BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INFORMATION SYSTEMS
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

EDUCATIONAL SYSTEM FOR RECOMMENDING STUDY ACTIVITIES

VZDĚLÁVACÍ SYSTÉM PRO DOPORUČOVÁNÍ STUDIJNÍCH AKTIVIT

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR **JAKUB ZAPLETAL**
AUTOR PRÁCE

SUPERVISOR **Ing. RADEK BURGET, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2018

Bachelor's Thesis Specification/20677/2017/xzaple36

Brno University of Technology - Faculty of Information Technology

Department of Information Systems | Academic year 2017/2018

Bachelor's Thesis Specification

For: **Zapletal Jakub**
Branch of study: Information Technology
Title: **Educational System for Recommending Study Activities**
Category: Information Systems

Instructions for project work:

1. Learn the basic principles and techniques of recommending systems. Focus on educational systems, in particular on recommending study materials.
2. Study different existing metrics of student's activity. Study the existing solutions.
3. Analyze and evaluate possible ways of including additional information about the study activities, for example text analysis, evaluation of educational experiments, etc.
4. Design and implement a recommending system or a module for an existing system that uses the chosen information based on the above analysis.
5. Evaluate the implemented methods on real-world data.
6. Summarize the obtained results.

Basic references:

- Gutmans, A., Rethans, D., Bakken, S.: Mistrovství v PHP 5, Computer Press, 2012
- Žára, O.: JavaScript - Programátorské techniky a webové technologie, Computer Press, 2015

Requirements for the first semester:
Items 1 to 4

Detailed formal specifications can be found at http://www.fit.vutbr.cz/info/szz/

The Bachelor's Thesis must define its purpose, describe a current state of the art, introduce the theoretical and technical background relevant to the problems solved, and specify what parts have been used from earlier projects or have been taken over from other sources.

Each student will hand-in printed as well as electronic versions of the technical report, an electronic version of the complete program documentation, program source files, and a functional hardware prototype sample if desired. The information in electronic form will be stored on a standard non-rewritable medium (CD-R, DVD-R, etc.) in formats common at the FIT. In order to allow regular handling, the medium will be securely attached to the printed report.

Supervisor: **Burget Radek, Ing., Ph.D.**, DIFS FIT BUT
Beginning of work: November 1, 2017
Date of delivery: May 16, 2018

L.S.

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

Dušan Kolář
Associate Professor and Head of Department

Abstract

The purpose of this work is to design and implement a module for an already existing recommender system at the Open University, Milton Keynes. The existing system uses information about user activity in the Virtual Learning Environment (VLE) gathered from previous years and uses it to recommend relevant study activities for students. This module uses semantical similarity of study materials to recommend those which help user to complete an assignment or to find materials similar to ones provided. Similarities between documents are computed using Term Frequency - Inverse Document Frequency and word embedding methods. RESTful API was devised to communicate with the OU Analyse interface.

Abstrakt

Cílem této práce je navrhnout a implementovat modul do existujícího doporučovacího systému Open University v Milton Keynes. Nyní nasazený doporučovací systém využívá informací o aktivitě uživatelů ve Virtual Learning Environment (VLE) nasbíraných z předchozích let a podle ní doporučuje studentům relevantní studijní aktivity. Tento modul využívá sémantické podobnosti mezi studijními materiály k doporučení těch, které pomohou uživateli vyřešit úkol nebo které jsou podobné k těm, o něž projevil zájem. K počítání podobnosti dokumentů je využíváno metod Term Frequency - Inverse Document Frequency a vnoření slov. Pro používání modulu a jeho komunikaci modulu s rozhraním OU Analyse je implementováno RESTful API.

Keywords

Recommender systems, Technology enhanced learning, Information systems, Educational systems, Document similarity

Klíčová slova

Doporučovací systémy, Vzdělávací systémy, Informační systémy, Technologií posilované učení, Podobnost dokumentů

Reference

ZAPLETAL, Jakub. *Educational System for Recommending Study Activities*. Brno, 2018. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Radek Burget, Ph.D.

