider nodes instead of pages. Therefore we will use un and vn to denote unvisited and visited nodes respectively. The equation 5.2 is now written as:

$$P(un|vn) = \frac{P(vn_1 \wedge vn_2 \wedge \dots \wedge vn_n|un) \cdot P(un)}{P(vn_1 \wedge vn_2 \wedge \dots \wedge vn_n)} \quad (5.7)$$

Just as for the previous algorithm, we need the *accesses* tracking variable along with the semantic ASL extensions we proposed in section 5.2.2 to express the

Algorithm 4 The Bayesian link recommendation algorithm.

```

1: script BayesianLinkRecommendations(threshold):
2: begin
3:   currentSession := currentSession();
4:   clearSessionLinks(navigationAid, currentSession);
5:   visitedNodes := {};
6:   forEach n in Nodes:
7:     begin
8:       if n[currentSession].accesses > 0
9:       then visitedNodes := visitedNodes + {n};
10:    end;
11:   numberOfVisitedNodes := length(visitedNodes);
12:   priorProbability := 0;
13:   forEach s in Sessions:
14:     begin
15:       accessedNodes := 0;
16:       forEach visitedNode in visitedNodes:
17:         begin
18:           if visitedNode[s].accesses > 0
19:           then accessedNodes := accessedNodes + 1
20:         end;
21:       if accessedNodes = numberOfVisitedNodes
22:       then priorProbability := priorProbability + 1
23:     end;
24:   priorProbability := priorProbability / length(Sessions);
25:   unvisitedNodes := Nodes DIFFERENCE visitedNodes;
26:   forEach u in unvisitedNodes:
27:     begin
28:       P := 0;
29:       PHu := 1;
30:       countu := 0;
31:       forEach s in Sessions:
32:         begin
33:           if u[s].accesses > 0
34:           then countu := countu + 1
35:         end;
36:       Pu := countu / length(Sessions);
37:       forEach v in visitedNodes:
38:         begin
39:           countvu := 0;
40:           forEach s in Sessions:
41:             begin
42:               if (u[s].accesses > 0) AND (v[s].accesses > 0)
43:               then countvu := countvu + 1
44:             end;
45:           PHu := PHu * (countvu / countu);
46:         end;
47:       P := (PHu * Pu) / priorProbability;
48:       if P > threshold
49:       then addSessionLink(nA, currentSession, currentNode(), u)
50:     end
51: end

```

algorithm in ASL.

Algorithm 4 shows all the ASL code of the bayesian link recommendation algorithm. This algorithm will be added as a runtime script to each node as follows:

```

forEach node in Nodes:
    monitor load on node
    do call BayesianLinkRecommendations(0.50)

```

This means that every time when a node is loaded by a visitor, this script is executed. As a result, the *currentSession* constant will be bound to the session of the visitor who loaded the node and is therefore accessible from within the script. This code should be put in the main ASL script. The ASL code of the Bayesian link recommendation algorithm can be explained as follows:

- **Line 1:** the script is named *BayesianLinkRecommendations* and takes the threshold value as an input argument.
- **Line 3 to 11:** these are exactly the same lines as in the k nearest neighbour link recommendation algorithm. They mainly retrieve the nodes visited in the session which loaded the node.
- **Line 12 to 24:** during these thirteen lines, the prior probability $P(vn_1 \wedge vn_2 \wedge \dots \wedge vn_n)$ that all nodes in the browsing history are visited together is computed and stored into the *priorProbability* variable. Recall from section 5.3.2.2 that the prior probability can be calculated by counting all sessions in which all these nodes are accessed together and by dividing this number by the total number of sessions. The total number of sessions is easy to retrieve in ASL by using the *length* operator on the *Sessions* set. The other value, the number of sessions in which all these nodes are accessed together, is a little bit harder to compute.
 - **Line 12:** the prior probability is initially set to 0.
 - **Line 13 to 23:** this *forEach* construct computes the number of sessions in which all the nodes are accessed together. This is done by iterating over the set of sessions and verifying for each session if all nodes in the browsing history are also visited in that session.
 - ◊ **Line 16 to 20:** these lines count how many nodes from the browsing history are visited by the given session.
 - ◊ **Line 21 and 22:** when the value of the *accessedNodes* variable computed during line 16 to 20 is equal to the number of nodes in this browsing history, the prior probability is increased.
 - **Line 24:** the result of the *forEach* loop from line 13 to 23 which is stored in the *priorProbability* variable is then divided by the total number of sessions.
- **Line 25:** Before we can compute the posterior probabilities for the nodes which are not yet visited, we have to retrieve this set of unvisited nodes. As can be seen on this line, this is done using the *DIFFERENCE* operator which takes the difference between all nodes in the web site and the visited nodes.

- **Line 26 to 50:** we now arrive at the main part of the algorithm which is the computation of the posterior probabilities for each unvisited node. The *forEach* construct computes the posterior probability for each unvisited node (lines 27-46) and recommends the node if its probability is larger than the threshold (lines 47-48). Let's first analyze how the posterior probability is computed.
 - **Line 28 to 30:** during these three lines, three variables are initialized. The variable P denotes the posterior probability $P(un|vn)$ of the unvisited node and is set to 0. The next variable PHu is the one which will hold the probability $P(vn_1 \wedge vn_2 \wedge \dots \wedge vn_n|un)$. The *countu* variable will be used to hold the number of sessions which accessed the unvisited node.
 - **Line 31 to 36:** these lines compute the next probability which is the prior probability $P(un)$ that the node u will be visited.
 - ◊ **Line 31 to 35:** this *forEach* loop computes the value of the *countu* variable by testing for each session if the node u was visited (line 33) and incrementing the *countu* value when this test is passed (line 34). Using this variable, we can compute the prior probability $P(un)$.
 - ◊ **Line 36:** the prior probability is computed by dividing the *countu* variable by the total number of sessions and the result is then stored in a new variable Pu .
 - **Line 37 to 46:** so far, we have computed two probabilities: $P(vn_1 \wedge vn_2 \wedge \dots \wedge vn_n)$ and $P(un)$. When we take a look at equation 5.7, we see that we still need to compute $P(vn_1 \wedge vn_2 \wedge \dots \wedge vn_n|un)$. This is what happens in this *forEach* construct. Recall that this probability will be stored in the PHu variable and that we will compute its value using equation 5.5. This equation computes, for each node vn_x , the probability that it will be visited given the fact that node un is visited. Multiplying these probabilities then gives the conditional probability $P(vn_1 \wedge vn_2 \wedge \dots \wedge vn_n|un)$. This is the reason why this *forEach* construct iterates over every visited node.
 - ◊ **Line 39 to 44:** for each visited node vn_x , the number of sessions that visited un and vn_x together is calculated and stored into the variable *countvu*.
 - ◊ **Lines 42 and 43:** the *countvu* variable is increased whenever a session accessed both nodes, un and vn_x , together.
 - ◊ **Line 45:** the value of the *countvu* variable is needed as the numerator of equation 5.6. This equation computes the probability $P(vn_x|un)$ by counting the number of sessions which accessed both nodes (the *countvu* variable computed on lines 39-44) and by dividing this value by the number of sessions which accessed un (the *countu* variable which was computed on lines 31-35). This is done on line 45. The result of the quotient computed on the previous line is multiplied by the PHu variable and stored into the PHu variable which represents the probability

Algorithm 5 The main adaptation specification which loads the Bayesian link recommendation algorithm.

```

1: when initialization do
2:   begin
3:     forEach node in Nodes:
4:       begin
5:         addTrackingVariable node.accesses;
6:         monitor load on node
7:         do node[currentSession()].accesses :=
8:           node[currentSession()].accesses + 1
9:       end;
10:    forEach node in Nodes:
11:      monitor load on node
12:      do call BayesianLinkRecommendations(0.50)
13:    end

```

$P(vn_1 \wedge vn_2 \wedge \dots \wedge vn_n | un)$. This probability is thus incrementally computed: for each visited node vn_x , $P(vn_x | un)$ is computed and multiplied by the value of the PHu variable which is initially 1. When the *forEach* loop ends, we have the correct value of PHu .

- **Line 47:** we now have all three probabilities required to compute the posterior probability. This posterior probability is then finally assigned to P on this line using equation 5.7. When we replace the probabilities in this equation with the variables used in algorithm 4 we get:

$$P = \frac{PHu \cdot Pu}{priorProbability}$$

This concludes the computation of our target posterior probability.

- **Line 48 and 49:** the only thing left to do is to determine whether the node un is interesting enough to be recommended. To do this, its probability has to be compared with the threshold value and to recommend the node when it is bigger than the threshold. This is what happens on these lines which conclude the algorithm. Keep in mind that this is done for every node which is not yet visited. For the same esthetical reasons as for algorithm 2 we have abbreviated the type of *navigationAid* as nA on line 49.

The main script which enables the algorithm can be seen on algorithm 5. This code sets up the *accesses* tracking variable and the monitors which update the tracking variable and call the Bayesian link recommendation script whenever a node is loaded.

5.3.2.4 Analysis and extensions

Just as we did for the k nearest neighbour link recommendation algorithm, we will now analyze the Bayesian link recommendation algorithm. We will also discuss how most of the extensions which we proposed for the k nearest neighbour

link recommendation algorithm can be applied on the Bayesian link recommendation algorithm.

We start this section with an analysis of the time complexity of the Bayesian link recommendation algorithm. We will first determine the complexity of the three main *forEach* loops, all other statements used such as assignments and functions have a complexity of $O(1)$. The complexity of the first *forEach* loop from lines 6 to 10 is $O(N)$ where N is the number of nodes in the web site. The second *forEach* loop ranging from line 13 to line 23 has a complexity of $O(S \cdot VN)$ where S is the number of sessions and VN the number of visited nodes. The last *forEach* loop (lines 26-50) is largest one and includes two other *forEach* loops. The first inner *forEach* loop (lines 31-35) has a complexity of $O(S)$, the second one has a complexity of $O(S \cdot VN)$. Taking into account the additive rule for sequential statements, the complexity of the body of the *forEach* loop on lines 26 to 50 is $\max(O(S), O(S \cdot VN))$ which is obviously $O(S \cdot VN)$. This results in a complexity of $O(UN \cdot S \cdot VN)$ where UN is the number of unvisited nodes. This is also the largest complexity of the three main *forEach* loops and is thus the total complexity, at least an upper bound on the running time of the algorithm. We can even bring down the number of involved variables as follows:

$$UN \cdot VN \leq \frac{N}{2} \cdot \frac{N}{2}$$

The value of both UN and VN can be at most $\frac{N}{2}$. This is the case when the number visited and unvisited pages is equal. Whenever these values are unequal, the product of both values will be less than $\frac{N}{2} \cdot \frac{N}{2}$. The complexity is therefore reduced to $O(S \cdot N^2)$ because the division by two is irrelevant for the upper bound. This worst case upper bound means that the running of this algorithm is more dependent on the number of nodes than the running time of the k nearest neighbour link recommendation algorithm which was more dependent on the number of sessions.

