Recommender System for an e-learning platform

Ricardo Jorge Mendes Oliveira
Mestrado Integrado em Engenharia de Redes e Sistemas Informáticos
Departamento de Ciência de Computadores
2016

Orientador
Alípio Mário Jorge, Professor Associado,
Faculdade de Ciências da Universidade do Porto

Coorientador
José Paulo Leal, Professor Auxiliar,
Faculdade de Ciências da Universidade do Porto

Todas as correções determinadas pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,

Porto, ______/______/_________

I want to thank all of my friends for their endless and tireless support throughout all of these years. Their impact in my life changed a great deal in me.
I owe thanks to my family. Specially my mother, who has been a beacon and has given me unwavering support my whole life.
Thank you to both my supervisors, Professor Alípio Jorge and Professor Zé Paulo for allowing me to undergo this journey with them and learn from them. I also want to deeply thank them for their understanding, their humanity, their help and their seeming limitless patience.
Many Thanks, tudo é Kanimambo. (João Maria Tudella)

Abstract

Technology Enhanced Learning (TEL) provides learners all over the world a customisable, rich in content, learning experience. Because of this, TEL has been experiencing a world wide growth and usage, both from learning institutions of every level, as well as corporations that need to provide constant training to their employees with a reliable and consistent tracking system. Information retrieval is a vital (and non trivial) task in a TEL environment, given the large libraries of content currently in existence. We are also faced with the need to adapt the learning process to the progress of the student, his preferences or handicaps, as well as existing competencies, to ensure that the system is both effective and efficient, especially considering the multitude of learning objects in existence. To this end, Recommendation Systems(RSs) are one of the possible solutions. Despite RSs being widely used and very extensively researched for commercial applications, where their purpose is to increase the volume of sales of products, their usage in TEL is still rudimentary, mostly due to the fact that both the goals to fulfil and the environment are more complex. In this dissertation we describe our solution to the adaptability issue of an existing e-learning platform. Our solution consists of a RS that provides recommendations using a hybrid technique of collaborative filtering and a precedence graph (formed of weighted precedences between pairs of Learning Objects (LOs), defined by educational experts).

Resumo

Technology Enhanced Learning (TEL) fornece a alunos de todo o mundo uma experiência de aprendizagem personalizável e rica em conteúdo. Visto isto, TEL tem tido um crescimento e uma utilização a nível global, quer por parte de instituições de ensino de todos os níveis, quer por empresas que precisam de providenciar formação constante aos seus funcionários, com um sistema de acompanhamento constante e fiável. A obtenção de informação é uma tarefa vital (e não trivial) num ambiente TEL, dadas as vastas bibliotecas de conteúdo existentes atualmente. Com isto em mente, sistemas de recomendação (RSs) são uma de várias soluções possíveis. Apesar de RSs serem muito usados e extensivamente pesquisados em aplicações comerciais, cujo proposito é aumentar o volume de vendas de produtos, o uso deles em TEL é ainda rudimentar, maioritariamente devido à diferença de objetivos a atingir e à maior complexidade do ambiente de ensino. Nesta dissertação descrevemos a nossa solução para o problema de adaptabilidade de uma plataforma de e-learning existente. A nossa solução é constituída por um RS que fornece recomendações recorrendo a uma técnica híbrida composta por filtragem colaborativa e um grafo de precedências (construído por precedências com pesos entre pares de objetos de aprendizagem, pesos estes definidos por peritos na área educacional).

Contents

List of Tables

List of Figures

Listings

Acronyms

ADL	Advanced Distributed Learning
AEH	Adaptive Educational Hypermedia
ANB	Anterior Negative Border
API	Applicational Programming Interface
ASIST	Association for Science and Technology
AWS	Adaptive Web Service
BFS	Breadth First Search
CAI	Computer Aided Instruction
CB	Content-Based
CD	Compact Disk
CF	Collaborative-Filtering
CP	Cognitive Profile
DB	DataBase
DCMI	Dublin Core Metadata Initiative
EJB	Enterprise Java Bean
HTTP	Hypertext Transfer Protocol
IEEE	Institute of Electrical and Electronics Engineers
ITS	Intellingent Tutoring Systems
JAX-RS	Java API for RESTful Services
JEE	Java Platform, Enterprise Edition
JPA	Java Persistance API
JSON	JavaScript Object Notation
KNN	K Nearest Neighbours
LO	Learning Object
LOM	Learning Object Metadata
MOM	Message Oriented Mode
NB	Negative Border
ORM	Object-Relational Mapping
PM	Policy Model
PNB	Posterior Negative Border
POJO	Plain Old Java Object
REST	Representational State Transfer
ROM	Resource Oriented Mode

RS	Recommender System
SCORM	Sharable Content Object Reference Model
SOAP	Simple Object Access Protocol
SOM	Service Oriented Mode
TEL	Technology Enhanced Learning
URI	Uniform Resource Identifier
W3C	World Wide Web Consortium
WS	Web Service
WSDL	Web Service Definition Language
WWW	World Wide Web
XML	eXtensible Markup Language

Chapter 1

Introduction

Technology Enhanced Learning (TEL) is a field of increasing popularity and usage. More companies and faculties on multiple countries adopt TEL environments to facilitate learning for students or training for professionals [1]. This is due to the fact that TEL environments are very customisable and rich in content, making them usable in most scenarios. However, the opulence of content present in these environments is also one of their greatest downsides. Information retrieval is a very time consuming task, which chips away at the quality of the learning experience if not automatised. Retrieving the right type of content for the active user, considering the user's strengths, flaws, preferences and psychological profile is a complex task to perform. To this problem, a multitude of possible solutions are under research and constant development, from Intelligent Tutoring Systems (ITSs) to the implementation of Recommender Systems (RSs) in the TEL platforms. This dissertation and our solution focus on the latter.

1.1 Motivation

Our project was born from a company's need to improve their existing e-learning platform, from its current static model, to an adaptable and customisable e-learning platform. The company wanted their system to be adaptable, but have the adaptability module decoupled from the e-learning platform.

1.2 Goals

Our goal is to provide prompt, accurate and relevant recommendations to the users of the e-learning platform, taking into consideration their existing knowledge as well as their difficulties and the curricular goals to be achieved, through the use of the learning platform's content metadata and the users' usage history, as to improve the usability of the learning platform as well as the users' experiences while on it. We aim to achieve this by designing a recommendation algorithm using the platform's existing content prerequisite metadata as well as a collaborative filtering algorithm over interaction data of users with resources.

1.3 Structure

We initially cover the current state of the art of TEL, RSs, related technologies, as well as the methods and algorithms used in such systems and software architecture standards followed in Chapter 2. In Chapter 3 we detail the design choices made for our solution. Next we present the implementation of our chosen design, detailing used libraries and APIs as well as justifying our choices in Chapter 4. Then, we proceed to present the validation evidence to our solution and corresponding test results in Chapter 5. Finally, in Chapter 6 we conclude this dissertation and present possible future work.

Chapter 2

State of the art

Recommender Systems (RSs), and RSs in Technology Enhanced Learning (TEL) environments, are broadly researched topics[2]. The focus of this chapter is the current state of the art in TEL, Data Mining, RSs and Web Services (WSs), as these are the most relevant technologies to our work. In this chapter we initially present the current state of TEL and underlying technologies. Then we discuss data mining and user modelling. We continue with discussion on adaptivity, from intelligent tutoring systems to recommender systems. Lastly, we conclude this chapter with the subject of web services.

2.1 Technology Enhanced Learning

Technology Enhanced Learning (TEL) is defined as "the study and ethical practice of facilitating learning and improving performance by creating, using, and managing appropriate technological processes and resources"[3]. It encompasses both physical and theoretical tools used to enhance learning. These tools can range from multimedia content to entire systems. TEL evolved from physical media, with its size restrictions and distribution constraints, to the World Wide Web (WWW). This allowed changes to occur, such as the appearance of Collaborative Learning[4], where users help each other learn, and Virtual Learning Environments[5], where a class occurs virtually. TEL aims to *facilitate* learning through the use of technological features and learning theories. Despite the apparent simplicity of the concept, TEL is a field of extent and continuous research and debate, under constant change[3] where various scientific fields converge.

Different learning theories lead to different existing TEL environments. An ongoing shift from traditional and more formal teacher/course centred environments, such as Moodle[6], to more personalised and informal student centred environments is noticeable. This shift is supported not only by technical innovations, but by the development of new theories on epistemology, cognitive science, education theory and other areas of psychology. Linear learning environments, are still predominant [7] (Khan Academy [8] is an example). In these environments all students are subject to a single pedagogy with a specific starting point, ending point and a flow between these points, predetermined by the instructor. Linear environments remain predominant despite the change in focus to more non-linear, adaptable learning environments, where the learner is responsible for the mastery of the subject at his own pace and chooses his own flow of information, receiving at most some guidance from an instructor[7].

TEL is used both in formal and informal learning settings. A formal setting is offered by educational institutions, is well structured and leads to specific accreditation, being supervised by experts of the respective domain in order to ensure quality, who usually provide content to the users[9]. However, in an informal setting, the user is responsible for his own education in every aspect and has to sift through content found in repositories, provided by the online community[9]. This leads to usability issues for the user, being the most relevant ones to our work, the costly and non trivial content selection, and the lack of guidance/supervision.

The shift from linear to non-linear learning environments also introduces new problems:

- Content must be designed to accommodate different learning styles and levels of difficulty;
- Users with different abilities and knowledge background need to be considered;
- Platform requires context sensitivity to provide the necessary information to the user, regardless of the user's choice, allowing users to chose their own path and learn at their own pace.

In the remainder are described some of the most popular solutions to this problem, relevant technologies and definitions, as well as RSs which we use in this project.