Rozšířený abstrakt

Cílem této práce je vytvořit modul do již existujícího doporučovacího systému na Open University Milton Keynes, nazvaného Open University Analyse Recommender (OUAR) [17] nebo vytvořit úplně nový doporučovací systém. OUAR využívá sledování aktivity studentů monitorováním toho, na co, kolikrát a kdy klikají v tamním univerzitním informačním systému Virtual Learning Environment (VLE). Autorem navrhovaný modul by mohl tento systém rozšířit tak, že uvede nové metriky aktivity studentů nebo přinese jiný způsob doporučování relevantních aktivit studentům. Součástí této práce je seznámení s oblastí doporučovacích systému a průzkum nejnovějších přístupů k doporučovacím systémům v oblasti vzdělávání. Dále jsou zkoumány různé metriky aktivity studentů ve VLE a případné způsoby jak jich využít k vylepšení doporučování.

Open University (OU) nabízí množství kurzů z různých oblastí, které jsou vyučovány čistě dálkově. Jednotlivé běhy kurzu jsou zde nazývány prezentace. Studijní cyklus na OU vypadá tak, že každý kurz je rozdělen na studijní bloky, které jsou dále děleny na části. Tyto části jsou probírány jednotlivých studijních týdnech. V rámci jednoho týdne může být probíráno více částí bloku nebo i blok celý. Toto se liší kurz od kurzu. Ve VLE má každá část bloku přidělené studijní materiály, které by si měl student v daném týdnu nastudovat. Tyto materiály jsou nejčastěji studijní texty, jejichž součástí mohou být i videa a jsou přímo dostupné z VLE. Studenti jsou v rámci kurzu rozděleni do několika skupin, které má na starost tutor - vyučující, který je dané skupině k dispozici a hodnotí veškeré práce. Student je v kurzu hodnocen na základě výsledků Tutor Marked Assignments (TMA). Tyto TMA jsou tutorem hodnocené domácí úlohy, obvykle rozdělené na několik podúkolů. Ty jsou obvykle ve VLE reprezentovány jako studijní materiály v tydnu, který je TMA přiřazen.

Systém OUAR funguje tak, že každému studijnímu materiálu kurzu přiřazuje hodnotu *relevance*. Tato hodnota je vypočítána z počtu kliknutí úspěšných studentů na daný materiál. Tato hodnota se počítá pro každý týden zvlášť. Další hodnotou, se kterou pracuje OUAR, je *effort*. Tato hodnota vyjadřuje, jak moc daný student s konkrétním materiálem pracoval. Doporučování pak probíhá tak, že se vypočítá rozdíl mezi *relevancí* materiálu v daném týdnu a *effort* studenta k tomuto materiálů. Výsledkem je důležitost dokumentu pro studenta v daném týdnu. Tento výpočet se provede pro všechny materiály a studentovi jsou doporučeny materiály, které mají v daném týdnu nejvyšší důležitost.

Pro účely rozšíření OUAR byly brány v potaz tyto další metriky: *čas*, který student stráví prohlížením materiálu a *délka* materiálu. Problémem při používání kliků k výpočtu *relevance* materiálu je fakt, že delší dokumenty mají množství kliků obvykle vyšší než dokumenty kratší. Toto je způsobeno tím, že student nepřečte dlouhý úsek textu najednou a vrací se k němu aby jej dočetl. Toto přináší do hodnoty relevance chybu, jelikož tento delší dokument nemusí být nutně důležitější než dokument, který je kratší. Pokud by do již existujícího výpočtu *relevance* a *effort* byla zanesena informace o *času* a o *délce* tohoto materiálu, tato chyba by mohla být minimalizována.

Používání aktivity studentů s sebou nese nepříjemný problém toho, že nelze vytvářet doporučení pro nové kurzy, jelikož nejsou k dispozici žádná historická data o aktivitě. Nakonec bylo rozhodnuto vytvořit místo modul nový doporučovací systém, který bude doporučovat na základě materiálů samotných. Jedná se o systém využívající sémantické podobnosti dokumentů k doporučování. Tento přístup byl zvolen, jelikož je relativně neprobádán a řeší problém nového kurzu.