We will now discuss some extensions and changes which can reduce the running time of the algorithm and increase the quality of the offered link recommendations. All the extensions which we proposed for the k nearest neighbour link recommendation algorithm are also applicable on this algorithm. We will not list the same code and advantages of these extensions again in this section because that would be redundant. We will only discuss how the algorithm can be extended with these changes.

In section 5.3.1.4, the following extensions were proposed:

1. Cutting down the number of sessions.
 - (a) By only considering the last x sessions.
 - (b) By only considering the sessions with the same audience class.
2. Cutting down the number of recommendable nodes by only considering nodes which lie in the audience class of the visitor.

3. Suspending the recommendation process until a certain number of nodes is visited.
4. Adaptive link sorting.
5. Taking into account the distance to nodes for the computation of the page interestingness value.

The two ways of cutting down the number of sessions can easily be used in the Bayesian link recommendation algorithm by replacing the *Sessions* set with the set containing either the last x sessions (a) or the sessions with the same audience class (b). This will improve the performance of the algorithm because the number of sessions is an important factor in the complexity of the algorithm. The quality of the link recommendations will also improve like we described in section 5.3.1.4. This includes better detection of shifting user interests (for a) and recommendations of more alike visitors (for b).

Cutting down the number of recommendable nodes can also lead to a better performance as the complexity is partly determined by the number of nodes in the web site. This can be realized by replacing the *unvisitedNodes* set on line 26 by the *unvisitedNodesInAC* set.

The third change which suspends the recommendation process will be as useful for the Bayesian link recommendation algorithm as for the k nearest neighbour link recommendation algorithm. Replacing lines 9 to 11 of 5 by the appropriate code in section 5.3.1.4 is the only thing which needs to be done.

Adaptive link sorting can be done in the same way as for the previous algorithm. The only thing which is needed is a tracking variable holding the prior probability.

The last change which takes in account the distance to nodes for the computation of the interestingness value will have the same effects as on the k nearest neighbour link recommendation algorithm. This can be easily done by multiplying the prior probability that a node will be visited with the inverse distance to this node.

Unlike the k nearest neighbour link recommendation algorithm, the Bayesian link recommendation algorithm only has one input argument which is the threshold value. Determining a suited threshold value is quite the same task as for the previous algorithm. The designer has to make a choice about whether he wants to offer many link recommendations, including less relevant recommendations, or few more interesting recommendations.

Chapter 6

Implementation

6.1 Introduction

In the previous chapter of this dissertation, we introduced two link recommendation algorithms: the bayesian link recommendation algorithm and the k nearest neighbour link recommendation algorithm. To be able to actually test and evaluate these algorithms, we need to design and construct a web site using WSDM. When this web site is available for use, an ASL interpreter needs to be written in order to be able to execute ASL scripts such as the ones containing the algorithms. This chapter will discuss the design and the implementation of a WSDM web site along with the implementation of an ASL interpreter.

The implementation is based on the web site which was constructed during my apprenticeship together with my fellow student Tom Vleminckx. The main goal of the apprenticeship was to transform the WISE web site into a web site which supports adaptive behaviour. This included annotation of the web site, creation of an at runtime changeable graph structure mirroring the Navigation Model and the implementation of an ASL interpreter. These assignments all deal with adaptivity and are mainly focused on the last two phases of WSDM: the implementation design phase and the implementation phase. By annotating the web site, the information can be much easier adapted and retrieved. The creation of an at runtime changeable graph structure permits applying adaptive techniques which change the structure of the web site. Using the ASL interpreter, the designer should be able to specify at design time certain kinds of adaptive behaviour that changes the graph structure at runtime.

Because the WISE web site was already designed using WSDM, we didn't need to design the web site ourselves. We only had to use all the available information which was already identified during these first three phases of the design process, with the Navigation Model as the most important result of these three phases.

The WISE web site which we will transform into an at runtime adaptable web site is a rather small web site containing only 17 nodes and links, 24 chunks and connections and only 2 real audience classes. The existing WISE web site

was designed using WSDM, but there was no separation between nodes and chunks. There were only 17 static HTML pages which had to be reconstructed into a dynamic graph containing nodes, chunks, links and connections.

Figure 2.2 in section 2.3.1 of chapter 2 shows the simplified Navigation Model of the WISE web site. There are 3 audience classes: the *Visitor*, *Student* and *Researcher* audience class. The audience track of the *Student* and *Researcher* audience class can not be seen on the figure 2.2 as the whole structure of the web site would overload the figure.

The remainder of this chapter is structured as follows. In section 6.2, we will briefly go through the design phases that have lead to the current structure of the WISE web site in order to be able to identify important design elements like chunks and nodes. The next section, section 6.3, discusses the implementation of a WSDM web site and section 6.4 discusses the implementation of an ASL interpreter. A global view on the implemented web site is then served in the last section, section 6.5.

6.2 Designing a Web Site using WSDM

6.2.1 Introduction

The web site graph structure which we want to create depends totally on the Navigation Model identified during the Conceptual Design phase. This is the reason why we will discuss the design of the WISE web site briefly in this section. Doing this will indicate which design elements are needed to create the graph structure.

We use the WISE web site as an example throughout this chapter because it was the web site for which we implemented the graph structure. It's obvious that the implementation discussed here is applicable for all web sites designed using WSDM. The purpose of this section is to show how the WSDM design phases, which we saw this already in chapter 2, are performed in practice, and to identify the design elements needed to implement a graph structure for any given web site designed using WSDM. Discussing the WSDM phases for the WISE web site briefly, will help us in understanding WSDM better. We will discuss each phase by using some examples of the WISE web site.

6.2.2 Mission Statement Specification

For the WISE web site, the Mission Statement Specification phase identifies the activities of the WISE research group as the subject of the site, and giving information about the WISE research group, their courses, research topics, research projects and publications as the purpose of the web site. Furthermore, students and researchers are identified as the target audience of the web site. This phase does not reveal any information which we will need to create the graph but it is nevertheless an essential phase.

6.2.3 Audience Modelling

Recall that the Audience Modelling phase has two sub-phases: the Audience Classification and the Audience Class Characterization phase. The first sub-phase uses the target audience identified in the previous phase and classifies this target audience into audience classes. This results in a *Student*, *Researcher* and *Visitor* audience class. This last one is the standard audience class which specifies the requirements that all visitors of the web site share. For each audience class which is a subclass of the *Visitor* audience class, additional informational, functional requirements and optionally navigation requirements are specified. For the *Student* audience class of the WISE web site, for example, this leads to the following requirements:

- General information about WISE
- Information about the WISE members
- Information about the courses offered by the WISE research group
- Information about apprenticeship and thesis proposals
- Information about the current thesis students and the past theses performed at WISE

Doing this for all audience classes of a web site outputs an audience class hierarchy containing all the requirements for every possible visitor of the web site. In the second sub-phase of the Audience Modelling phase we identify characteristics for each audience class. For the *Researcher* audience class of the WISE web site, for example, we can identify the following characteristics:

- English speaking
- Have much experience with web sites
- Older than 20
- Experts in their research field

The audience class hierarchy together with all the requirements and characteristics is the input for the next phase which will produce all the design elements needed to create the graph structure.

6.2.4 Conceptual Design

The Conceptual Design phase is the most important phase for the implementation of the web site because the Navigation Model is constructed in this phase. During the first sub-phase, the Task & Information Modelling phase, a task model needs to be defined for each requirement. This task is then elaborated further into more detail and decomposed into elementary tasks. For the WISE web site, there are no functional requirements such as buying items and filling in forms. These are functional requirements which require large task models in contrast to the task model of the following functional requirement of researchers:

Retrieving a list of publications and selecting a publication.

This functional requirement results in the simple task model shown on figure 6.1. The notation of this task model is the *Concurrent Task Tree* notation (CTT) [44]. This technique is used in WSDM for task modeling because it allows, besides the task decomposition, to specify temporal relationships between different subtasks. The cloud icon on the figure indicates that the task is an abstract task which is a task consisting of complex activities, the computer icon indicates that the task is an application task executed by the application and the last icon denotes an interaction task which has to be performed by the user by interaction with the system. The abstract task is in this case the “Retrieve A Publication” task which is decomposed in two sub-tasks: the “Show Publications” task where the system should show the publications to the user and the “Select Publication” task which should be performed by the user. The $>>$ operator means that the “Show Publications” task enables the “Select Publication” task. More extensive examples of how this task modelling is done for more complex tasks can be found in [29].

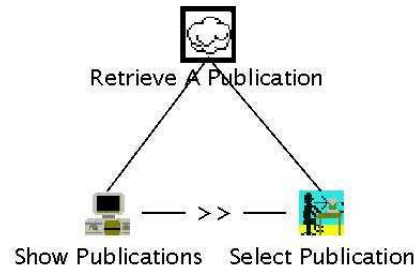


Figure 6.1: The task model for retrieving a publication.

In the next sub-phase of the Task & Information Modelling phase, the Information and Functional Modelling phase, objects chunks are created for each elementary task. For the task “Show Publications” the information that needs to be displayed about the publications should be modelled. For the modelling of this task into an object chunks, ORM is used. *ORM* stands for *Object Role Modelling* and is a widely used method for designing objects. More information on ORM can be found in [32]. The object chunk of the “Show Publications” task can be seen on figure 6.2.