2.1.1 Learning Object Metadata

With the evolution of TEL from physical media, such as CDs, to the WWW, centralised repositories for storing Learning Objects (LOs) appeared. These repositories developed tools to facilitate access to their own LOs by their own users, which led to issues in the interoperability of content. Different content providers would use multiple different techniques and representations, each specific to their own unique platforms. Such diversity led to little to no interoperability as well as complicated and expensive setup and maintenance of learning environments. These issues led to the standardising LOs representation and development. *Learning Object Metadata* was developed as a solution to the before mentioned issues. It is a specific type of *metadata* that focuses on the standardisation of the representation of LOs. This allows learning content interoperability between different TEL environments and facilitates LO cataloguing, selection and utilisation. Different entities have developed different standards. For instance, IEEE[10] developed LOM[11], ADL[12] developed SCORM[13] and ASSIS&T[14] is currently responsible for the DCMI[15]. All of these standards have different levels of detail and features. Despite this, some entities, such as ourselves, develop their own metadata representation of learning objects to fit their own needs[16], due to the unnecessary complexity and detail of the existing standards.

2.2 Intelligent Tutoring Systems

Intelligent Tutoring System (ITS) usually refers to "any computer program that contains some form of intelligence and can be used for learning" and has evolved from the early Computer Aided Instruction (CAI) model[17]. The aim of these systems is to work as an automated tutor, to facilitate TEL environments to provide immediate feedback and achieve individualisation. This way the system can accurately respond to each individual user's needs, such as lack of previous knowledge or a slower learning rate, without needing intervention from a human tutor. Development of ITSs relies on multiple scientific areas such as psychology, education theory and computer science. Traditionally, the ITS model is composed of:

- the domain/cognitive model;
- the student model;
- the teaching/tutoring model;
- the learning environment/user interface.

Any or all of these components may contain intelligence[18][17]. We briefly describe each of these components below.

The domain/cognitive model "contains the concepts, rules, and problem solving strategies of the domain to be learned. It can fulfil several roles: as a source of expert knowledge, a standard for evaluating the student's performance or for detecting errors, etc."[19]. This model relies heavily on epistemology since it needs to accurately formalise what represents knowledge for a particular domain. Different domains have different mechanisms and values at play. The model must be capable of providing accurate answers to students' questions, as well as solve the problems it presents to the students.

The student model usually represents student's content skills (what the student knows), knowledge about learning (how the student learns), affective characteristics (how the student is/feels) and other relevant characteristics. Some of these details, such as the affective ones, cannot be easily determined and must be inferred over the course of time through long term observation[19]. This model is used to supply data to other models and allow the ITS to adapt to the student.

The teaching/tutoring model is responsible for the interaction and adaptability, which differentiate the act of *tutoring* from lecturing, of the ITS. It analyses information from the domain and student models and adapts to the student in question, through the usage of tutoring strategies. The model is also responsible for providing the student with feedback, be it upon the student's request during problem solving or when the system detects deviations from the path by the student.

The learning environment/user interface "integrates three types of information that are needed in carrying out a dialogue: knowledge about patterns of interpretation (to understand a speaker) and action (to generate utterances) within dialogues; domain knowledge needed for communicating content; and knowledge needed for communicating intent"[18].

2.3 Data Mining

DM techniques are typically used in RSs as part of other techniques and methodologies. The data mining process is divided in three iterations, in the following order: *Data Preprocessing*, *Data Analysis* and *Result Interpretation*. Data preprocessing consists of preprocessing data into a computer processable format to be used in *Data Analysis*. In the data anlysis iteration, the preprocessed data is subject to techniques (such as classification) where relevant patterns are sought. Finally, result interpretation is the extraction of relevant and useful information from the previously analysed data.

2.4 User modelling

User modelling consists of building a computational representation of a user, according to features found relevant. This representation is then used by algorithms that produce user specific responses. User models can be produced implicitly through observation of user interactions with the system and analysis and storage of the relevant data derived from those observations, or explicitly through the filling of questionnaires or preferences set by the users[20]. They may also be of either static (initially formed and not subject to changes) or dynamic (constantly updated with information) design. They can also be mix of both having static user model with information such as date of birth and name, and a dynamic user model containing system interaction information[21]. Which data is relevant is heavily dependant on the goal of the system. For instance, an adaptive system usually requires information from interface navigation and functionality usage from the user, while a RS may rely on information about user preferences. For this same reason, there is no universal template for a user model, and migration and interoperability of models between systems is a major challenge.

2.5 Adaptive Educational Hypermedia

Adaptive Educational Hypermedia (AEH) is a type of Adaptive Web System (AWS) and, therefore, a top-down approach. AWSs aim to solve the problems inherent to the use of a single solution in a Web Service (WS), by investigating possible ways in which to adapt the WS's characteristics in function of the different characteristics of the WS's different users, through user modelling. A system is considered adaptive if "it is able to change its own characteristics automatically according to the user's needs"[22]. More specifically "an interactive system that adapts its behaviour to individual users on the basis of processes of user model acquisition and application that involve some form of learning, inference, or decision making."[23]. An adaptive system aims at improving the users' experience through a series of functions. Firstly, adaptive systems provide users with support in system usage, support which can be given in multiple different ways, from taking over part of the users' time consuming routine tasks to adapting the system's interface to better fit the users' models[23]. More relevant to our work are the functions of supporting information acquisition and obtaining information about users.

Supporting information acquisition reduces the impact that information overload has on the users by helping the users find the desired information in a helpful manner. One way of achieving this is by filtering content that fits the user model and providing it to the user. This is applicable to web browsing, query-based search and even spontaneous provision of information[23]. Another means to support information acquisition is by supporting learning. The system can adapt to each user in the learning content, problems and tests that are presented and how they are presented[23]. Obtaining information about users is strongly connected with user modelling . AEHs tackle these same issues in a more restrict setting, specifically in educational applications. Most of AEH systems are used in a formal learning setting[24][25]. In these settings there are predefined curricula, accreditation procedures and student/teacher profiles, dependencies between learning resources, all very well defined and structured through the use of metadata. This provides more useful information for the adaptive technologies as well as recommendation systems. In informal learning settings much of this information is lacking and there is no formal accreditation, which makes the learning process itself much more open ended at the expense of requiring additional effort in information gathering to allow the use of AEH. Being a top down approach, AEHs require plenty design work in preparation and during maintenance of the TEL environment, since knowledge domains having to be described in detail.

2.6 Recommender Systems

RSs are a subclass of information filtering systems with the purpose of successfully predicting the standing of a user towards an item.

These systems assist users by selecting potentially interesting choices within a vast set of alternatives. The initial focus of RSs was to deal with the overload of information, initially through the use of collaborative, content-based and demographic based techniques[26], being later researched and implemented for commercial purposes.

The principle here is that a user might buy an item that is recommended to him, that he would not buy nor actively seek otherwise.

In commercial domains, the most widespread use of RSs, is to ensure more sales of a certain product or more consumption from a certain user. For RSs in TEL, the goal is to provide learning objects and fulfil an array of tasks to improve the user's learning experience, taking in consideration many factors and particularities that are exclusive to TEL[9][27].

These particularities may range from the type of learning that is desired to the user's prior knowledge and even the complex nature of learning. For instance, RSs in TEL should take context (such as the learner's competencies and the time investment required for an object) into consideration as to provide the user with content in the best degree of detail, or abstraction, according to different learner settings, to ease learning[28]. Another factor to account for is the important role that recapitulation and reiteration take in the learning process and how that differentiates RSs in TEL in comparison to RSs in a commercial environment. Another particularity of RSs in TEL stems from the cold start problem and the way it has to be handled. Unlike products or movies that a user can rate early to build a profile, either because he has seen/used them or has a positive impression of them from reading/hearing about them, the same cannot happen in a learning context, where it is improbable that a user has had contact with certain learning objects before[27]. RSs in this setting are also very platform specific, and consequently it is very hard to migrate one RS from one learning platform to another without having to undergo major adjustments. In short, a RS in TEL has more complex goals that need to meet the users' pedagogical needs as well as their characteristics and more subjective information to consider. RSs generally rely on techniques that can be divided in three categories, which we describe below.

2.6.1 Content-based recommendation techniques

Content-based (CB) recommendation techniques produce recommendations of items similar to ones that the users have liked in the past. A profile of user interests is built based on previous ratings the user gave to items, and that profile is then used to recommend new items that match the characteristics desired by the user, which can be very effective if the profile is accurately built[9]. For this reason, item representation is a very important factor in content-based RSs. Items are represented by *attributes* or *proprieties*[9]. These can be described through the same set of attributes with a known set of values or through textual features extracted from Web pages, news, product descriptions and more[9]. In the first case, the use of machine learning algorithms to improve recommendations is possible since attributes are well defined making machine interpretation straightforward. In the later, though we are faced with a greater challenge, since textual features are not well defined and are subject to the ambiguous nature of language. Traditional keyword based profiles work mainly by string matching, which do not allow for semantic comprehension of the users' interests[9]. This poses a problem because one same word can have multiple meanings in different contexts (polysemy) and multiple words can have one same meaning (synonymy). These phenomena lead to the possibility of a profile containing different words for the same characteristic or different characteristics characterised by one word, making string matching ineffective and insufficient. One of the most popular technique to deal with this problem is *semantic analysis*, which is done by adopting knowledge bases for both annotating items and representing profiles to obtain a semantic aware representation of users' needs[9].

A content-based RS is composed by three separate components, each with different functions[9]:

Content analyser analyses the content of objects (items) through feature extraction techniques, in order to appropriately extract structured relevant information to feed as input to the other two components;

Profile learner aims to construct the user's profile by using generalisation strategies to infer a model of user interests from items the user (dis)liked;

Filtering component matches the user's profile representation to the items to be recommended, obtaining a relevance judgement (may be binary or continuous) which determines if an item should be recommended or hidden.