Modul byl ve výsleku navržen a implementován tak, že k doporučování využívá podobnosti studijních materiálů ve vektorovém prostoru. Tento systém je tedy omezen pouze na doporučování studijních textových materiálů. Účelem je studentům doporučit studijní

materiály, které se týkají zadání specifických TMA. Tyto TMA se obvykle skládají z jednotlivých otázek, které jsou ve VLE dostupné jako studijní materiály a mohou tedy být reprezentovány ve stejném prostoru jako ostatní studijní materiály. Myšlenkou je, že materiály, které jsou těmto otázkám nejblíže ve vektorovém prostoru jsou ty, které je pomohou vypracovat.

Dokument je převeden do vektorové podoby metodou Term Frequency - Inverse Document Frequency (TF-IDF) nebo metodou vnoření slov. Před vektorizací je dokument nutné zredukovat na seznam slov v základním tvaru (tokenů) a odstranit z něj tzv. stop slova jako například předložky, spojky a podobně.

Metoda TF-IDF vypočítává frekvenci výskytu každého tokenu dokumentu a jeho převrácenou hodnotu frekvence výskytu ve všech dokumentech korpusu. Dokument je poté reprezentován jako vektor těchto TF-IDF hodnot, přičemž počet dimenzí vektoru je roven počtu unikátních tokenů v rámci celého korpusu.

Metoda vnoření slov je založena na princpu reprezentace jednotlivých tokenů jako mnohorozměrných vektorů reálných čísel, které vyjadřují význam tokenu vzhledem k jeho vztahu k jiným tokenům. Tyto vektory obvykle mají význam pouze v kontextu s vektory jiných tokenů a lze pomocí nich zjistit podobnost libovolných dvou tokenů ze stejného vektorového prostoru. Způsobů převedení tokenů na vektory je mnoho a jsou obvykle realizovány pomocí strojového učení. Pro potřeby doporučovacího systému bylo využito předtrénovaných vektorových modelů, přesněji model Stanford GloVe Wikipedia a Stanford GloVe Common Crawl [14] a model ConceptNet Numberbatch 5 [16]. Vektor dokumentu byl získán tak, že byla vypočítána střední hodnota vektorů všech tokenů z dokumentu.

Podobnost dvou dokumentů byla počítána jako kosínová vzdálenost jejich vektorů. Tato podobnost je vypočítáná mezi všemi materiály kurzu. Z podobností je sestavena matice, v níž sloupce a řádky reprezentují jednotlivé studijní materiály a hodnoty v buňkách obsahují reálné číslo reprezentující podobnost dvou materiálů. Tyto matice jsou pro kurzy předpočítány před samotným doporučováním. Pro získání doporučení je třeba specifikovat materiály pro něž se mají najít nejpodobnější materiály kurzu. Dotazem může být například otázka z TMA, k níž by uživatel chtěl najít podobné materiály z kurzu.

Tento nový systém je nasazen do již existujícího prostředí OU Analyse, které funguje jako rozhraní k oběma doporuovacím systémům pro uživatele. Komunikace je řešena pomocí RESTful API, které umožňuje vyžádat doporučení pro libovolnou otázku TMA kurzu, pro nějž byla provedena veškerá příprava a vypočítány podobnostní matice. Pro vyžádání přípravy kurzů a předpočítání matic je k dispozci rozhraní na bázi fronty zpráv, které přijímá požadavky na zpracování kurzu a provede je při nejbližší volné příležitosti.

Evaluace byla provedena pouze offline s použitím metrik *precision*, *recall* a vlastní metriky *rank*. Bylo při ní využito kurzu, jehož TMA obsahují tutory doporučené materiály k vypracování otázek, které byly pro potřeby těchto metrik označeny za dokumenty relevantní k otázce. Evaluace ukázala, že používání podobnosti dokumentů k doporučování studijních materiálů je nadějné a pokud budou implementované metody dostatečně zdokonaleny, může být tento systém úspěšně používán. Metody zjišťování podobnosti dokumentů, které v evaluaci uspěly nejvíce jsou TF-IDF a vnoření slov. Vnoření slov ukázalo nadějné výsledky, jelikož systém s touto metodou doporučoval i pokročilé nepovinné materiály, které se týkaly tématiky otázek. Toto může být problematické v případě doporučování pro TMA, ale užitečné pro doporučování studijních materiálů podobných těm, o které student projevil zájem. Evaluace ukázala, že zvolené přístupy vedou k uspokojivým výsledkům, a proto byl systém nasazen do systému OU Analyse pro další testování.