This figure shows that a publication has a title, a year, optionally a location, one or more authors, and also optionally comments, an URL and a description. Such an object chunk has to be constructed for all elementary tasks. We will soon see that we need all this information of the object chunks to annotate the chunks in the WISE web site. When all object chunks are identified, they are linked together into a Business Information Model. The purpose of the Business Information Model is to describe and control possible redundancy by defining information chunks as views on this model. The Business Information Model of the WISE site will not be displayed here as it is quite large and not relevant. The model itself will have its use for the annotation process as we will see in section 6.3.

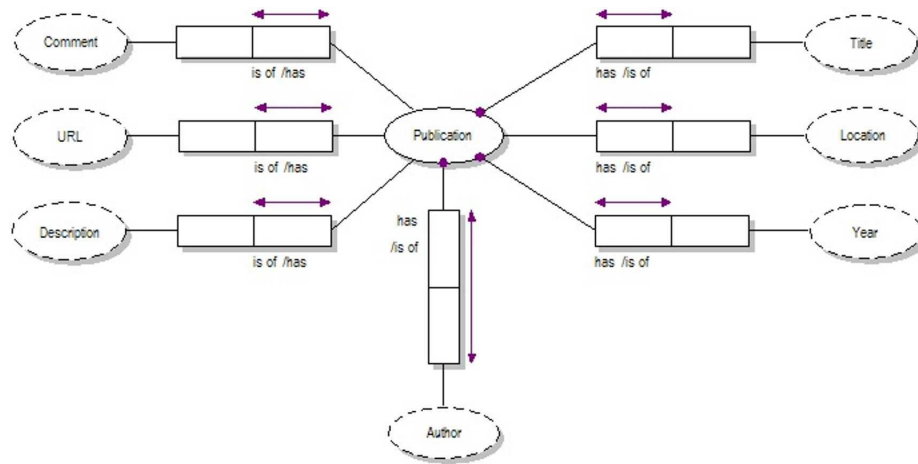


Figure 6.2: The publication object chunk.

The second sub-phase of the Conceptual Design phase is the Navigation Design phase in which the Navigation Model of the web site is designed. The first thing which needs to be done is creating a navigation track for each audience class. The task models and objects chunks are the input for this phase. For each task model, a Task Navigation Model is constructed using *components* and *project logic links*. For each elementary interaction task, a component is defined in the hierarchical decomposition of the task. Between these components, process links are used to express the workflow or process logic expressed in the task model by means of the temporal CTT relations (i.e. CTT relations are translated to links). Because none of the task models of the WISE web site are complex enough to transform to a Task Navigation Model with two components, we will not illustrate this process for the WISE web site. For an in-depth explanation of this process including examples, I refer to [29].

When all Task Navigation Models are created, they are linked together by means of structural links to form navigation tracks. The components of each Task Navigation Model which represent tasks are connected to the object chunks for the tasks which they represent. A component is in fact, conceptually the same as a node, which brings us to the Navigation Model. When we put all audience tracks together we get the Navigation Model containing the nodes (or components) and the object chunks. We have already seen a part of the Navigation Model of the WISE web site in chapter 3 so it would be redundant to include it again in this section.

6.2.5 Implementation Design

In the implementation design phase nodes are mapped to pages and the positions of chunks on pages are laid out. This is of great importance for the graph structure as we need to define this mapping and positioning in the graph structure. In the implementation design phase, layout templates are also created

and mapped to pages. These are all very important decisions which have to be mirrored in the graph structure which will be designed. For the WISE web site, all nodes map to one page.

6.2.6 Implementation

In this last phase, the web site is constructed using all the output data acquired in the previous phases. For the visitors of the web site, the implementation has no importance whatsoever. Of course some technologies are better suited for certain web sites than others but the key issue is that every detail of the web site is already defined in the previous phases.

So why not choose for the easiest implementation and go for manually created static HTML pages? Because HTML encapsulates the information on the level of pages. A HTML page does not reveal the nodes and chunks it contains, it is an indivisible unit. For a static web site which does not change often, the simplest straight on implementation might be the best. But even when the web site is small, it will be easier to maintain by the web master when he can work on the finest level, i.e. on the level of nodes and chunks instead of pages. When an information chunk which appears on multiple pages needs to be edited, the web master needs to edit each HTML page containing the chunk page separately. When pages are created at runtime and chunks are editable separately out of their page context, only one chunk needs to be edited.

The main reason for choosing for the graph structure is to be able to support adaptive behaviour. Adaptive behaviour requires that all design elements (e.g. nodes and chunks) need to be separately manipulatable. This will give the designer the chance to, for example, create links between nodes and to disconnect chunks from nodes at runtime. In the next section, we will introduce the first graph structure based implementation of a WSDM web site. All design elements will be reflected in the underlying graph structure of the web site. The result will be the same HTML web site as the original one, but this time generated at runtime using the information about nodes and chunks in the graph structure.

6.3 Implementing the WSDM Web Site

6.3.1 Introduction

This section will discuss the implementation of a WSDM web site as a dynamic graph structure mirroring the Navigation Model. The graph structure will be dynamic because its structure will be alterable at runtime using the ASL interpreter we will discuss in section 6.4. Prior to writing such an ASL interpreter, we first need to define the graph structure upon which it will act. The implementation process will be described in a bottom-up manner from the smallest design elements to the largest one comprising all others. The smallest design elements are the chunks containing information and functionality and the largest design element is the Navigation Model.

6.3.2 Annotation

During the Conceptual Design phase, more precisely the Information and Functional Modelling phase, object chunks have been created. The publication chunk ORM diagram on figure 6.2 contains some very important information. When all the chunks are inserted in HTML pages, all this information disappears into HTML code which only indicates how text should be rendered. A much better approach is to annotate the chunks, which is the process of adding a meaning to the content of a text using tags. The information identified during the Conceptual Design phase can be used to add semantic tags to the text. The publication chunk ORM diagram tells us of what a publication consists. All the objects and the relationships on this diagram are the only thing needed to annotate the publication chunk. The annotation process can therefore be described as tagging the chunk information with the object and relationship information of the conceptual design model of an object chunk.

For the annotation of the chunks of the WISE web site, the following technologies were used:

- **XML.** A markup language which stands for *eXtensible Markup Language* [8] and is designed to describe data. This language is used to annotate the chunks.
- **DTD.** A *Document Type Definition* language used to declare the legal structure of an XML document. DTD is used to validate the annotated XML chunks.
- **XSLT.** A language which stands for *Extensible Stylesheet Language Transform* [20] and is used for transforming XML documents into other XML documents. This technology transforms the annotated chunks to HTML.

These technologies are perfectly suited for the annotation of chunks. The XSLT language transforms the annotated XML chunks to HTML and therefore hides the XML for the visitors of the web site.

The implementation itself was pretty straightforward. The first thing we did was creating a global DTD containing all possible tags used to describe the data on the web site. These tags are retrieved by analyzing the Business Information Model which is constructed during the Conceptual Design phase. This model contains all the data objects and the relationships between the data of the whole web site and is thus exactly what we need for the construction of the DTD. When the DTD is defined, the object chunks can be annotated using the appropriate XML tags defined in the DTD. The last step is to write XSLT code to transform the XML chunks into HTML code.

Let's turn theory into practice with an example of an annotation process. This process will annotate a new object chunk: the member object chunk. This chunk contains information about a member of the WISE research group. We introduce this new object chunk because it contains less objects than the publication object chunk and is therefore a better choice for the demonstration of the annotation process.

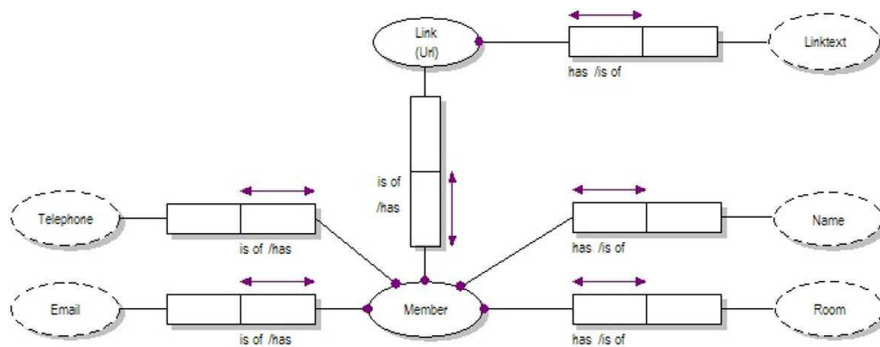


Figure 6.3: The member object chunk.

The first step of this annotation process is to analyze the ORM diagram of the member object chunk. From this ORM diagram, we can derive a DTD which defines the structure for this object chunk. This diagram can be seen on figure 6.3. Normally, the DTD is defined for the complete Business Information Model. For simplicity however, we only consider this fragment of the Business Information Model. The next step in the annotation process is the creation of the XML file, the creation of the content. This is an uncomplicated task where the member data is annotated with the corresponding XML tags defined in the DTD. Defining the XSL transformation of XML to HTML ends the annotation process. This step defines the output format of the chunk which is in this case HTML. The XML file containing the annotated object chunk and the XSLT file are then processed by an XSLT processor into a HTML page. The whole process of annotation, structuring and transformation of this member object chunk can be seen on figure 6.4. The arrow from the XML file to the DTD file denotes that the XML data is validated by the DTD.

In this section, we have seen the importance of the ORM diagrams of the object chunks. The annotation process can be summed up as follows:

1. Define the structure of the object chunk by deriving a DTD from the ORM diagrams.
2. Annotate the content using XML tags.
3. Create an XSL transformation which defines the output format.
4. Generate the output by processing the XML and XSLT files by an XSLT processor.