Using content-based techniques have great advantages. For instance, these techniques are user independent. This means that only the ratings given by the user are accounted for when building its profile and producing recommendations. Another beneficial characteristic of these systems resides in recommendations being given based on the characteristics of the items that match the users profiles and new items can promptly be recommended without having to be previously rated by users.

On the downside, content-based techniques suffer from the limitation of available domain knowledge regarding items, since not every aspect that can make an item of interest to a user may be described or defined, therefore making recommendations less accurate than desirable. These techniques also suffer from the over-specialisation problem, which regards the lack of ability of the system to recommend something that resides outside of the user's profile, since only items that match the user's interests will be recommended. New users also pose a challenge to this approach, since a number of ratings must be recorded before recommendations become accurate towards the user's interests.

2.6.2 Collaborative-filtering recommendation techniques

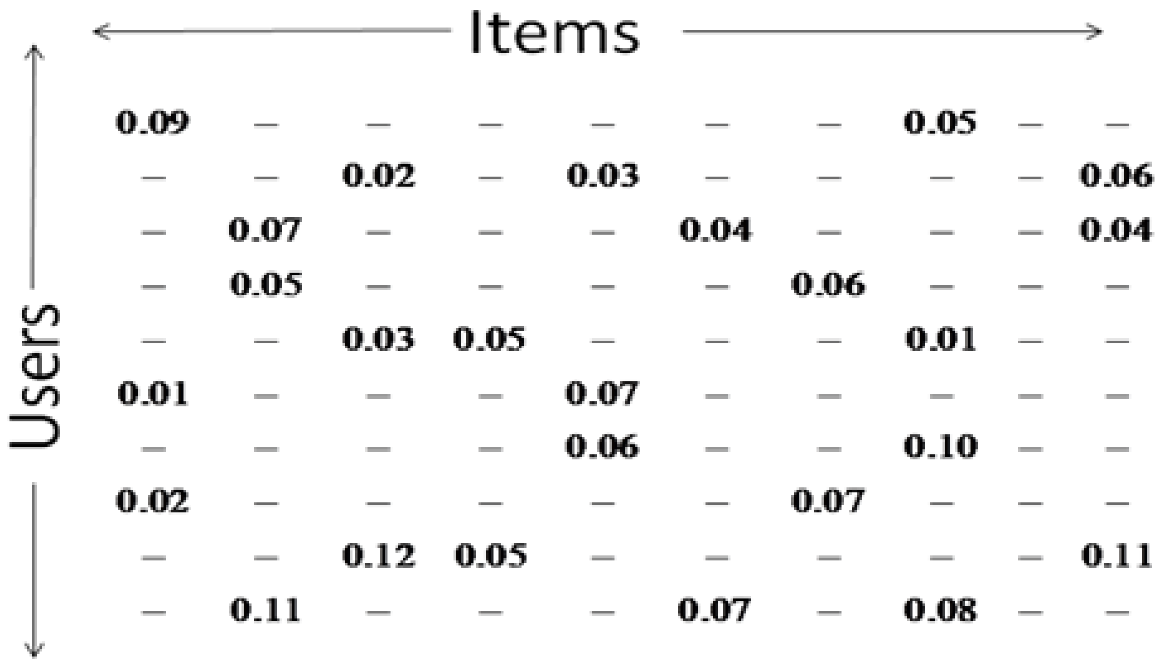
Collaborative-filtering (CF) recommendation techniques produce recommendations based on the user ratings matrix. Early recommender systems relied on this matrix directly to compute a neighbourhood. With the evolution in this field, recommendation started being treated as a classification problem, and most techniques now use the user ratings matrix indirectly to induce a collaborative model. Based on the tradition of people resorting to friends, family or even coworkers with similar taste to obtain recommendations on what movies to watch, content to view, places to eat, these techniques assume that if different *user*'s *rate* items similarly, then they will probably rate other items similarly as well[29]. Generally, these *ratings* on *items* a user has used/bought are considered his history. These ratings may be:

Unary depicts an interaction with an item such has having bought or seen said item, which may be simply observed by the system;

Binary one of two possible values, may be the user (dis)liking an item or have added said item to a wish/watch or exclusion list;

Discrete a value from an array of possible values, may be a number of stars between zero to five stars or a numeric value between zero and ten, either integer or real.

The collection of these values from every user are compiled into a matrix, called ratings matrix in which each entry corresponds to the rating a certain user gave to a certain item, if there is any as seen in Figure 2.1. Early CF recommender systems were distance based. These systems used the ratings matrix and calculate a user's neighbourhood through the use of similarity functions[29]. Possible recommendations are produced by filling the ratings for the items that the user has no ratings for, based on its neighbourhood and then listing the N best *scored* items, where score can be influenced by other variables besides ratings. From its start in neighbourhood/distance based algorithms, CF has evolved. We now have CF algorithms categorised as either memory or model based. Memory based algorithms requires that all ratings, users and items are stored in memory at all times. Model based algorithms periodically create a summary of ratings patterns offline. Since memory based algorithms suffer from scalability issues, most systems currently are either purely model based or have some form of pre-computation. Because of this, a much better organisation is organising algorithms as probabilistic or non-probabilistic. Probabilistic algorithms, as the name indicates, are based on some form of probabilistic model while non-probabilistic algorithms


Items
Users

0.09	–	–	–	–	–	–	0.05	–	–
–	–	0.02	–	0.03	–	–	–	–	0.06
–	0.07	–	–	–	0.04	–	–	–	0.04
–	0.05	–	–	–	–	0.06	–	–	–
–	–	0.03	0.05	–	–	–	0.01	–	–
0.01	–	–	–	0.07	–	–	–	–	–
–	–	–	–	0.06	–	–	0.10	–	–
0.02	–	–	–	–	–	0.07	–	–	–
–	–	0.12	0.05	–	–	–	–	–	0.11
–	0.11	–	–	–	0.07	–	0.08	–	–

Figure 2.1: CF Ratings Matrix

are not. In this section we'll present the more classic, non-probabilistic neighbourhood based CF algorithms.

2.6.2.1 User-based collaborative filtering

We begin with user-based CF, or user-user CF or neighbourhood-based CF, where we resort to ratings users give on items. The assumption is that users that rated the same items similarly have similar preferences and are classified as neighbours[9]. It recommends unseen items to a user that have been positively rated by his neighbours. This approach requires the use of a similarity function to compute the neighbourhood. It suffers from the cold-start problem, which occurs when: not enough users have rated enough items; when a new user has not rated enough items to have his neighbours computed; and even when a item is new or has not been rated enough times. This approach also has the problem of recommending only popular content, which may not be the best for every student, leaving other quality items out of the pool. Scalability issues are another concern. A large user base causes increases in data, which in turn leads to difficult or inaccurate computation of neighbourhoods. This is particularly problematic since similarities between users can very quickly shift with new ratings or changes of previous ones and are therefore computed when recommendations are requested. These issues lead to the next algorithm.

2.6.2.2 Item-based collaborative filtering

In item-based CF, also item-item CF, we focus on item rating patterns. The principle followed is that the items that a set of users rate similarly are similar, and recommendations are produced taking into consideration that similar users tend to have similar preferences[9]. Much like user-based CF, this approach requires the use of a similarity function to compute the neighbourhood. Therefore, it is susceptible to the cold-start problem when an item is new and hasn't received enough ratings and recommends only the most popular content. The advantage of this approach is the scalability. Although it is still required to find the closest items to produce recommendations, pre-computing of the ratings matrix becomes possible in systems with a high user to item ratio, since in such systems a user chantging or adding a new rating to an item should not be enough to change similarity between items with any significance. This characteristic allows for the pre-computation of similarities in an item-item similarity matrix. Despite the similarity matrix not being always up to date, staleness should have no major impact in the quality of obtained recommendations as seen before.

2.6.3 Hybrid recommendation techniques

Hybrid recommendation techniques involve, as the name suggests, the joining of multiple techniques. This is done to produce recommendations with better performance, while overcoming some of the weaknesses of the individual techniques. These techniques are a common approach to dealing with cold start or scalability problems. There are multiple categories of hybrid recommendation techniques, some of which we describe below.

2.6.3.1 Weighted

This technique uses the scoring of all the different recommendation techniques within the system to produce a recommendations (for instance, a linear combination of all the different techniques' recommendation scores). The benefit here is that all the techniques are used in a fairly simple way and adjustments to the hybrid system are easy to perform. The downfall lies on the fact that this technique is based on the principle that the different techniques do not differ much in value and usefulness across multiple scenarios, which we know not to be true[30].

2.6.3.2 Switching

In this case the technique used is switched on the fulfilment of certain criteria. For instance, a recommender system that provides content based recommendations initially but switches to collaborative filtering recommendations if the content based ones do not have the desired degree of certainty. Switching hybrid recommendation techniques allow to take great advantages of the strengths of each of the underlying techniques, while overcoming their weaknesses, with the right parameters defined (which is their greatest fault, the need for additional parametrisation)[30].

2.6.3.3 Mixed

In a mixed strategy recommendations from multiple different techniques are presented simultaneously. This can be particularly useful if a large volume of recommendations is needed[30].

2.6.3.4 Cascade

The cascade hybridisation method implies the sequential use of techniques. The first technique produces a set of ranked candidates to be refined by a second technique[30].