Educational System for Recommending Study Activities

Declaration

Hereby I declare that this bachelor's thesis was prepared as an original author's work under the supervision of Ing. Radek Burget Ph.D. The supplementary information was provided by prof. Zdeněk Zdráhal, Ing. Martin Hlosta, Phd., Ing. Michal Huptych, Phd. and Ing. Jakub Kočvara of Open University Analyse. All the relevant information sources, which were used during preparation of this thesis, are properly cited and included in the list of references.

.....
Jakub Zapletal
August 14, 2018

Acknowledgements

I would like to thank my supervisor Ing. Radek Burget, Phd. for supervising my thesis. Also I would like to thank people of Open University Analyse for helping me with integrating my work into the existing system and providing insight into the work already done.

Contents

Chapter 1

Introduction

In the modern day, the amount of available information on the internet can be overwhelming. With the rise of online services, focused on providing resource access to users, be it media, news, information or merchandise, the problem of information overload has surfaced. The need for personalization in the online world has resulted in the increasing demand for recommendation systems which aim to alleviate this problem. These systems should allow service providers to cater to each individual user or group of users and their needs, resulting in benefit to both the provider and the user. This is done by filtering content according to the user's needs, tastes, relevance or affordability. The systems try to predict the users preferences as accurately as possible in a variety of ways. The possible methods include analyzing the user's previous behavior or profiling the users, grouping them by their preferences or common attributes and then basing the recommendations on each others behavior. More often than not more ways of predicting preferences are used at once.

Recommender systems are used in variety of applications, examples being Youtube's video recommendations [5], Facebook's friend recommendations or e-shop product recommendations. Recommender systems are seeing more and more use, as their usefulness can be immense. This thesis will focus on recommender systems in education. Purpose of this type of recommender system is to help students find the best possible approaches to their studies and most relevant study materials to focus on to achieve their study goals. Struggling students can often find themselves in a situation, where they have been unable to direct their full attention to studies, be it because of health problems or other unexpected reasons. This puts them at serious disadvantage, mainly in distance-learning courses. If they were unable to study for a longer period of time, they can find themselves overwhelmed by the amount of available study materials and may not be able to, or have time to, find out the ones most relevant to passing the course. The educational recommender systems are supposed to help with this problem. They should be able to find the most relevant study materials or activities to help the struggling student in catching up with the course.

This work focuses on developing a module for an existing recommender system OU Analyse Recommender (OUAR) at Open University, Milton Keynes [17]. OUAR uses collaborative filtering in the form of collecting clicks of students in the university Virtual Learning Environment (VLE) to compute relevancy of study materials and effort of students themselves to recommend relevant study materials. The module adds content based recommending functionality to allow recommendation for courses which are new and don't have any previous student activity logged. This is done using document similarity computation methods, specifically Term Frequency - Inverse Document Frequency and word embedding vector space model. This allows students to find materials closest to for exam-

ple assignment question or a material that interests them. In the end, the resulting module is a standalone addition to the OUAR designed to work alongside it as an alternative recommender for user-requested recommendation.

The work will first explore recommender systems as a whole with the different methods that these systems employ and the state of the art in recommender systems, especially in education environment. Then the OUAR will be briefly touched, along with the Open University study plan, to put the module into context and discuss possible metrics for evaluating student activity. As the module uses document similarity to recommend materials, different methods to represent documents in vector space will be detailed. Design decisions and implementation of the module itself will be discussed. The resulting module was evaluated offline by using precision and recall metrics along with custom rank metric, which will be discussed in the evaluation section.

Chapter 2

Recommender Systems

In [15], recommender systems are defined as software tools and techniques that recommend *Items* to *Users*. Their goal is to provide as accurate as possible suggestions of items to users and present them. These items represent the object of recommendation, be it a book in an online library system, movie on a movie rental site, country on a vacation seller site or an academic text in a school information system. Recommendation systems are usually tailored to provide suggestions for specific items so that they can make accurate suggestions and present them in a suitable way.