6.3.3 Graph structure implementation

The next step in the implementation process is designing a dynamic graph structure which mirrors the Navigation Model defined during the Conceptual Design phase. For the implementation of the WSDM graph structure, we used Java version 1.5. The reason why Java is chosen as implementation language is because Java provides Java Servlets. A Java Servlet is an application which runs

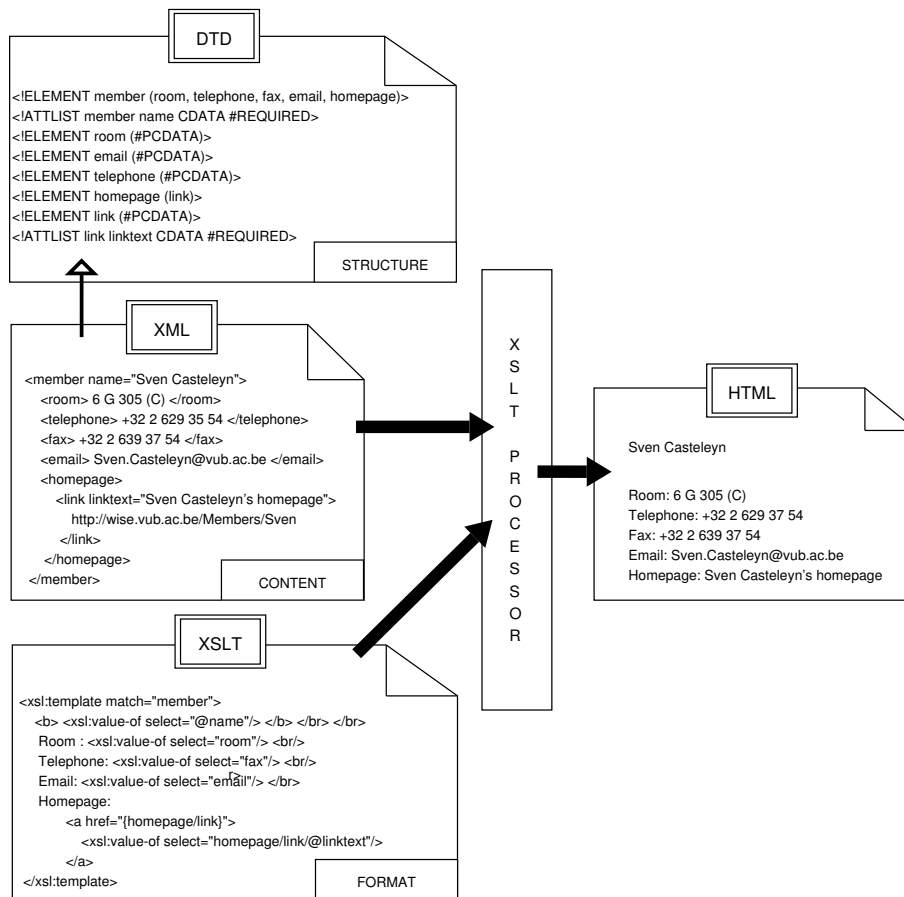


Figure 6.4: The annotation/transformation process for the member chunk.

on a web server and interacts with web clients, in this case the browser of the visitors. By using Java to implement the graph structure, Java Servlets can be used to handle HTTP requests and to retrieve the requested nodes and their chunks out of the graph structure holding the web site.

Recall that a Navigation Model contains nodes, chunks, links between nodes and connections between nodes and chunks. For the design of the graph structure, we initially designed three simple classes: a *Graph* class, a *Node* class and an *Edge* class. This graph structure encapsulated a graph containing nodes connected together by edges. The graph structure of the Navigation Model is more complex as there are two kind of edges (links and connections), and two kind of nodes (nodes and chunks). This more complex graph structure is modelled by subclassing the three base classes. This way, six new classes are designed: the *WSDMGraph* class which is a subclass of the *Graph* class, the *WSDMNode* and *WSDMChunk* classes which are subclasses of the *Node* class, the *WSDMLink* and *WSDMConnection* classes which are subclasses of the *Edge* class, and the *WSDMSessionLink* class which encapsulates the session links needed for

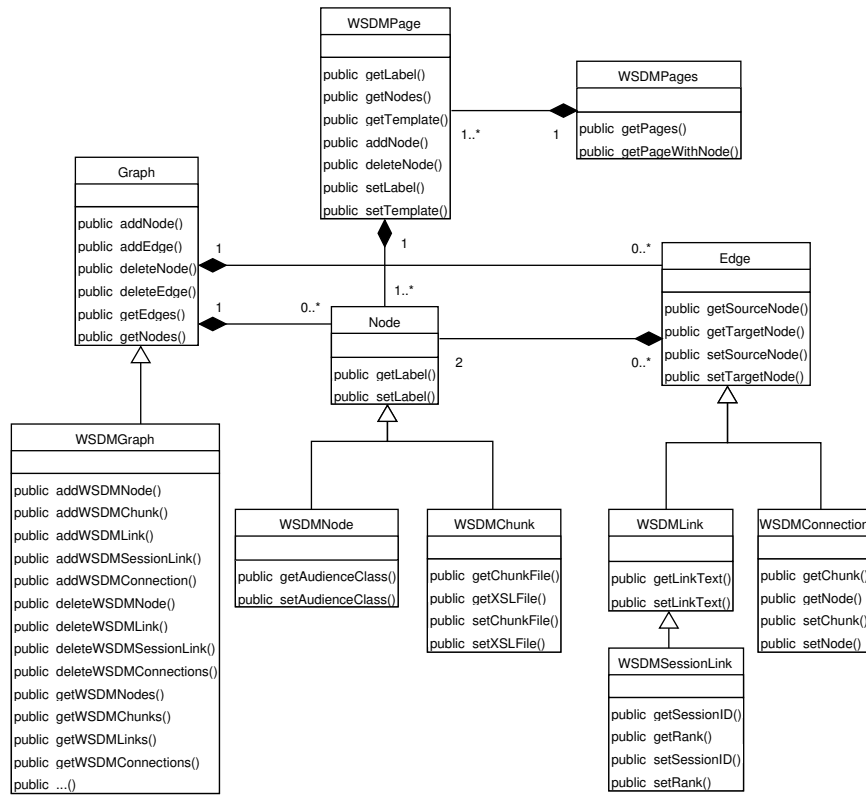


Figure 6.5: The design of the graph structure.

the link recommendations. These are all the classes needed to encapsulate the Navigation Model. To know which nodes map to which pages, classes for the encapsulation of pages have to be created. We created two classes for this purpose: a *WSDMPage* class which contains references to the nodes the page contains and a *WSDMPages* class which is a set containing all pages in the web site. A simplified UML diagram of these classes can be seen on figure 6.5.

Storing and initializing the graph structure

The graph structure described above has to be initialized and stored on permanent storage in some way. This permanent storage is necessary in case of a server crash and to be able write a new adapted web site to disk. The permanent storage used is an XML file containing the whole web site structure in XML format. When the web site is launched, the XML file is read and the graph structure is initialized. Whenever the graph structure changes at runtime, it is written back to the XML file so that the changes can not be lost when the server crashes. Note that this is evidently only done for permanent changes such as the addition of new links and not for temporal changes such as session links. The following shortened XML code shows how the graph structure can be stored in an XML file:

```

<graph>
  <node label="Root" audienceClass="Visitor" page="RootPage">
    <link linktext="About WISE">
      <node label="About WISE" page="AboutWISEPage">
        ...
      </node>
    </link>
    <link linktext="Members">
      <node label="Members" page="MembersPage">
        ...
      </node>
    </link>
    <link linktext="Information for Students">
      <node label="Students" audienceclass="Students"
        page="StudenstPage">
        ...
      </node>
    </link>
    <link linktext="Information for Researchers">
      <node label="Researchers" audienceclass="Researchers"
        page="ResearchersPage">
        ...
      </node>
    </link>
    <connection>
      <chunk label="Coordinates" xslFile="coordinates.xsl">
        <url>coordinates.xml</url>
      </chunk>
    </connection>
  </node>
</graph>

```

6.3.4 Running the web site

To run the web site, Java Servlets are used, as was already mentioned in section 6.3.3. To understand how the pages are constructed, let's take a look at the URL of the homepage of our experimental version of the WISE web site:

`http://wilma.vub.ac.be:8080/ts/servlet/MainServlet?label=Root`

MainServlet is the name of the Servlet which controls the web site. The question mark followed by *label=root* is a parameter mechanism that is used to select a node in the graph which is in this case the *Root* node. The Servlet then looks up the page which contains this node and constructs this page. This page construction can be summarized as follows:

1. All nodes on the page are retrieved.
2. All the chunks belonging to the retrieved nodes are collected.
3. Using an XSL transformer to process the chunks and their corresponding XSLT file, HTML code is constructed for each of these chunks.



Figure 6.6: The homepage of the WISE web site.

4. All links from the nodes of the page to other nodes are converted to hyperlinks.
5. The session ID of the visitor session which requested the page is retrieved and the session links for the session are fetched.
6. The whole HTML page is constructed by inserting chunks and links on their corresponding position¹ in the template for the page. The session links are inserted in a predefined area on the page.

The result of this process can be seen on figure 6.6 which shows the homepage of the WISE web site. This page contains one node, the *Root* node, and contains links to four other nodes and a connection to a chunk which is shown in the page. On the right of the page, the session links can be seen. Notice how the page structure exactly mirrors the structure of the XML code which we saw earlier on.

6.4 ASL Interpreter Implementation

The purpose of the ASL interpreter is to read ASL scripts and to execute the ASL statements in the scripts. These statements then act upon the graph structure so that ASL can be used to manipulate the structure of the web site. The implementation of the ASL interpreter can be divided in three parts:

¹This position of a link and chunk on the page is the same as its position in the XML file which contains the graph structure.

1. Construction of an ASL syntax in *Backus-Naur Form*. This is a widely used notation for describing exact syntax and grammar of a language.
2. Generation of an ASL lexer and parser with an ANTLR program, starting from the syntax constructed in the previous step. *ANTLR*² is a parser generator for Java.
3. Implementation of an ASL evaluator.

The second step outputs an Abstract Syntax Tree (AST). An *Abstract Syntax Tree* is a data structure which holds parsed data. Such an AST is then used by the evaluator which maps these AST's to their corresponding operations on the graph structure.

Elementary operations such as *addlink()* or set expressions such as *Nodes* can be easily mapped to their corresponding operations in the graph structure classes. The biggest challenge for the implementation of the interpreter however, is the evaluation of tracking variables. For the implementation of the ASL interpreter for the WISE web site, a tracking mechanism was used to store session information into a MySQL relational database. Two tables are used to store this session information. In the first table, session related information is stored such as the session ID and the audience class. The second table contains information for each page view, i.e. every time when a page is viewed information about the page, the referrer page, the followed link and the session ID of the session which visited the page is stored. This session information is used to evaluate the *accesses* tracking variable defined in section 5.2.2 of chapter 5. For the evaluation of this tracking variable, SQL was used to query this page information out of the database.