2.7 Web Services

The W3C[31] defined a web service (WS) as " a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the WS in a manner prescribed by its description using Simple Object Access Protocol (SOAP) messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards"[32]. This definition was later changed[33] to accommodate Roy Fielding's proposal of the Representational State Transfer (REST) architectural style[34]. Web services are designed to standardise communication between distributed systems, through the use of well defined technologies, where each participating side may be on a different platform or system, with different specifications. To better define a desired web service there are four different architectural models for web services: the Message Oriented Model (MOM); the Service Oriented Model (SOM); the Resource Oriented Model (ROM); and the Policy Model (PM)[32]. Each model has different

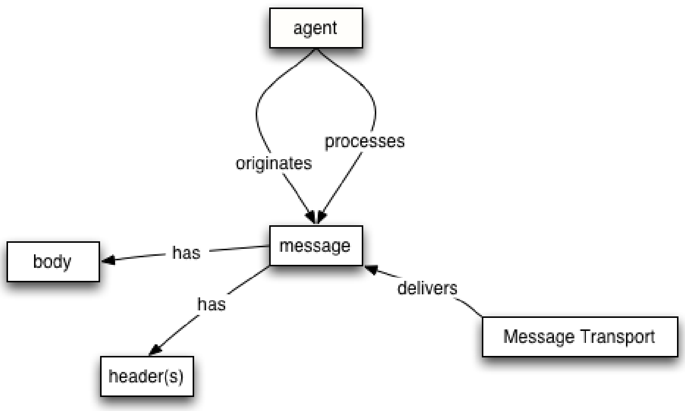


Figure 2.2: Message Oriented Model

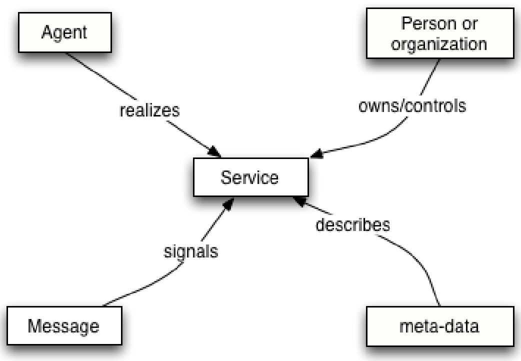


Figure 2.3: Service Oriented Model

focuses on different parts of the web services, as to adequately detail every aspect of the web service. The Message Oriented Model has messages and their processing as its focus, from protocol used to message body standardisation, as seen in Figure 2.2. To the Service Oriented Model pertain the aspects of the services provided by the agent to their agent and the requests made, as shown in Figure 2.3. SOM is an increment on the MOM. The Resource Oriented Model explicits resources and their properties as well as their interactions with everything relevant to the service, as depicted on Figure 2.4. Finally, the Policy Model portrays the policy related aspects of the service, from security to quality of service, viewable on Figure 2.5. In this section, we focus on presenting some of the Simple Object Access Protocol (SOAP) and the Representational State Transfer (REST) guidelines.

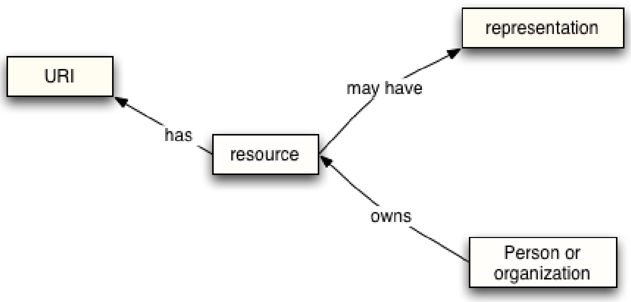


Figure 2.4: Resource Oriented Model

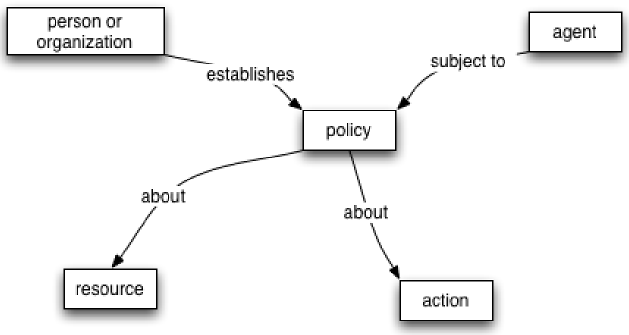


Figure 2.5: Policy Model

2.7.1 Architectural style - Simple Object Access Protocol

Simple Object Access Protocol (SOAP), according to the W3C[31], is "a lightweight protocol intended for exchanging structured information in a decentralised, distributed environment. It uses XML technologies to define an extensible messaging framework providing a message construct that can be exchanged over a variety of underlying protocols. The framework has been designed to be independent of any particular programming model and other implementation specific semantics."[35]. It operates through the exchange of messages from an initial SOAP sender - the originator of the SOAP message at the starting point of a SOAP message path - to an ultimate SOAP receiver through any number of SOAP intermediaries. SOAP senders, SOAP receivers and SOAP intermediaries are all SOAP nodes, which behave differently according to the SOAP processing model. Each SOAP node is identified by an URI.

2.7.2 Architectural style - Representational State Transfer

The standard for Representational State Transfer (REST) states that a service should be stateless and operate through the use of HTTP messages and verbs (GET,PUT,POST,DELETE) from Client to Server. Each verb has very well defined functionality and purpose, as defined for HTTP 1.1 [34]. Every message contains a header and body. The header contains basic information describing the message, such as the HTTP verb used, as well as having specific fields that can describe the message body, the format used, if any multimedia is attached, etc. The body may contain textual information, in specific languages or models (as signaled in the header) to be sent to the receiving party. The HTTP verbs and their behaviour are as follows:

GET should not trigger any changes on the Server. It only serves the purpose of obtaining data from the Server, as is;

PUT is a destructive writing operation. It can be used to create a new entry or overwrite a previously existing one;

POST functionality is to create new entries, as well as to perform incremental changes and/or insertions to already existing entries;

DELETE is used specifically to signal the Server which data should be removed.

Since there's no state synchronisation, each transaction is processed as a unit, independent of every other one. Any service that fully complies to these guidelines is called RESTful.

2.8 Chapter Summary

In this chapter we presented a study of technologies related to TEL and adaptability in TEL systems to better understand how to tackle the problem we were tasked with solving. We discussed several technologies, of which RSs and WSs are the most relevant ones for our work, and the ones we will use in the remainder of this dissertation.

Chapter 3

Design

The existing e-learning platform was a linear, static learning environment. Learning objects (LOs) were catalogued as rigid sequences within a topic, leaving no room for adaptability. When users entered the platform, they were given the choice to be taught a given subject or to be evaluated on what they had previously learned. If the user chose to be taught, he was presented with a static content structure to chose from. Despite the topic he chose to start from, he would always be presented with a static sequence of LOs. The sequence began with expository (contain information in video, audio or text format about the subject to be taught) LOs and it would end with evaluative (a quiz to evaluate what the user has apprehended about the subject at hand) LOs. When a sequence was completed the user could chose to follow the content's predetermined order or jump to another unit by manually navigating the menus. Users could freely backtrack within the sequence. If the user chose to be evaluated, he is subject to a set of quizzes chosen by the system at random and given a grade at the end.

The company chose to improve their e-learning platform by changing the content cataloguing schema and introducing adaptability features. The changes to the cataloguing schema allowed to re-use LOs as well as to introduce adaptability into the system. These cataloguing schema changes were handled by a team of e-learning specialists. To potentiate these changes, we were tasked with adding adaptability features to the e-learning platform. The updated e-learning platform was being developed simultaneously with our solution. The company wanted to add adaptability to their platform, but maintain it functional even without the adaptability features. To this end, they provided us with information on how and when the system should be adaptive to the user, without any knowledge of the inner working of their platform. These constraints

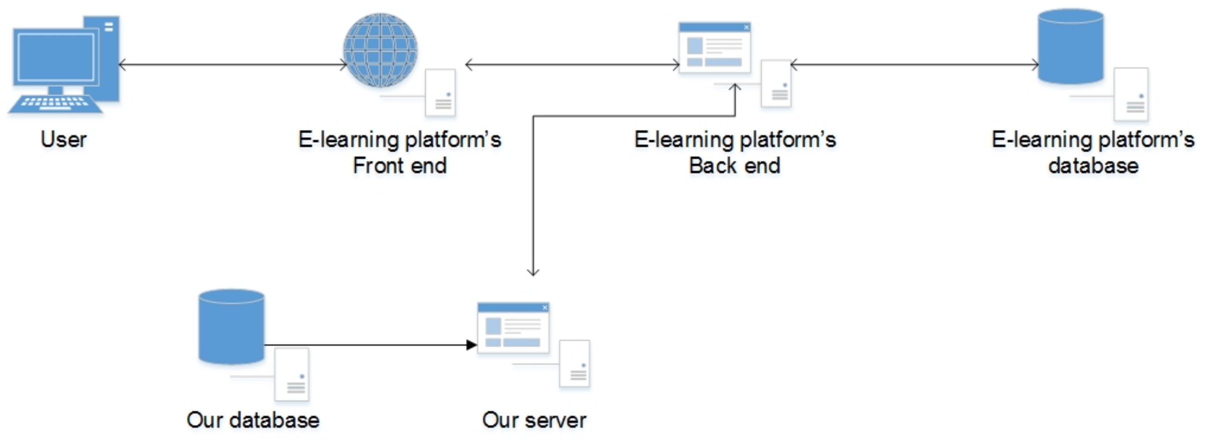

User
E-learning platform's Front end
E-learning platform's Back end
E-learning platform's database
Our database
Our server

Figure 3.1: Service Interaction Scheme

led us to design our recommender system (RS) a RESTful web service (WS). The proposed architecture follows the design showed in Figure 3.1.