The users targeted by the recommender systems are ones who lack the insight into the items at question or are in need of items of some type. The recommender systems provide the individual user with suggestions based on attributes relevant to the item and user in question. For example, an online book recommendation site may recommend works of author Terry Pratchett to user, who is known to enjoy works of author Neil Gaiman.

In order for the system to provide a suggestion, there is often a need for information about the individual users. This information can be collected explicitly, for example through analysis of user's past ratings of items or by asking the user to provide examples of items he prefers, which is employed by the site MovieLens [13] for example. Other than explicitly stated information, actions by the users can be interpreted as implicit signs of preference, example being browsing a particular product category page.

In e-learning, recommender systems must face additional challenges. When recommending commercial items like movies or books, user preferences are usually sufficient to start making recommendations. In e-learning, the system must take user's proficiency, learning goals or context [8] into account, as it would not be ideal to recommend complex study materials to a beginner learner. This creates the need to track the user's proficiency level or needs and recommend accordingly. The recommender system should be able to create a path of learning for the user and to guide him along the path, modifying it as is necessary. This is easier in the environment of an university information system as it provides continual information about the user's success in learning through graded assignments or quizzes.

Alternatively, the recommender system can disregard the user and make recommendations only on the basis of recommended item attributes and similarity, this can be used for example to recommend study materials or academic texts.

Formally, the recommendation problem can be formulated as follows [1]: Let C be the set of all users of the system and let S be the set of all possible items that can be recommended, be it books, study activities or vacation places. Let u be a utility function measuring the usefulness of item s to user c, i.e., $u : C \times S \rightarrow R$, where R is a totally ordered set. Then, for each user $c \in C$, we want to choose such item $s' \in S$ that maximizes the user's utility. More formally:

$$\forall c \in C, s \in S \qquad s'_c = arg\,max\, u(c, s) \tag{2.1}$$

2.1 Recommendation techniques

As there are many applications for recommender systems and different ones require different approaches, recommender systems vary. Generally, they can be classified into groups based on their approach to recommendation. According to [8], the categories are content-based filtering (CB), collaborative filtering (CF), matrix/tensor factorization (MTF) or association rules mining (ARM). A few example recommender systems that use said techniques can be seen in table 2.1.

Recommender system	**Application domain**	**Approach**	**User type**
Smart trade exhibition finder	E-government	CF, Hybrid	Business
Group Lens	E-resource	CF	Individual
ACR News	E-resource	CB, clustering	Individual
If Web	E-resource	CB	Individual
GRec_OC	E-group	CF, CB	Group
CinemaScreen	E-resource	CF, CB	Individual
AHA!	E-learning	ARM	Individual
E-learning recommender agent	E-learning	ARM	Individual

Table 2.1: Examples of recommender systems, their used approaches and user type [11]

2.1.1 Content-Based Filtering

Content-Based recommender systems work by analyzing the items previously rated by the user to create a model or profile of said user based on the features of the rated items [10]. This profile then represents the user's tastes and preferences and is then used to recommend new items to the user. The profile is matched against the items and according to information in the profile, item's relevance to the user is determined. The higher the calculated relevance the better the recommendation. From the calculated relevances, the top N are selected and then presented to the user as recommendations.

An example would be a user, who gave high ratings to movies from a specific director and low ratings to romantic comedies. Each item's attributes would be compared to the user's preferences and relevance would be computed. For this user, the items with the highest relevance would be the films from the specific director that are not romantic comedies.

This approach is beneficial in that it is transparent as in it is clear how the recommendation was made according to the user profile's and item's attributes. Also, it is applicable in systems with few users as it relies solely on the specific user's preferences. There is also no problem with recommending items with few ratings, such as new items or relatively

unknown items because the recommendation is made based on the item's attributes, which are entered into the system along with the item.