6.5 Integration

This chapter will be concluded with a global view on the integration of all components used to construct the adaptive WISE web site. Figure 6.7 shows the integration of all the components which make up the web site. The two computers on the figure indicate the view points of respectively the web master and a visitor to the web site.

For the web master, an ASL editor is created. This is a password protected web page which makes it possible to edit and evaluate the main ASL script. When the web master presses the designated evaluation button in the ASL editor, the script is sent to the interpreter for evaluation. For this evaluation, the interpreter sends queries to the relational database for the retrieval of session data. The ASL script evaluated by the interpreter also acts upon the graph data structure as can be seen on the figure.

On the other side of the web site we have the visitors. When a visitor visits the web site and sends a HTTP request, session and page view data is sent to the tracker which stores this session data in the relational database. The next thing which happens is the page construction. The requested web site data is

²ANTLR is available at <http://www.antlr.org>

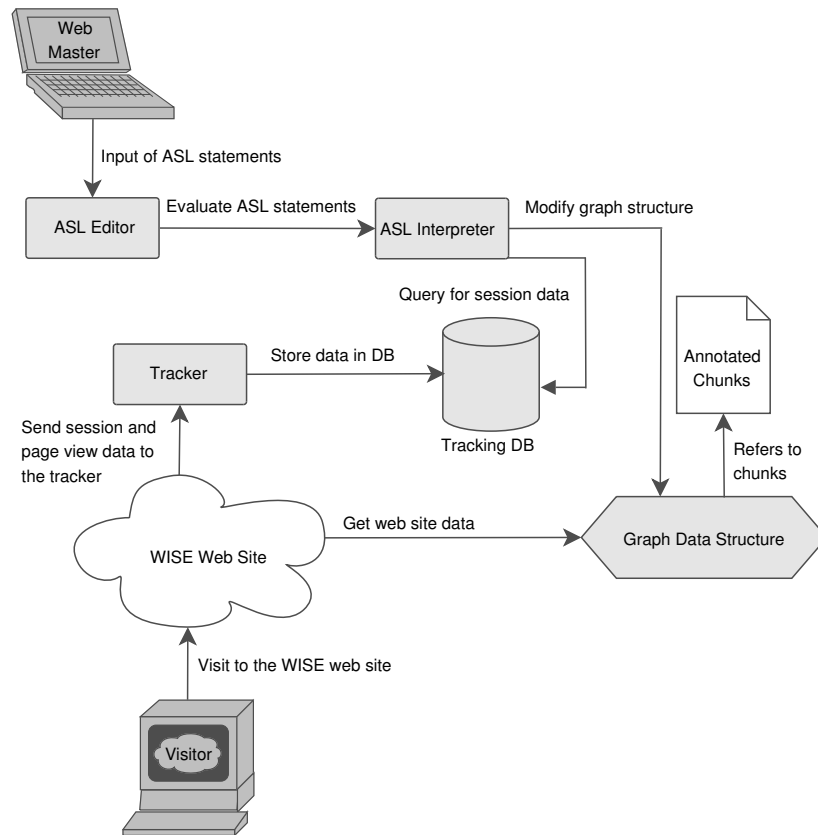


Figure 6.7: The integration of all components.

retrieved out of the graph structure as we have seen in section 6.3.4. This web site data includes the annotated chunks which are stored in XML files and which are referenced by the graph.

Chapter 7

Evaluation of the Algorithms

7.1 Introduction

In this chapter, we will evaluate and compare the two algorithms described in chapter 5: the k nearest neighbour link recommendation algorithm and the Bayesian link recommendation algorithm. By performing experiments with visitors, the usefulness of the proposed link recommendation techniques will be determined. This usefulness is determined by counting the number of clicks and the amount of time needed to find the information for which a visitor is looking for. Our goal is to show that the use of link recommendations heavily decreases the number of clicks and amount of time needed to find the information the visitor needs.

The remainder of this chapter is organized as follows. Section 7.2 describes the performed experiments in detail. Section 7.3 lists the results of these tests and in the last section, section 7.4, the results are interpreted and evaluated.

7.2 Performed Experiments

The experiments were performed on the WISE web site which is implemented as described in the previous chapter. To test the algorithms, we constructed some related tasks which had to be performed by test groups. The following three experiments were performed by three different test groups of 6 people each:

1. The first group performed the tasks without link recommendations.
2. The second group performed the tasks with link recommendations generated using the k nearest neighbour link recommendation algorithm.
3. The third group performed the tasks with link recommendations generated using the Bayesian link recommendation algorithm.

The three groups of 6 people were made up out of 18 people which were all unfamiliar with the WISE web site. The first experiment results in 6 sessions which are added to the already available session data which already contained 20 standard sessions which were not lead by means of predefined tasks. The purpose of this first experiment is two-fold:

- Monitoring the number of clicks and time needed to perform the tasks without link recommendations.
- Providing session data about the related tasks on which the link recommendations will be based.

Monitoring how long it takes to perform the tasks without link recommendations is needed to be able to determine if link recommendations reduce the effort to find the needed information. The session data of the tasks is also needed, otherwise no recommendations can be made. When the link recommendation algorithm “detects” that these six sessions browsed for the same information, it will recommend the pages which were accessed during these sessions but not yet accessed during the current session. The second and the third experiment are obviously necessary in order to be able to measure the impact of link recommendations.

We already mentioned in the previous chapter that the WISE web site is rather small and contains only 17 pages. The *Student* and *Researcher* audience track consist of respectively 11 pages and 5 pages. Add to these numbers the 2 pages which are part of the *Visitor* audience track and thus also meant for both audience class, each student or researcher visiting the web site should find his information respectively 13 pages and 7 pages. To follow the audience-driven character which places related pages into an audience track, the tasks should actually be situated in one audience track. Because we wanted to compose tasks which involved seeking information on more than 5 pages, the information which had to be found for the experiments was situated on pages in all audience tracks.

For each experiment, the following tasks were given:

1. Find the page containing information about the course entitled “Design Methods of Internet-based Systems”.
2. Find the page on which all publications of the WISE research group are listed.
3. Find the page containing information about the course entitled “Inleiding tot Databanken”.
4. Find the page containing general information about the WISE research group.
5. Find the page which lists all current thesis students at the WISE research group.
6. Find the page containing the coordinates of Sven Casteleyn, a member of the WISE research group.

All these tasks lead to a specific page. By performing the first experiment, six sessions are created with a like-minded purpose. Without link recommendations, the minimum number of clicks needed to find all this information is 25. Note that the monitored clicks include both normal link clicks and “back button clicks” which are the clicks which bring a user to the previous page he visited.

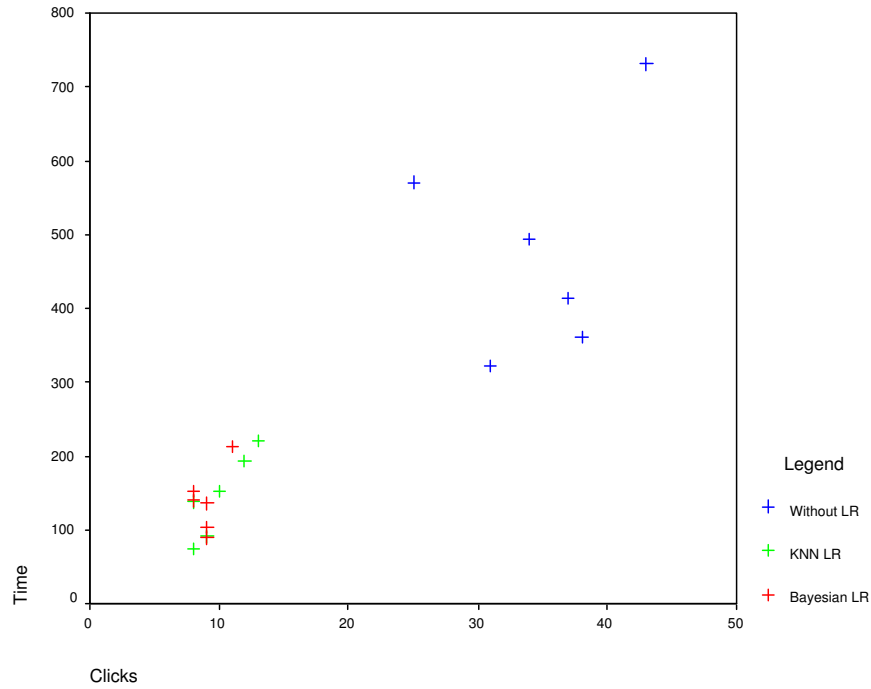


Figure 7.1: A scatter plot of the experiments' data.

The threshold of both algorithms was set to 0.49. This means that only pages with an interestingness value of 50 % or more are recommended. For the k nearest neighbour algorithm, the value of k was set to 4 because of the small size of the session data.

7.3 Results of the Experiments

The results of the experiments are presented on figure 7.1 which is a scatter plot of the time versus the number of clicks needed for all 18 sessions. The legend shows which sessions belongs to which experiment. It's already obvious that there is a big difference between the number of clicks and time needed with and without the aid of link recommendations.

A numerical summary of these results is provided in tables 7.1 and 7.2. Table 7.1 shows the mean number of clicks needed by each of the three experiments along with its standard deviation. Table 7.2 does the same for the time needed to find all the information.

| Experiment | Mean | Standard Deviation |
|-----------------------------------|-------|--------------------|
| Without Link Recommendations | 35,67 | 6,22 |
| With KNN Link Recommendations | 10 | 2,10 |
| With Bayesian Link Recommendation | 9 | 1,10 |

Table 7.1: The number of clicks needed for each experiment.

| Experiment | Mean | Standard Deviation |
|-----------------------------------|-----------|--------------------|
| Without Link Recommendations | 8.02 min. | 3.31 min. |
| With KNN Link Recommendations | 2.25 min. | 0.56 min. |
| With Bayesian Link Recommendation | 2.19 min. | 0.43 min. |

Table 7.2: The time needed for each experiment.