In this chapter we present the design goals and the actual design of the proposed solution. The recommender system is required to work as a standalone platform, separate from the e-learning platform. It will be able to receive data and recommendation requests from the e-learning platform. The data received is vital to build the recommendation models. The data received must be stored and its manipulation should be as simple as possible, in order to keep it updated, as it is necessary in order to provide accurate recommendations. The recommender system will be able to consider users' knowledge background and precedence relationships within the learning content to process recommendations.

To this end we began by designing an abstract data model, to simplify the excessively detailed information of the e-learning platform. This helps us focus on information that's pertinent to build a recommendation model and simplify the communication process. We then designed a RESTful API to facilitate data transaction and manipulation from the e-learning platform our service. The API also handles recommendation requests, taking into consideration the previously designed data model. Next we tackled the recommendation strategy, designing user profiles models and strategies to effectively use the learning content's precedence relationships, as well as algorithms to generate recommendations.

We describe our design choices in further detail in the following sections.

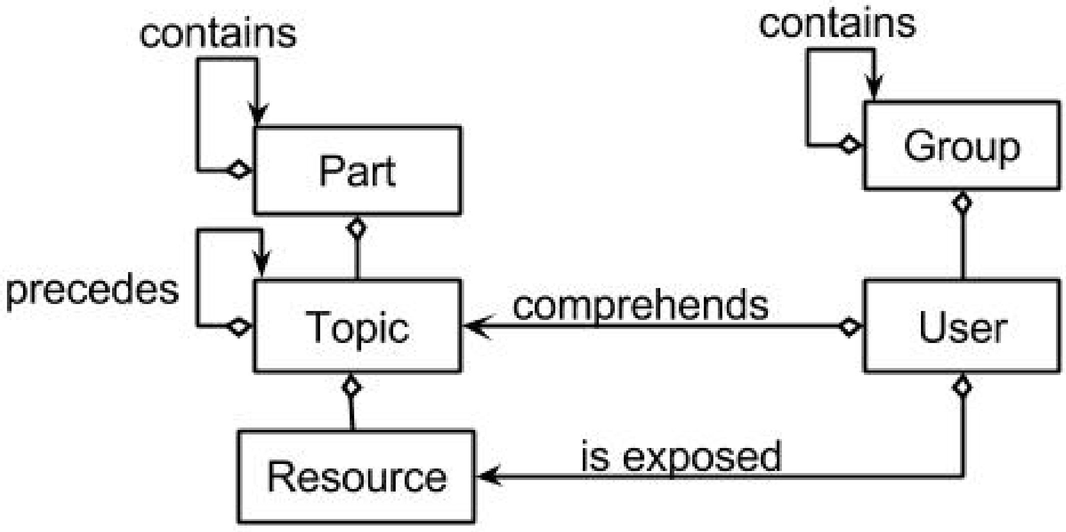


Figure 3.2: Users HTTP Request

3.1 Abstract Data Model

We designed an abstract data model to simplify the excessively (for a RS) detailed data from the e-learning platform and to allow data transactions from it to our recommendation service through our API. This model aims to encapsulate the vital metadata of the learning platform's content and users, disregarding irrelevant information for our recommendation service. The content is organised in a hierarchy composed of three different levels: *parts*, *topics* and *resources* as shown in Figure 3.2. Parts are the highest of the hierarchy and contain other parts and topics. Different parts may contain one same topic. Topics are collections of resources. Resources belonging to one topic cannot belong to another topic. Precedence relations can be established between topics and its weight is represented by a real value named *relevance*. The value of the relevance variable is directly proportional to the importance of the topic in order to understand the topic it precedes. This precedence relationship can exist between topics contained by different parts. Resources are the atomic element in the hierarchy, and each correspond to a learning object. These resources may be of an expository type, exposing users to a subject in any form of multimedia, or of an evaluative type, in order to determine the mastery of a user over a subject.

3.2 Recommendation strategy

Our system aims to produce recommendations for the following cases:

- Recommend a prerequisite topic to a student in order to address his flaws;
- Recommend a subsequent topic to a student who is ahead and seeking to learn new subjects;
- Recommend a list of sorted topics that compose a part.

With the mentioned use cases in mind, we consider two different recommendation scenarios:

- Metadata based recommendation, where we:
  - Analyse user metadata (his position on the curricular plan);
  - Analyse content metadata (the precedence relationships in the content);
  - Compute recommendations from this information.
- Usage history based recommendation, where we:
  - Analyse evaluation results;
  - Analyse content observation history;
  - Compute the user's neighbourhood;
  - Recommend items that similar users have been successful with.

Regardless of scenario, we reduce every recommendation to a topic recommendation. The assumption is that every recommendation is correlated to determining which are the relevant topics to recommend to a student in a given moment. Throughout the recommendation process, we assume that topics have a predefined partial order, which is established by the learning platform. We also have the assumption that the data about the student's knowledge is potentially incomplete. We tackle this issue in subsection 3.2.1.

3.2.1 Cognitive Profile

We define the user's Cognitive Profile (CP) as the set of topics he has comprehended. This profile is built either by implicit observation of the user's interactions with the system, through

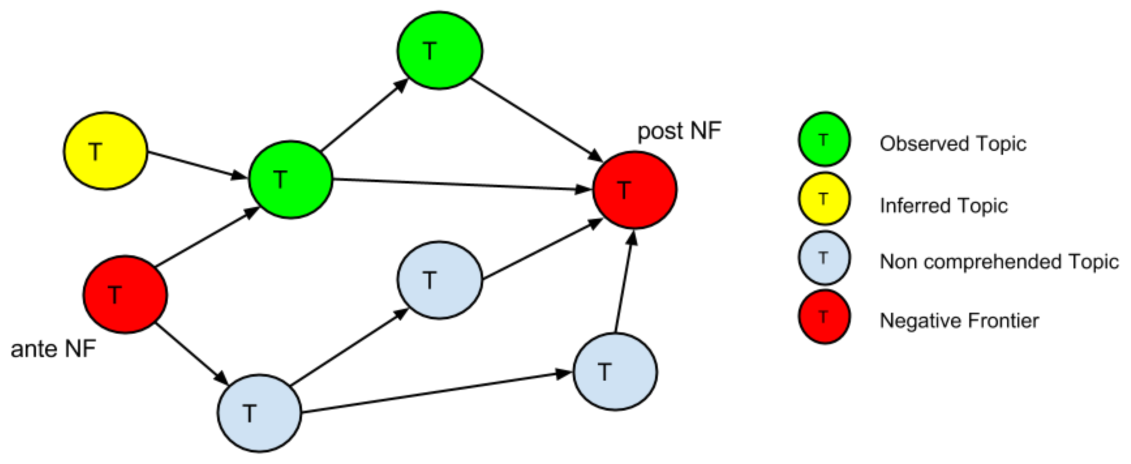


Figure 3.3: Topic Graph

exposure to or evaluation of topics, or inference over topics the user might have comprehended, despite the lack of observation of the interaction. A formal definition is as follows:

$$CP_{student} = \{T \mid topic\ T\ has\ been\ learned\ successfully\ by\ student\}$$

3.2.2 Negative Border

We define the Negative Border (NB) as the set of topics immediately preceding (*Anterior* Negative Border - ANB) or succeeding (*Posterior* Negative Border - PNB) a given set of topics. Applying this concept to the user's cognitive profile allows us to determine topics that the user should revise to overcome his difficulties or topics he can advance into. We define the anterior and posterior negative borders as follows:

$$ANB_{student} = \{T' \mid topic\ T'\ precedes\ topic\ T\ in\ CP\ and\ T'\ not\ in\ CP\}$$

$$PNB_{student} = \{T' \mid topic\ T\ precedes\ topic\ T'\ in\ CP\ and\ T'\ not\ in\ CP\}$$

3.2.3 Graph Based Recommendation Algorithms

These algorithms produce recommendations using information from the above described cognitive profiles and precedence relations. Through the precedence graph, we have the ability to produce recommendations that fulfil specific pedagogical goals, either introducing new content to allow the student to progress or recommend content which the student should already know and might improve his comprehension of the subject. If we add the information from the user's cognitive profile we can infer what the user may already know, therefore avoiding recommending topics the student has comprehended. We now proceed to describe the three algorithms that produce recommendations purely based on the precedence graph and cognitive profile information.

Algorithm 1 is direct at the first use case and produces recommendations of topics which precede a given position in a cognitive profile. It starts by expanding the directly preceding neighbours of the topics in the *StartList*, denominated *Preceding.* At each recursion of the algorithm, for each topic p in *Preceding* we add the pair (p,relevance) to the list of results (*ResultList*) and add p to the auxiliary list *NextList*, that will be the starting list on the next recursion. The relevance mapped in the pair (p , relevance) corresponds to the minimum between the relevance of the currently observed precedence relationship and the relevance of the relationship of the topic seen in the last iteration. After all the topics in *Preceding* have been processed the algorithm takes a recursive step with *NextList* as the starting list argument. The result of the recursive call is appended to *ResultList* When then maximum number of recursions is reached, *ResultList* is scored and then the scored results are returned.

Algorithm 2 produces recommendations of topics which succeed a given position in a cognitive profile and aims to satisfy the second use case. To achieve this, the neighbours directly succeeding the topics in the *StartList*, named *Succeeding*, are expanded. In every recursion of the algorithm, for every topic s in *Succeeding* we add the pair (s , relevance) to the list of results (*ResultList*) and add s to the auxiliary list *NextList*, which is the starting list for the next recursion step. Upon having processed all the topics in *Succeeding*, the algorithm recurses on *NextList* as the starting list argument. The result of the recursive call is appended to *ResultList*. Similarly, *ResultList* is scored and the scored results are returned after all the recursions have been completed.