The problems with this method are three-fold. The amount of attributes of items is limited and all of them have to be specified either manually or automatically. This means that not every item will have all the attributes to discern whether or not the user will like or dislike it. For a film, the list of all actors, director and writers, along with the film genre is needed to make a perfect recommendation. Sometimes all that information may not be present in the system. The next problem is that the nature of items recommended to the user is based solely on his rating history, so the user will always be recommended items with the same or very similar attributes. But there may be items slightly different from the preferred items that the user might enjoy. Recommender system using only content-based approach will not be able to recommend those. The biggest problem of the approach is the new user problem. When the user first joins the system, his profile is without any information on his preferences so the recommendations cannot be made. The user must first rate some items for the recommender to function. The workaround some systems use is that after user joins the system, it asks him to rate few random items with different attributes. Then the system has some reference data and can base the early recommendations on those and then build the user's profile as he rates more and more items.

The architecture of such recommender system can be seen in figure 2.1.

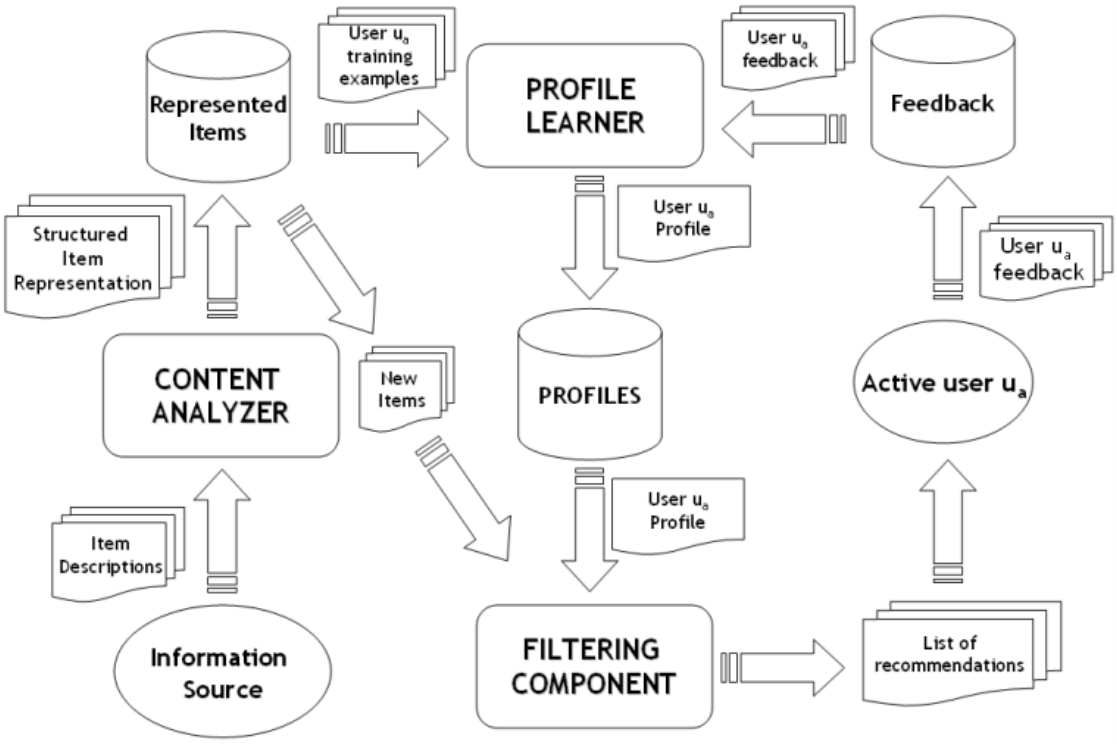

Figure 2.1: High level architecture of a Content-based Recommender [10]

2.1.2 Collaborative Filtering

Collaborative filtering, unlike content-based filtering, tries to predict the utility of an item to a user based on the items previously rated by other users [1], example of rating matrix

from a movie recommendation site can be seen in table 2.2. This approach finds the users similar to the user who is being recommended to. This is done by evaluating each user's rating history and finding users with similar tastes and marking them as peers. These systems can also use stereotypes to deem users peers, such as location, age, gender, etc. For each user's peer a similarity function determines how similar the two users are. This is used as weight in the calculations to enable more similar users to be used as more relevant factor than users with lower similarity score who could also be peers.