7.4 Interpretation of the Results

Both tables show that the effect of providing link recommendations is astonishing. The people who performed the tasks with the aid of link recommendations required only a quarter of the amount of clicks and time needed by those who performed the tasks without link recommendations. To test if this difference is significant, we will compare the results of the experiments with and without link recommendations. We will only compare the results of the link recommendation algorithm which gave the worst results of both algorithms, which is the k nearest neighbour link recommendation algorithm, to the results of the experiment without the aid of link recommendations. The reason why is because when this difference turns out to be significant, the difference to the other algorithm which performs better will be even more significant. To test if the experiment with the k nearest neighbour link recommendation algorithm gives significantly better results than the experiment without link recommendations we will use a widely known statistical test.

The statistical test which we will use is the independent samples t -test [24]. This statistical test compares the means of two independent normally distributed data sets and determines if the distance between the means is significant. Both data sets should thus be independent, which is the case for both data sets of experiments 1 and 2 as no people took part in both experiments. We will perform this test on both the clicks and time data sets.

When a statistical test is performed, a null hypothesis is tested. This null hypothesis is the antithesis of whatever hypothesis we want to test. The hypothesis which we want to test is the following:

There is a significant difference in the number of clicks and the amount of time required to find information between finding the information with the aid of link recommendations generated using the k nearest neighbour link recommendation algorithm and finding the information without the aid of link recommendations.

The null hypothesis states the contrary, that there is no significant difference between both.

We will not list the whole statistical process but show the results of both tests. Note that we first successfully tested if the data is normally distributed because the t -test is only valid for normally distributed data. The SPSS statistical package is used to perform the t -test. Before we can run the t -test, we need to set the value of p . This value of p is the probability that the difference between means in a sample occurred purely by chance, and that in the population from which the sample was drawn, no such relationship or differences actually exist. In literature, researchers usually conclude that a real difference exists if $p < 0.05$. We will therefore also set the value of p to 0.05. If the significance of the test is smaller than p , we can reject the null hypothesis. The results of both tests are displayed in table 7.3 and table 7.4.

| Levene's Test | | t-Test | |
|---------------|-------|--------|-------|
| F | Sig. | t | Sig. |
| 3,971 | 0,074 | 9,207 | 0,000 |

Table 7.3: The results of the t -test for the number of clicks (exp. 1 vs. 2).

| Levene's Test | | t-Test | |
|---------------|-------|--------|-------|
| F | Sig. | t | Sig. |
| 4,188 | 0,068 | 5,108 | 0,000 |

Table 7.4: The results of the t -test for the amount of time (exp. 1 vs. 2)..

These tables show the results of two tests. The first test, Levene's test, is used to test if equality of variances can be assumed. If the significance value is larger than p , the variances are not significantly larger. This enabled us to perform the t -test which assumes equal variances. Note that the F value is the ratio of the variance between the two groups to the variance within each individual group. This is of no further importance.

The significance value of both t -tests tells us that we can reject the null hypothesis as both values are equal to 0. This is true for both link recommendation algorithms as the Bayesian link recommendation algorithm performs even better than the k nearest neighbour link recommendation algorithm. Therefore, we can state that the t -test shows that the aid of link recommendations significantly increases the task performance (which is measured by the number of clicks and the amount of time).

The difference between both algorithms seems to be much less significant. We can see that the Bayesian link recommendation algorithm has the lowest mean and standard deviation for both the number of clicks and the amount of time needed. To test if the Bayesian link recommendation algorithm gives significantly better results than the k nearest neighbour link recommendation

we will use the same t -test.

This time, the hypothesis we want to test is the following one:

There is a significant difference in the number of clicks and the amount of time required to find information between finding the information with the aid of link recommendations generated using the Bayesian link recommendation algorithm and finding the information with the aid of link recommendations generated using the k nearest neighbour link recommendation algorithm.

Recall that the null hypothesis says the contrary, that there is no significant difference between both algorithms.

The results of both t -test are listed in table 7.5 and table 7.6.

| Levene's Test | | t-Test | |
|---------------|-------|--------|-------|
| F | Sig. | t | Sig. |
| 3,462 | 0,092 | 1,035 | 0,325 |

Table 7.5: The results of the t -test for the number of clicks (exp. 2 vs. 3).

| Levene's Test | | t-Test | |
|---------------|-------|--------|-------|
| F | Sig. | t | Sig. |
| 0,649 | 0,439 | 0,219 | 0,831 |

Table 7.6: The results of the t -test for the amount of time (exp. 2 vs. 3).

First of all, we can assume equal variances according to Levene's Test. The significance value of both t -tests tells us that we can not reject the null hypothesis as both values are much larger than 0,05. Therefore, there is no significant difference between the impact on the task performance of both link recommendation algorithms.

We can conclude that both link recommendation algorithms have no significant different impact on the amount of time and the number of clicks needed to find the information. What is most important is that the use of link recommendations heavily reduces the visitors' effort to find information. Based on the experiments, it takes 4 times less effort which is a big improvement.

Although the results are spectacular, we need to keep in mind that the WISE web site is small and that the experiments are performed with small groups. It's questionable if we would have obtained the same impressive results with groups of more than 100 people, a web site containing 200 pages and multiple experiments consisting of different tasks. Although the small scale of the experiments, we believe that large experiments will show that both algorithms will strongly reduce the number of clicks and amount of time needed to search information.

Chapter 8

Conclusions

8.1 Introduction

To conclude this dissertation, an overview of the achievements will be given on which we will draw the conclusions. We will also discuss possible future work closely related to the subject of this thesis.

The main goal of this thesis was to incorporate design time support for session-based adaptation into WSDM with which it should be able to specify link recommendation algorithms. Once this design time support was available, our goal was to specify some link recommendation algorithms and to test and evaluate them. An additional requirement was that the link recommendation techniques had to be based on user access information instead of on a user model. This was necessary in order to keep faithful to the philosophy of adaptation in WSDM.

In section 8.2, we recapitulate how we achieved the goal of this thesis with a summary of all chapters. The next section, section 8.4, lists the achievements of this dissertation. Section 8.4 presents possible future work.

8.2 Summary

In chapters 2, 3 and 4, the reader was introduced to the necessary background information needed for the research part of this thesis. Chapter 2 introduced WSDM and its adaptation specification language ASL, which both served as the framework for our work. In chapter 3, Bayes' Theorem, the Naive Bayes Classifier and the k-Nearest Neighbour Classifier were introduced. These machine learning techniques have been used as the foundation for the link recommendation techniques which we proposed later on in the research part. The last background chapter, chapter 4, gave an overview of the existing web site adaptation methods and more specifically the existing link recommendation techniques.

The most important chapter of this dissertation was chapter 5 in which we first extended WSDM and ASL to provide design time support for session-based adaptation. Using these extensions we then proposed two link recommendation

algorithms: the k nearest neighbour link recommendation algorithm and the Bayesian link recommendation algorithm. These were both discussed in detail and expressed in ASL. To be able to test these algorithms, we implemented a web site designed using WSDM which was described in detail in chapter 6. This description included an overview of the design process and the implementation process of all the elements of the web site and the ASL interpreter. The evaluation of both algorithms was carried out in chapter 7. In this chapter, we described the performed experiments and interpreted the results.

8.3 Achievements

In this section we list the achievements which are the result of this thesis. The achievements can be summed up as follows:

- **Design support for session-based adaptation in WSDM:** WSDM and ASL are extended with design time support for session-based adaptation. These extensions include the ability to access session information of all previous sessions and to add session links at design time. Using these extensions, it is possible to personalize the web site for individual users in addition to the already available support for providing customization for all users.
- **Two reusable link recommendation algorithms:** the two link recommendation algorithms first of all show how session-based adaptation techniques can be expressed in ASL. They both only need a very simple format of session information and don't require any user model or feedback from the user. We have also shown that both the number of clicks and time needed to search information can be reduced significantly when link recommendations are provided.
- **A prototype implementation of a WSDM web site and ASL evaluator:** we implemented the WSDM web site and ASL evaluator in such a way that they can be re-used easily. The same implementation process can be applied to another web site designed using the WSDM approach. The ASL evaluator can also easily be adapted to changes in the ASL syntax and semantics.

8.4 Future Work

Based on the session-based adaptation idea which is explored throughout this dissertation, some interesting future work can be done. Possible future work includes the following issues:

- **More advanced experiments:** to get a better view of the real impact of both link recommendation algorithms, more advanced experiments should be conducted. This includes extending the number of users which perform the experiments, increasing the size of the web site and performing more than one experiment. Other future experiments can test the algorithms with different threshold values and with the extensions discussed in sections 5.3.1.4 and 5.3.2.4.

- **Other link recommendation techniques:** both link recommendation techniques use a simple session matrix. A big challenge would be to specify more advanced link recommendation techniques such as content-based link recommendation techniques and hybrid techniques. This would require many extensions to ASL such as the ability to compare nodes on the content level.
- **Specifying other session-based adaptation methods:** the link recommendation techniques are both a adaptive link generation methods. This is only one of the web site adaptation methods which we have seen in chapter 4. In the future, other personalization methods could be integrated into WSDM. The biggest problem is that most of these other methods rely on a user model. Specifying a user model and making it accessible in ASL will make it possible to specify these methods. Incorporating some of these other adaptation methods without a user model can also be a very challenging task.

It's obvious that there are many possible extensions which can be investigated in the future. We think that the biggest future challenge among the ones listed above is the incorporation of the other web site adaptation methods. When these are all specified in ASL, WSDM will offer a large library of re-usable web site adaptation methods which are a very valuable tool for every web designer.