With algorithm 3 we tackle the third use case. It recommends Topics within a Part, sorted by the order of the precedence relations. The recommendations consist of the topics contained in that part, sorted by their order in the precedence graph. This fulfils the purpose of obtaining the starting point(s) for the user when starting a session on a new part. Unlike the cases above, the *StartList* here is composed of the topics that are contained within a part that is identified on the input, much like a SCORM Organization [36]. It starts by finding every topic in the part and their precedence relations with each other. Topics that don't belong in the part are not accounted

Algorithm 1 Obtain Preceding Topics

```
Input:
$RT$ return type,
$Context$: recommendation context,
$OID$: object ID,
$OT$: object type,
$UID$: user ID,
$Size$: num. of return objects

procedure GenerateAnte($Niter$,$StartList$)
    $Niter \leftarrow Size\ /\ 2$
    $ResultList \leftarrow$ empty list
    if $Niter == 0$ then
        return $ResultList$
    $NextList \leftarrow$ empty list
    for each Topic entity $t$ in $StartList$ do
        for each Precedence $p$ in $t$ do
            if $p$ is not validated then
                Append $p$ to $ResultList$ and $NextList$
    Append $GenerateAnte$(Niter−1,NextList) to $ResultList$
    $ResultList \leftarrow Scoring(ResultList$,Size)
    Return: $ResultList$
```

for. Then, for every topic found by the algorithm a Node containing it is created and mapped. The algorithm also maps the topics' preceding and succeeding relations with other topics into two maps, *MapPreceding* and *MapSucceeding* respectively. A tree is built with all the mapped nodes and their respective relations. Starting from the tree's toots (nodes without precedences within the collection) the nodes are visited, breadth first, and coloured to avoid repeated visits. The *Size* first visited nodes that haven not been validated by the user are returned.

3.2.3.1 Scoring

Scoring occurs as the final step of the algorithms. For scoring, every occurrence of a Topic is accounted for. For instance, if Topic A precedes both Topic B and Topic C, then the relevance

Algorithm 2 Obtain Succeeding Topics

```
Input:
$RT$ return type,
$Context$: recommendation context,
$OID$: object ID, $OT$: object type,
$UID$: user ID,
$Size$: num. of return objects

procedure GeneratePost($Niter$,$StartList$)
    $Niter \leftarrow Size$ / 2
    $ResultList \leftarrow$ empty list
    if $Niter == 0$ then
        return $ResultList$
    $NextList \leftarrow$ empty list
    for each Topic entity $t$ in $StartList$ do
        for each Succeeding $p$ in $t$ do
            if $p$ is not validated then
                Append $p$ to $ResultList$
            else
                Append $p$ to $NextList$
    Append $GeneratePost$(Niter−1,NextList) to $ResultList$
    $ResultList \leftarrow Scoring(ResultList$,Size)
    Return: $ResultList$
```

in both precedences is considered. For Topics that are not a direct precedence to those in the StartingList, the score is equal to the sum of the minimum value between the Topic's weight as a precedence, for a Topic X, and the weight of the precedence of the Topic it precedes, for a Topic Y, where X precedes Y.

For every occurrence of T', the weight of the occurrence is calculated as shown in Equation 3.2.3.1.

$$WEIGHT_{topic} = \{min(relevance(T', T), WEIGHT_T)) \mid topic\ T'\ precedes\ topic\ T\}$$

Algorithm 3 Obtain Sorted Topics

```
Input:
$RT$ return type,
$Context$: recommendation context,
$OID$: object ID,
$OT$: object type,
$UID$: user ID,
$Size$: num. of return objects
procedure BuildGraph($OID$)
    $TopicSet \leftarrow GetChildren$(OID)
    $PrecedingMap \leftarrow$ empty precedence map
    $SucceedingMap \leftarrow$ empty precedence map
    $NodeMap \leftarrow$ empty node map
    for each Topic entity $t$ in $TopicSet$ do
        $BeforeList \leftarrow$ empty precedence list
        $AfterList \leftarrow$ empty precedence list
        $TempNode \leftarrow CreateNode$(t)
        $PrecedingList \leftarrow GetPreceding$(t)
        $SucceedingList \leftarrow GetSucceding$(t)
        for each Preceding $p$ in $t$ do
            $TempTopic \leftarrow GetTopic$(p)
            if $TopicSet$ contains $TempTopic$ then
                Append $TempTopic$ to $BeforeList$
        for each Succeeding $s$ in $t$ do4
            $TempTopic \leftarrow GetTopic$(s)
            if $TopicSet$ contains $TempTopic$ then
                Append $TempTopic$ to $AfterList$
        $PrecedingMap$gets <$t$,$BeforeList$>
        $SucceedingMap$gets <$t$,$AfterList$>
        $NodeMap$gets <$t$,$TempNode$>
    $Tree \leftarrow LinkNodes(NodeMap)$
    $Roots \leftarrow$ nodes without precedences in $Tree$
    $ReturnList \leftarrow$ first $SizenonvalidatedNodesobtainedfrom$BFS($Roots$)
    Return: $ResultList$
```

$$SCORE_{topic} = \{sum(WEIGHT_{topic}) \mid for\ every\ occurrence\ of\ topic\ found\}$$

3.2.3.2 Node Tree

Each Node is composed of the Topic it refers to in the graph, a list of its preceding and a list of its succeeding Nodes and a boolean value that signals if the Node has been visited during the Breadth First Search (BFS). A Tree is composed of a list of its root Nodes - or nodes without precedences.

3.2.4 Collaborative Filtering Recommendation Algorithm

Our choice for the collaborative filtering algorithm was the cosine similarity measure. We measure the distance between two users with the given formula:

$$\text{sim}(u, v) = \cos(u, v) = \frac{|CP_u \cap CP_v|}{\sqrt{|CP_u| \times |CP_v|}}$$

After calculating all the similarity between every pair of users, we can build their neighbourhoods, which consist of the K most similar users, and use said neighbourhoods to score recommendations of Topics for the user in question with:

$$Score_{uT'} = \frac{\sum_{v \in K_u} \text{sim}(u, v) I_{vT'}}{\sum_{v \in K_u} \text{sim}(u, v)}$$

, where $I_{vT'}$ is an indicator function, valued 1 if the user has comprehended T' ($T' \in CP_v$) or 0 otherwise.

3.2.5 Hybrid scoring

We use a cascade based strategy, where the results of the CF algorithm are matched to the results of the graph based algorithms. The ones with the highest scores that are present in both sets of recommended topics are presented.

3.3 API

We chose to use a RESTful design in the API, since it is only used to perform requests in which data is transitioned or recommendations are obtained. Data can be updated through an HTTP message with an URI indicating the resource(s) to be changed, the HTTP verb that determines how to change and JSON on the HTTP message body describing the structures to be updated, if necessary. Each transaction is unique and independent of others and has to be fully completed, being rolled back otherwise. We created URLs for different types of data, as well as different sized sets of data and defined the fields of each entity as seen in the examples displayed in Tables 3.1 and 3.2 for the Users entity.

3.4 Chapter Summary

In this chapter we presented the design choices for the components in our solution. The solution is a RS designed as a RESTful WS. An abstract data model was designed to be used in communication between the e-learning platform and our service. Our solution is comprised of a RESTful API, recommendation model builders and recommendation algorithms.

Table 3.1: Users HTTP Request

Method	HTTP Request	Description
put	put /users/[id]	creates or rewrites the data belonging to the user identified by the ID in the URL
post	post /users[id]	updates or increments the data belonging to the user identified by the ID in the URL
post	post /users/set	inserts the user(s) data as described in the message body
delete	delete /users/[id]	eliminates all the data belonging to the user identified by the ID in the URL

Table 3.2: Users HTTP Body

Name	Type	Description
id	String	User's ID
year	Integer	User's school year
group	String	ID of the Group the user is in

Chapter 4

Implementation

This chapter describes the implementation of our service, which is comprised of an Application Programming Interface (API), Object-Relational Mapping (ORM), recommendation engine and relational database (DB). In Figure 4.1 we show how these components interact with each other. Each of the parts was built and integrated in a modular fashion, taking in consideration the possibility future work, additions and tweaks to the service, as well as adaptation to other platforms. Despite all of the components working with each other, they remain independent of each other and are usable in other settings. The entirety of the service was developed using JEE and deployed on a JBoss Application Server (AS) 7.1 servlet container. We chose JEE due to the standards and APIs it provides out of the box. Every request goes through a Controller class, which in turn invokes a Resource class (Stateless Enterprise Java Bean (EJB)) and finally the requests are fulfilled through the use of Entity classes to query and manipulate data. Below, we describe in further detail the implementation details of the different components.

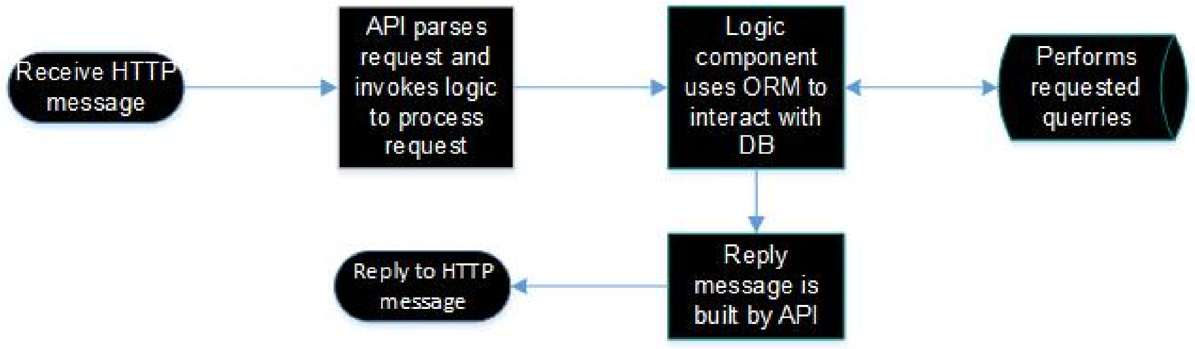

Receive HTTP message
API parses request and invokes logic to process request
Logic component uses ORM to interact with DB
Performs requested querries
Reply message is built by API
Reply to HTTP message

Figure 4.1: Implementation Scheme

4.1 API

The API is the interface through which the Client (the company's service) can request recommendations and changes to the Server's (our service) database in order to maintain data updated for the recommendations to be accurate. It is a RESTful Web Service developed with the use of the JAX-RS API (the Java API for RESTful Web Services)[37]. Through the use of the JAX-RS API we are allowed to use Plain Old Java Objects (POJOs) as RESTful Web resources. Every request processing begins with URI matching and message body processing, and finally Stateless EJBs are invoked to deal with the logic of the request. We describe all these steps in further detail below.

4.1.1 URI matching

When a request is received by the server, URI matching is made by the JAX-RS API through path annotated classes. Path annotated classes are Java POJOs with declarative annotations. The path annotation defines the URI that the class corresponds to. Besides defining the URI path that the class should respond to, it is possible to define what type of data should be expected in the HTTP message. The classe's methods can be annotated with the path annotation as well, allowing for partially different URLs to invoke different methods. To respond to the HTTP requests for a determined URI, the classe's methods are annotated with the HTTP request. All of these details are showed in Listing 4.1.

We defined different classes for the different data that the client could send to our server and one to process recommendation requests. All of these classes are path annotated, making use of the JAX-RS API. Each of the classes' methods are annotated to respond to specific HTTP requests and sub paths from the classes' main paths. All of the classes receive the client-side information in the HTTP body messages in JSON, which is automatically parsed and built as a POJO through the use of the Java API for JSON Processing [38], so we defined classes for the different types of expected data to be received. Having the URI matched and the message body turned into the respective POJOs, the respective method from the corresponding Stateless EJB[39] is invoked.

Listing 4.1: Path Annotated Class

```
package pt.sea.service.controllers;
```

```java
import pt.sea.service.objects.UsersObject;
import pt.sea.service.resources.UsersRS;

import javax.inject.Inject;
import javax.ws.rs.*;

@Path("users")
public class UsersController {

    @Inject
    private UsersRS usersRS;

    @Path("{id}")
    @PUT
    @Consumes("application/json")
    public void putUser(@PathParam("id") String id, UsersObject usersObject)
    {
        usersRS.put(usersObject);
    }
}
```

4.1.2 Stateless EJBs

Stateless EJBs are one type of Session EJBs which's lifecycle is tied to the duration of the invoked method, making them ideal to implement the business logic in a RESTful Web Service. Upon invocation, they implement the logic tied to the specific HTTP request that triggered their invocation and perform the adequate consultations or alterations to the server-side database through the use of the Entity Manager transactions.

4.2 ORM

To aid the implementation of the business logic we use ORM. More specifically, we use the Java Persistance API (JPA) [40]. JPA facilitates a POJO persistence model for ORM, through the use of Entity EJBs. Entities are a type of EJB which are managed by the Entity Manager[41]. Through the use of the JPA, DB manipulation is simplified, since every transaction and its integrity (and necessary rollbacks) are managed by the persistence framework.

4.2.1 Entity EJBs

An entity EJB is a lightweight persistance domain object. Each entity EJB generally represents a database table, and each instance of an entity EJB corresponds to a row in said table.

4.3 Recommendation Engine

The recommendation engine consists of a negative border generator, a collaborative filtering recommendation engine, graph based recommendation engine and a data model trainer.

4.3.1 Negative border generator

The negative border generator searches for all the topics the user has apprehended, the Cognitive Profile(CP), and seeks all preceding and/or succeeding topics of the CP that the user has not apprehended. This last set of topics forms the negative border, which is used in the graph based recommendation engine.

4.3.2 Graph based recommendation engine

The graph based recommendation engine implements all of the three algorithms described in subsection 3.2.3. Depending on the context of the recommendation request, one of the three implementations is invoked. In case of a request of a preceding or succeeding recommendation, the negative border generator is used and an implementation of 1 or 2 is called. In the case of a request to obtain a list of topics contained by a part, a implementation of 3 is invoked, which seeks the root topics within the part and creates a graph by expanding the topics preceded by them in a breadth first manner.

4.3.3 Collaborative filtering recommendation engine

The collaborative filtering recommendation, which was designed and implemented by a member of our team, is an implementation of the K Nearest Neighbours classifier with the cosine similarity measure described in 3.2.4. It is composed of a data model trainer, which uses the e-learning platform's usage data to create the neighbourhoods for each user, and a user KNN recommendation engine. We persist the data model in memory, eliminating the need for constant disk access, therefore increasing the speed of the recommendation process, through the use of a Singleton EJB[39].

4.4 Chapter Summary

We described the implementation of our solution. We used JEE and related technologies to deploy a RESTful API. Data transactions are managed by JPA. All the recommendation algorithms are implemented in Java. The recommendation model is persisted through the use of EJBs.

Chapter 5

Validation

Our recommender system (RS) was subject to two types of tests. Offline validation tests, and validation and usability tests. The offline validation tests aimed to validate the precision of the recommendations. Precision is very important but not sufficient to measure the success of a recommender system within a learning platform. With this in mind validation and usability tests were conducted by specialists in the validation of e-learning tools, who were part of the team. In these tests the precision of the recommendations as well as the efficacy, efficiency and satisfaction of the users while using the e-learning platform in conjunction with our system were tested. The efficacy corresponds to the capability of the users when it comes to completing tasks using the system, as well as their quality in doing so. Efficiency is related to the amount of resources consumed during the tasks performed. The satisfaction metric is based on the subjective reactions of the users while using the system. In total four metrics were analysed, but only three are relevant for our work. These are:

Precision if the recommendations obtained match the expected recommendations;

Efficacy how many topics the user validates during the session, and how high his mark while doing so;

Efficiency how many resources the user consumed during the session.

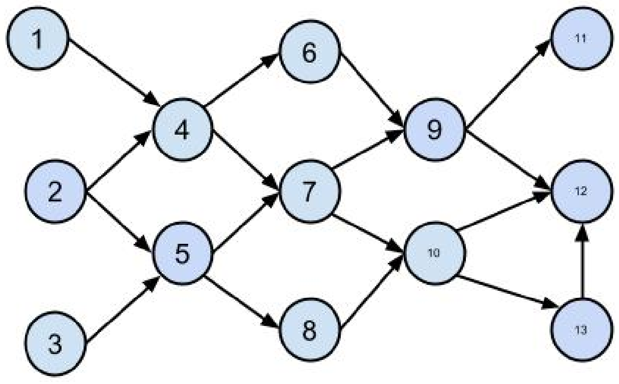


Figure 5.1: Artificial Data Set

5.1 Offline tests

Offline validation tests were performed during development and integration of the recommender system with the company's platform development and content provider teams to ensure the RS's correct operation and precision. We also conducted our own validation tests. We created a set of artificial CPs from an artificial set of topics and precedences shown in Figure 5.1. In this set, all the topics have the same weight in every relation they are a part of. This simplifies human comprehension of the results. The nodes that are visited the most in the recursion steps are the ones with the highest score.

Based on this data set we requested recommendations of 3 items, for different users with different cognitive profiles. For each different user, we requested either:

a preceding topics based on the user's CP;

b succeeding topics based on the user's CP;

c topics preceding topic 9, considering the user's CP;

d topics succeeding topic 9, considering the user's CP.

Requests *a* and *c* use Algorithm 1 while requests *b* and *d* use Algorithm 2.

The test results confirmed that all the recommendations obtained were as expected, for every scenario tested. The results are shown in Table 5.1. The return for request *a* is always empty because the student has no observed yet unvalidated topic in his history.

Table 5.1: Offline Test Results

Case	Topics in CP	Returned recommendation
1	[4,6,7,9]	a: [] b: [12,11,10] c: [] d: [11,12]
2	[4,6,9]	a:[] b: [12,11,7] c: [7,5] d: [11,12]
3	[4,5,9]	a:[] b: [7,9,8] c: [7] d: [11,12]
4	[4,6,7,10]	a: [] b: [9,12,13] c: [] d: [12,11]
5	[7]	a: [] b: [10,9] c: [6,4] d: [12,11]
6	[4,6,10]	a: [] b: [] c: [4,6,7] d: []
7	[11,12,13]	a: [] b: [] c: [4,6,7] d: []
8	[2]	a: [] b: [5,4] c: [4,6,7] d: [11,12]

Figure 5.2: Usability Tests

5.2 Usability Tests

The usability tests were conducted in real time with a group of young pre-university students interacting with the system, as seen in Figure 5.2.

The usability tests were done by having multiple users using the e-learning platform in either a learning mode or a testing mode. In the learning mode, the users were presented with a set of expository learning objects pertaining a topic. At the end of this set they were tested on what they apprehended during the session. In the testing mode, the users were subjected to a set of quizzes in order to validate their existing knowledge. Despite the mode the user was subject to, efficacy and efficiency where measured. Efficacy corresponds to the number of tasks a student successfully completed (number of Topics comprehended). In other words, it's the amount of topics the user validates during the session. Efficiency is the amount of recommended resources the user consume during the test.

The usability tests revealed a visible correlation between the efficacy and efficiency of the students, which can be seen in Table 5.2. We represent the best fit regression curve between efficiency and efficacy in Figure 5.3, where we can see that the trend suggests that the increase in success is related to the number of recommendations followed by the student.