Appendix A

ASL BNF Specification¹

```
<adaptationPolicy> ::= (<script> ';')*  
                        (<adaptationSpecification> ';')*  
  
<adaptationSpecification> ::= 'when' <trigger> 'do' <adaptationStrategy>  
  
<trigger> ::= <timeEvent>  
<trigger> ::= <systemEvent>  
<trigger> ::= <userEvent> ['on' <reference>]  
<trigger> ::= <condition>  
  
<userEvent> ::= 'click' | 'sessionStart' | 'sessionEnd' | 'load'  
  
<systemEvent> ::= 'initialization'  
  
<timeEvent> ::= <dateSpecifier> | ['every'] <timeExpression> ['from now']  
  
<dateSpecifier> ::= <number> <month> <number>  
  
<month> ::= 'January' | 'February' | 'March' | 'April' | 'May' | 'June' | 'July'  
           'August' | 'September' | 'October' | 'November' | 'December'  
  
<timeExpression> ::= <timePrimitive>*  
  
<timePrimitive> ::= <number> <timeUnit>  
<timePrimitive> ::= <day>  
<timePrimitive> ::= <month>  
  
<timeUnit> ::= 'seconds' | 'minutes' | 'hours' | 'days' | 'weeks' | 'months' | 'years'  
  
<day> ::= 'Monday' | 'Tuesday' | 'Wednesday' | 'Thursday' | 'Friday' |  
          'Saturday' | 'Sunday'  
  
<adaptationStrategy> ::= <ruleSequence> | <rule>  
  
<ruleSequence> ::= 'begin' <rule> [ ';' <rule> ]* 'end'
```

¹The changes to ASL are emphasized in *italics*.

```

<rule> ::= <foreachRule> | <simpleRule>

<foreachRule> ::= 'forEach' <setIterator> [<setIterator>]* ';'
                (<adaptationStrategy>)

<setIterator> ::= <reference> 'in' <setExpression>

<simpleRule> ::= <ifStatement> | <trackingVariableStatement> |
                <letStatement> | <assignment> |
                <setLetStatement> | <setAssignment> |
                <nativeOperation> | <callStatement>

<ifStatement> ::= 'if' <condition> 'then' <adaptationStrategy>

<letStatement> ::= 'let' <variable> ['be' <expression>]

<setLetStatement> ::= 'let' <setVariable> ['be' <setExpression>]

<addTrackingVariable> ::= 'addTrackingVariable' <trackingVariable>
                        ['initialized on' <expression>]

<monitorStatement> ::= 'monitor' <userEvent> ['on' <reference>]
                    ['if' <condition>] 'do'
                    <adaptationStrategy>

<trackingVariableAssignment> ::= <trackingVariable> ':' <expression>

<assignment> ::= <trackingVariableAssignment>
<assignment> ::= <variable> ':' <expression>

<setAssignment> ::= <setVariable> ':' <setExpression>

<script> ::= 'script' <reference> '(' (<parameter> [',' <parameter>]* ')' ':'
                <adaptationStrategy>)

<parameter> ::= <variable> | <setVariable>

<callStatement> ::= 'call' <reference> '(' (<expression> | <setExpression>)* ')'

<condition> ::= <expression>

<expression> ::= <comparand> [<comparator> <comparand>]*

<comparand> ::= <term> [<adder> <term>]*

<term> ::= <factor> [<multiplier> <factor>]*

<factor> ::= <primitive>
<factor> ::= <not> <primitive>

<primitive> ::= <number>
<primitive> ::= <boolean>
<primitive> ::= <reference>
<primitive> ::= <variable>

```

```

<primitive> ::= <trackingVariable>
<primitive> ::= <nativeFunction>
<primitive> ::= <runtimeConstant>
<primitive> ::= <statisticalOperator>
<primitive> ::= '(' <expression> ')'
<primitive> ::= projection(<setExpression>)

<comparator> ::= '<' | '=' | '>' | '<=' | '>=' | '!='

<adder> ::= 'OR' | '+' | '-'

<multiplier> ::= 'AND' | '/' | '*'

<not> ::= 'NOT' | '-'

<trackingVariable> ::= <reference> [ '[' <reference> [ ',' <reference> * ] ']' ].<reference>

<variable> ::= <reference>

<setVariable> ::= <capitalizedLetter> [<letter> | <digit>]*

<reference> ::= [<nonCapitalizedLetter> | <digit>]* <letter> [<letter> | <digit>]*

<nativeSet> ::= 'Nodes' | 'Chunks' | 'Links' | 'Connections' | 'Pages' |
               'AudienceClasses' | 'Sessions'

<setCreator> ::= 'nodesInAudienceTrack(' <expression> ')' |
               'chunksInAudienceTrack(' <expression> ')' |
               'chunksFromNode(' <expression> ')' |
               'nodesLinkedTo(' <expression> , <expression> ')' |
               'nodesLinkedFrom(' <expression> , <expression> ')' |
               'nodesFromChunk(' <expression> ')' |
               'ALL' <expression> |
               'addElement(' <setExpression> , <expression> ')' |
               'addElementAtIndex(' <setExpression> , <expression> ,
                                   <expression> ')' |
               'subset(' <setExpression> , <expression> , <expression> ')'

<setOperation> ::= <setSorter> | <setFilter> | <setMap>

<setSorter> ::= 'SORT ON' <reference> ':' <trackingVariable>
<setFilter> ::= 'FILTER ON' <reference> ':' <condition>
<setMap> ::= 'MAP ON' <reference> ':' <expression>

<setExpression> ::= <setComparand> [<setComparator> <setComparand>]*

<setComparand> ::= <setTerm> [<setAdder> <setTerm>]*

<setTerm> ::= <setPrimitive> [<setMultiplier> <setPrimitive>]*

<setPrimitive> ::= <set>
<setPrimitive> ::= (<setExpression>)
<setPrimitive> ::= <setExpression> '[' <setOperation> [ ';' <setOperation> ] * ']'

```

```

<set> ::= <nativeSet> | <setCreator> | <setOperation> |
        '{' <expression> ['<expression>']* '}'

<setComparator> ::= '='

<setAdder> ::= 'DIFFERENCE' | '+' | '-'

<setMultiplier> ::= 'UNION' | 'INTERSECT' | '*' | '/'

<nativeOperation> ::= 'deleteNode('<expression>')' |
        'addNode('<expression>')' |
        'connect('<expression>','<expression>')' |
        'disconnect('<expression>','<expression>')' |
        'deleteLink('<expression>','<expression>','<expression>')' |
        'addLink('<expression>','<expression>','<expression>')'
        ['as' <reference>] |
        'addPage('<expression>')' |
        'deleteNodeFromPage('<expression>','<expression>')' |
        'addNodeToPage('<expression>','<expression>')' |
        'alert('<expression> | <string> )* '}' |
        'deleteSessionLink('<expression>','<expression>','
        <expression>','<expression>')' |
        'addSessionLink('<expression>','<expression>','
        <expression>','<expression>')'
        ['as' <reference>] |
        'clearSessionLinks('<expression>','<expression>')'

<nativeFunction> ::= 'root('<expression>')' | 'audienceClass('<expression>')' |
        'audienceTrack('<expression>')' | 'currentAudienceClass()' |
        'inAudienceTrack?('<expression>','<expression>')' |
        'linked?('<expression>','<expression>','<expression>')' |
        'project('<setExpression>','<expression>')' |
        'target('<expression>')' | 'source('<expression>')' |
        'linktype('<expression>')' | 'currentSession()' |
        'currentNode()'

<statisticalOperators> ::= 'average('<setExpression>')' | 'max('<setExpression>')' |
        'min('<setExpression>')' | 'range('<setExpression>')' |
        'mad('<setExpression>')' | 'median('<setExpression>')' |
        'middle('<setExpression>')' | 'stdev('<setExpression>')' |
        'length('<setExpression>')'

<boolean> ::= 'false' | 'true'

<string> ::= "" [<letter> | <digit>]* <letter> [<letter> | <digit>]* ""

<letter> ::= <nonCapitalizedLetter> | <capitalizedLetter>

<nonCapitalizedLetter> ::= 'a' | 'b' | 'c' | 'd' | 'e' | 'f' | 'g' | 'h' | 'i' | 'j' | 'k' |
        'l' | 'm' | 'n' | 'o' | 'p' | 'q' | 'r' | 's' | 't' | 'u' | 'v' |
        'w' | 'x' | 'y' | 'z'

<capitalizedLetter> ::= 'A' | 'B' | 'C' | 'D' | 'E' | 'F' | 'G' | 'H' | 'I' | 'J' | 'K' |
        'L' | 'M' | 'N' | 'O' | 'P' | 'Q' | 'R' | 'S' | 'T' | 'U' |

```

'V' | 'W' | 'X' | 'Y' | 'Z'

<number> ::= <digit> [<digit>]*

<digit> ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'

Bibliography

- [1] Aggarwal, C. C., Wolf, J. L., Wu K., Yu, P. S. (1999). Horting Hatches an Egg: A New Graph-theoretic Approach to Collaborative Filtering. In *Proceedings of the ACM KDD99 Conference*, pp. 201- 212.
- [2] Agrawal, R., Imielinski, T, Swami, A.N. (1993). Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pp. 207-216.
- [3] Ayars, J. et al (Ed) (2001). Synchronized Multimedia Integration Language (SMIL 2.0). W3C Recommendation 07 August 2001. Available at <http://www.w3.org/TR/2001/REC-smil20-20010807/>.
- [4] Backus, J. W., Wegstein, J. H., van Wijngaarden, A., Woodger, M., Bauer, F. L., Green, J., Katz, C, McCarthy, J., Perlis, A. J., Rutishauser, H., Samelson, K. and Vauquois, B. (1960). Revised Report on the Algorithmic Language ALGOL 60. In *Communications of the ACM*, Volume 3 Number 5, pp. 299-314, ACM Press, ISSN 0001-0782.
- [5] Balabanovic, M., Shoham, Y. (1997). Fab: content-based collaborative recommendation. In *Communications of the ACM*, Volume 40, Number 4, pp. 66-72.
- [6] Baudisch, P. (2001). *Dynamic Information Filtering*. Ph.D. Thesis, Technical University of Darmstadt.
- [7] Bieliková, M., Habala, R. (2004). Time-Based Extensions to Adaptation Techniques. In *Proc. of 3rd Int. Conference on Adaptive Hypermedia and Adaptive Web-Based Systems AH*, pp. 376-379, Springer-Verlag, LNCS 3137, ISBN 3-540-22895-0.
- [8] Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E. (1998). Extensible Markup Language (XML) 1.0 (Second Edition). Available at <http://www.w3.org/TR/REC-xml> .
- [9] Breese, J. S., Heckerman, D., Kadie, C. (1998). Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In *Proceeding of the Fourteenth Conference on Uncertainty in Artificial Intelligence*.
- [10] Brusilovsky, P. (1997). Efficient techniques for adaptive hypermedia. In *Intelligent Hypertext: Advanced Techniques for the World Wide Web*, pp. 12-30, Lecture Notes in Computer Science, Volume 1326, Springer-Verlag, ISBN 3-540-63637-4.

- [11] Brusilovsky, P. (1998). Methods and techniques of adaptive hypermedia. In *Adaptive Hypertext and Hypermedia*, pp. 1-43, Kluwer Academic Publishers, ISBN 0-7923-4843-5.
- [12] Brusilovsky, P. (2001). Adaptive hypermedia. In *User Modeling and User-Adapted Interaction, Ten Year Anniversary Issue 11*, pp. 87-110.
- [13] Brusilovsky, P. (2004). Adaptive navigation support: From adaptive hypermedia to the adaptive Web and beyond. In *Psychology Journal*, Volume 2, Number 1, pp. 7-23.
- [14] Burke, R. D. (2002). Hybrid Recommender Systems: Survey and Experiments. In *User Modeling and User-Adapted Interaction*, Volume 12, Number 4, pp. 331-370, ISSN 0924-1868.
- [15] Casteleyn, S., De Troyer, O. (2001). Structuring Web Sites Using Audience Class Hierarchies. In *Conceptual Modeling for New Information Systems Technologies, ER 2001 Workshops, HUMACS, DASWIS, ECOMO, and DAMA, Lecture Notes in Computer Science*, Volume 2465, Springer-Verlag, ISBN 3-540-44-122-0.
- [16] Casteleyn, S., De Troyer, O., Brockmans, S. (2003). Design Time Support for Adaptive Behaviour in Web Sites. In *Proceedings of the 18th ACM Symposium on Applied Computing*, pp. 1222 - 1228, ACM Press, ISBN 1-58113-624-2.
- [17] Casteleyn, S., Garrigós, I., De Troyer, O. (2004). Using Adaptive Techniques to Validate and Correct an Audience Driven Design of Web Sites. In *Proceedings of the ICWE 2004 Conference, Lecture Notes in Computer Science 3140*, pp. 55-59, Springer, ISBN 3-540-22511-0.
- [18] Casteleyn, S. (2005). *Designer Specified Self Re-organizing Websites*. Ph. D. Dissertation, Vrije Universiteit Brussel.
- [19] Ceri, S., Fraternali, P., Bongio, A. (2000). Web Modeling Language (WebML): a modeling language for designing Web sites. In *WWW9 Conference, First ICSE Workshop on Web Engineering, International Conference on Software Engineering*.
- [20] Clark, J. (1999). XSL Transformations (XSLT) 1.0. Available at <http://www.w3.org/TR/xslt>.
- [21] Conklin, E.J. (1987). Hypertext: An introduction and survey. In *IEEE Computer*, Volume 2, Number 9, pp. 17-41.
- [22] Cormen, T.H., Charles, E. L, Rivest. R. L. (1990). *Introduction to Algorithms*. The MIT Electrical Engineering and Computer Science Series. MIT Press, ISBN 0-262-03141-8.
- [23] Cristianini, N., Shawe-Taylor, J. (2000). *An introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, ISBN 0521780195.
- [24] Crow, E. L., Davis, F. A., Maxfield, M. W. (1960). *Statistics Manual*. New York, Dover Publications, ISBN 048660599X.

- [25] Dale, R., Green, S.J., Milosavljevic, M., Paris, C., Verspoor, C., Williams, S. (1998). Using Natural Language Generation Techniques to Produce Virtual Documents. In *Proceedings of the 3rd Australian Document Computing Symposium*.
- [26] De Troyer, O., Leune, C. (1998). WSDM: A User-Centered Design Method for Web Sites. In *Computer Networks and ISDN systems, Proceedings of the 7th International World Wide Web Conference*, pp. 85 - 94, Elsevier.
- [27] De Troyer, O. (2001). Audience-driven web design. In *Information modelling in the new millennium*, IDEA Group Publishing, ISBN 1-878289-77-2.
- [28] De Troyer, O., Casteleyn, S. (2001).: The Conference Review System with WSDM. In *First International Workshop on Web-Oriented Software Technology, IWWOST01*.
- [29] De Troyer, O., Casteleyn, S. (2003). Modeling Complex Processes for Web Applications using WSDM. In *Proceedings of the Third International Workshop on Web-Oriented Software Technologies* (held in conjunction with ICWE2003), IWWOST2003.
- [30] De Vrieze, P. , Van Bommel, P., Klok, J., Van der Weide, Th.P. (2004). Adaptation in Multimedia Systems. In *Multimedia Tools and Applications*, Volume 25, Number 3, pp. 333-343.
- [31] Domingos, P. , Pazzani, M. (1997). On the optimality of the simple Bayesian classifier under zero-one loss. In *Machine Learning*, Volume 29, number 2-3, pp. 103-130.
- [32] Halpin, T. (2001). *Information Modeling and Relational Databases: From Conceptual Analysis to Logical Design*, 1st Edition, Morgan Kaufmann Publishers, ISBN 1558606726.
- [33] Joachims, T. (1997). A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In *Proceedings of the 14th International Conference on Machine Learning*, pp.143-151.
- [34] Joachims, T. (1998). Text categorization with support vector machines. In *Proceedings of the 10th European Conference on Machine Learning (ECML'98)*, pp. 137-142, Springer.
- [35] Karypis, G. (2001). Evaluation of Item-Based Top-N Recommendation Algorithms. In *Proceedings of CIKM*, pp. 247-254, ACM Press.
- [36] Kilfoil, M., Ghorbani, A., Xing, W., Lei, Z., Lu, J., Zhang, J., Xu, X. (2003). Toward an Adaptive Web: The State of the Art and Science. In *Proceedings of the 1st Annual Conference on Communication Networks & Services Research (CNSR 2003)*, pp. 119-130.
- [37] Kobsa, A., Koenemann, J., Pohl, W. (2002). Personalized hypermedia and international privacy. in *Communications of the ACM*, Volume 45, Number 5, pp. 64-67.

-
- [38] Koychev, I., Schwab, I. (2000). Adaptation to Drifting User's Interests. In *Proceedings ECML2000/MLnet workshop "ML in the New Information Age"*.
 - [39] Linden, G., Smith, B., York, J. (2003). Amazon.com Recommendations: Item to Item Collaborative Filtering. In *IEEE Internet Computing*, Volume 7, Number 1, pp. 76-80, ISSN 1089-7801.
 - [40] Melville, P. , Mooney, R. J., Nagarajan, R. (2001). Content-boosted collaborative filtering. In *Proceedings of the SIGIR Workshop on Recommender Systems*.
 - [41] Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill Higher Education, ISBN 0070428077.
 - [42] Montaner, M., López, B. , de la Rosa, J.L. (2003). A Taxonomy of Personalized Agents on the Internet. In *Artificial Intelligence Review*, Volume 19, Number 4, Kluwer Academic Publishers, ISSN 0269-2821.
 - [43] Oard, D. W., Kim, J. (1998). Implicit feedback for recommender systems. In *Proceedings of the AAAI Workshop Enabling*.
 - [44] Paterno F. (2000). *Model-Based Design and Evaluation of Interactive Applications*. Springer Verlag, ISBN 1852331550.
 - [45] Pazzani, M. J. (1999). A framework for collaborative, content-based and demographic filtering. In *Artificial Intelligence Review*, Volume 13, Number 5-6, pp. 393-408, ISSN 0269-2821.
 - [46] Perkowit, M., Etzioni, O. (1997). Adaptive Web Sites: an AI Challenge. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, Volume 1, pp. 16-23.
 - [47] Perkowit, M., Etzioni, O. (1997). Adaptive Sites: Automatically Learning from User Access Patterns (poster). In *Proceedings of WWW6*.
 - [48] Plessers, P., Casteleyn, S., Yesilada, Y., De Troyer, O., Stevens, R., Harper, S., Goble, C. (2005). Accessibility: A Web Engineering Approach. In *Proceedings of the 14th International World Wide Web Conference (WWW2005)*, pp. 353-362, ACM Press, ISBN 1-59593-046-9.
 - [49] Prabhakaran, B. (2000). Adaptive Multimedia Presentation Strategies. In *Multimedia Tools and Applications*, Volume 12, Number 2-3, pp. 281-298, ISSN 1380-7501.
 - [50] Resnick, P., Iacovou, N., Suchak, M., Bergstorm, P., Riedl, J. (1994). GroupLens: An open architecture for collaborative filtering of netnews. In *Proceedings of ACM Conference on Computer Supported Cooperative Work*, pp. 175-186, ACM Press, ISBN 0897916891.
 - [51] Rish, I. (2001). An empirical study of the naive Bayes classifier. In *IJCAI Workshop on Empirical Methods in Artificial Intelligence*, pp. 41-46.

-
- [52] Rocchio, J. J. (1971). Relevance Feedback in Information Retrieval. In *The SMART Retrieval System: Experiments in Automatic Document Processing*, pp. 313-323, Prentice-Hall.
 - [53] Sarwar, B. M., Karypis, G., Konstan, J. A., Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International World Wide Web Conference (WWW10)*.
 - [54] Sebastiani, F. (2002). Machine learning in automated text categorization. In *ACM Computing Surveys*, Volume 34, Number 1, pp. 1-47.
 - [55] Shardanand, U. (1994). *Social Information Filtering for Music Recommendation*. Ph. D. Dissertation, Learning and Common Sense Group, MIT Media Laboratory .
 - [56] Shardanand, U., Maes, P. (1995). Social information filtering: Algorithms for automating 'word of mouth'. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pp. 210-217.
 - [57] Si, L., Jin, R. (2003). A Flexible Mixture Model for Collaborative Filtering. In *Proceedings of the Twentieth International Conference on Machine Learning*.
 - [58] Ten Hagen, S., van Someren, M, Hollink, V. (2003).. Exploration/exploitation in adaptive recommender systems. In *Proceedings of the third European Symposium on Intelligent Technologies, Hybrid Systems and their implementation on Smart Adaptive Systems*, pp. 189-196.
 - [59] Tudhope, D., Taylor, C., Benyon-Davies, P. (1995). Navigation via similarity in hypermedia and information retrieval. In *Proceedings HIM'95*, pp. 203-218.
 - [60] Ungar, R., Foster, D. (1998). Clustering Methods for Collaborative Filtering. In *Proceedings of the AAAI-98 Workshop on Recommender Systems*, pp. 112-125.
 - [61] Wu, H. (2002). *A reference architecture for adaptive hypermedia applications*. Ph. D. Dissertation, Technische Universiteit Eindhoven.