Table 5.2: Users' efficacy and efficiency

User	Mode	Efficacy	Efficiency
S3	T	2	2
S4	T	1	2
S5	L	2	4
S6	L	3	4
S9	L	3	4
S10	T	1	2
S11	T	3	2
S12	L	4	8

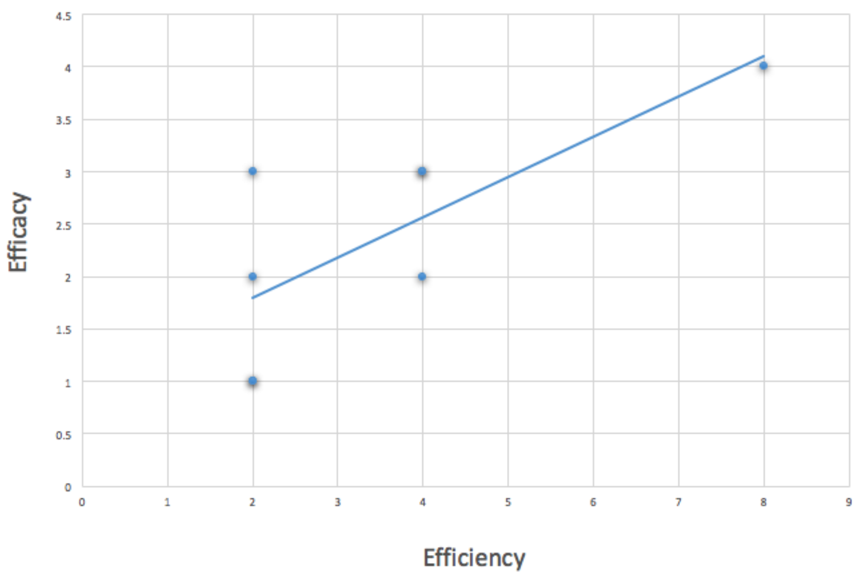


Figure 5.3: Linear Regression Graph

5.3 Chapter Summary

We described the test environments and how the tests were conducted. From the results of the tests we were able to validate the correct functioning of the recommender system and its positive impact in the e-learning platform.

Chapter 6

Conclusion

The purpose of our work was the development of a recommender system for an e-learning platform, that would improve the platform's usability and adaptability, and the user's experience along with it. Our finished work meets all the specifications we detailed and achieves the goals we set.

We started by investigating the state of the art of the technologies related to the scope of our project. From ITS to RSs in TEL, during this investigation we were able to realise that adaptivity in TEL is a very discussed and ample subject, with multiple possible solutions available.

We designed several aspects of the recommendation solution, such as the recommendation algorithms, the API and the abstract data model.

The solution we proposed uses JEE and underlying technologies such as JPA and EJBs along with a MySQL DB for data storage. We implemented our hybrid recommender system as a RESTful web service. Our DB and every transaction with it is managed by ORM, which ensures the security and integrity of the transactions. Our solution is completely modular so that future updates are easy to perform, previous features are easy to tweak and new features easy to integrate.

The validation to our solution showed that the recommender system was working as expected. They also revealed a visible correlation between usage of recommended resources and success in the validation of topics.

6.1 Future work

A possible improvement in terms of time performance would be to not use ORM. Using a driver, code all the queries and the process of handling every transaction with the database reduces DB query time. But it is a very time costly solution, since it requires the programming of every query and fallback, while ORM eases DB operation and deals with transactions with minor intervention necessary. The recommendation engine can certainly be improved, but we need more tests with a large user base and data sets to discern what those may be, which will only be available when the platform's published.

Bibliography

[1] eLearning Industry. The Top eLearning Statistics and Facts For 2015 You Need To Know - eLearning Industry. https://elearningindustry.com/elearning-statistics-and-facts-for-2015. Visited on 2016-09-15.

[2] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowl. and Data Eng.*, 17(6):734–749, June 2005.

[3] Rhonda Robinson, Michael Molenda, and Landra Rezabek. Facilitating learning. *Educational technology: A definition with commentary*, pages 15–48, 2008.

[4] Gerry Stahl, Timothy Koschmann, and Dan Suthers. Computer-supported collaborative learning: An historical perspective. *Cambridge handbook of the learning sciences*, 2006:409–426, 2006.

[5] Martin Weller. *Virtual learning environments: Using, choosing and developing your VLE*. Routledge, 2007.

[6] Moodle. Moodle. https://moodle.org/, 2016. Visited on 2016-06-01.

[7] Ronald Robberecht. Interactive nonlinear learning environments. *The Electronic Journal of e-Learning*, 5(1):59–68, 2007.

[8] Khan Academy. Khan academy. https://www.khanacademy.org/, 2016. Visited on 2016-06-01.

[9] Nava Tintarev and Judith Masthoff. *Recommender Systems Handbook*, volume 54. 2011.

[10] IEEE-SA. Ieee-sa - the ieee standards association. http://standards.ieee.org/index.html. Visited on 2016-06-02.

[11] IEEE-SA. Ieee sa - 1484.12.1-2002 - ieee standard for learning object metadata. https://standards.ieee.org/findstds/standard/1484.12.1-2002.html. Visited on 2016-06-02.

[12] ADL. Adl net – advanced distributed learning. https://www.adlnet.gov/. Visited on 2016-06-02.

[13] ADL. Scorm – adl net. https://www.adlnet.gov/adl-research/scorm/. Visited on 2016-06-02.

[14] ASSIST. Asist | the information association for the information aget. https://www.asist.org/. Visited on 2016-06-02.

[15] ASSIST. Dcmi home: Dublin core metadata initiative (dcmi). http://dublincore.org/. Visited on 2016-06-02.

[16] Keith Harman. *Learning objects: standards, metadata, repositories, and LCMS*. Informing Science, 2007.

[17] Reva Freedman, Syed S Ali, and Susan McRoy. Links: what is an intelligent tutoring system? *intelligence*, 11(3):15–16, 2000.

[18] Indira Padayachee. Intelligent tutoring systems: Architecture and characteristics. In *Proceedings of the 32nd Annual SACLA Conference.(Cité p. 2), August*, 2002.

[19] Roger Nkambou, Riichiro Mizoguchi, and Jacqueline Bourdeau. *Advances in intelligent tutoring systems*, volume 308. Springer Science & Business Media, 2010.

[20] Addie Johnson and Niels Taatgen. User modeling. *The handbook of human factors in web design*, pages 424–438, 2005.

[21] Jatinder Hothi and Wendy Hall. An evaluation of adapted hypermedia techniques using static user modelling. In *Proceedings of the second workshop on adaptive hypertext and hypermedia*, pages 45–50, 1998.

[22] Reinhard Oppermann. Adaptively supported adaptability. *International Journal of Human-Computer Studies*, 40(3):455–472, 1994.

[23] Anthony Jameson. Adaptive interfaces and agents. *Human-Computer Interaction: Design Issues, Solutions, and Applications*, 105:105–130, 2009.

[24] Milos Kravcik, Marcus Specht, and Reinhard Oppermann. Evaluation of winds authoring environment. In *International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, pages 166–175. Springer, 2004.

[25] LM Aroyo, Riichiro Mizoguchi, and CK Tzolov. Ontoaims: ontological approach to courseware authoring. 2004.

[26] Michael J Pazzani. A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review*, 13(5-6):393–408, 1999.

[27] Hendrik Drachsler, Hans GK Hummel, and Rob Koper. Personal recommender systems for learners in lifelong learning networks: the requirements, techniques and model. *International Journal of Learning Technology*, 3(4):404–423, 2008.

[28] Katrien Verbert, Nikos Manouselis, Xavier Ochoa, Martin Wolpers, Hendrik Drachsler, Ivana Bosnic, and Erik Duval. Context-aware recommender systems for learning: a survey and future challenges. *IEEE Transactions on Learning Technologies*, 5(4):318–335, 2012.

[29] Michael D Ekstrand, John T Riedl, and Joseph A Konstan. Collaborative filtering recommender systems. *Foundations and Trends in Human-Computer Interaction*, 4(2):81–173, 2011.

[30] Robin Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, November 2002.

[31] W3C. World Wide Web Consortium (W3C). http://www.w3.org/. Visited on 2015-11-15.

[32] W3C. Web Services Architecture. http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/. Visited on 2015-11-15.

[33] W3C. Relationship to the World Wide Web and REST Architectures. http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/#relwwwrest. Visited on 2015-11-15.

[34] Roy Thomas Fielding. *Architectural styles and the design of network-based software architectures*. PhD thesis, University of California, Irvine, 2000.

[35] W3C. SOAP Version 1.2 Part 1: Messaging Framework (Second Edition). http://www.w3.org/TR/soap12-part1/. Visited on 2015-11-15.

[36] Rustici Software. SCORM Content Packaging. http://scorm.com/scorm-explained/technical-scorm/content-packaging/. Visited on 2015-10-10.

[37] Oracle Corporation. Java API for RESTful Services (JAX-RS). https://jax-rs-spec.java.net/. Visited on 2015-10-29.

[38] Oracle Corporation. The Java Community Process(SM) Program - JSRs: Java Specification Requests - detail JSR# 353. https://jcp.org/en/jsr/detail?id=353. Visited on 2015-10-29.

[39] Oracle Corporation. Entities - The Java EE 7 Tutorial. https://docs.oracle.com/javaee/7/tutorial/ejb-intro002.htm#GIPJG. Visited on 2015-10-29.

[40] Oracle Corporation. Java Persistence API. http://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html. Visited on 2015-10-29.

[41] Oracle Corporation. 37.3 Managing Entities - Java Platform, Enterprise Edition: The Java EE Tutorial (Release 7). https://docs.oracle.com/javaee/7/tutorial/persistence-intro003.htm. Visited on 2015-10-29.