	K-PAX	Life of Brian	Memento	Notorious
Alice	4	3	2	4
Bob	NA	4	5	5
Cindy	2	2	4	NA
David	3	NA	5	2

Table 2.2: A fragment of a rating matrix [1], numbers determine score by a corresponding person for a movie

This allows the system to recommend a wider spectrum of items to a user and solves the narrowness problem in content-based recommender systems. The system can recommend items that the user might not think that they will enjoy, because it may be a film outside their usually preferred genre for instance, but can enjoy because people of similar taste liked it.

Collaborative filtering carries it's own problems though. There is a problem with the fact that different users may use the scale differently. For instance with a scale from one to five (five being best), one user might consider movies rated as three not enjoyable, while other users might think of a three rating as average but not bad. Recommender systems try to solve this problem by using the deviations from the average rating of the corresponding user instead of raw ratings. Example being a user who almost never uses rating of five and considers three a slightly above average score. His three star rating could then be interpreted as another user's four star rating.

Another problem is the new item problem. With collaborative recommender systems, newly added items are hard to recommend, as they rely solely on ratings, which are null for the new item. This is usually solved by introducing content-based approaches into the collaborative ones. This approach solves both the narrowness problem of content-based approach and new item problem of collaborative filtering. These systems are considered hybrid systems. A problem that stays even in hybrid systems is the new user problem. For a new user, peers cannot be determined and he does not have a profile yet, so recommending is difficult. This is again partially solved by asking new users to provide examples of their taste and basing the peers and profile on that.

2.1.3 Matrix/Tensor Factorization Techniques

Matrix or tensor factorization techniques are used primarily for predicting student performance. In student performance prediction, there are two crucial aspects, as stated by [8], which are:

1. Probability of student, who does not possess the knowledge to solve a problem, to guess the correct solution (*guess factor*) and the probability of a student, who knows how to solve the problem, to make a mistake (*slip factor*).

2. Improvement of subject knowledge over time, e.g. the more the student repeats the task, the better he performs on average.

Tensor and matrix factorization techniques are appropriate here, because they implicitly take the guess and slip factors into consideration and are suitable for solving the time aspect problem.

Matrix factorization is a task of approximating a matrix X by the product of two smaller matrices W and H, i.e. $X \approx WH^T$. In the recommender system context, the X matrix is the partially observed ratings matrix - here, ratings represent if the problem has been solved, e.g. 0 - not solved, 1 - solved. The $W \in R^{U \times K}$ is a matrix where each row u is a vector containing K latent factors describing the user u and $H \in R^{I \times K}$ is a matrix where each row i is a vector containing K factors describing the item i. Let w_{uk} and h_{ik} be the elements of W and H, respectively, then the rating given by a user u to an item i is predicted by:

$$\hat{r}_{ui} = \sum_{k=1}^{K} w_{uk} h_{ik} = (WH^T)_{u,i} \tag{2.2}$$

W and H are the model parameters that can be learned by optimizing the objective function given a criterion such as root mean squared error [8].

The *slip* and *guess* factors can be encoded within the matrix factorization by using the biased matrix factorization model. This uses user and item bias, respectively, the student and solving-step biases. Student bias models how likely is the student to success in a task and solving-step bias models how likely is the step to be performed successfully - it's difficulty. This modified prediction function is as follows:

$$\hat{r}_{ui} = \mu + b_u + b_i + \sum_{k=1}^{K} w_{uk} h_{ik}, \tag{2.3}$$

μ, b_u and b_i meaning global average, user bias and item bias, respectively.

To further expand the factorization, temporal effect can be taken into account. With taking time into account as another dimension of the tensor, we get these equations:

$$\hat{r}_{ui} = \mu + b_u + b_i + \sum_{k=1}^{K} w_{uk} h_{ik} \Phi_{Tk}, \tag{2.4}$$

$$\Phi_{Tk} = \frac{\sum (T - T_{max} + 1) * q_{tk}}{T_{max}}, \tag{2.5}$$

where q_k is a latent factor vector representing the time, and T_{max} is the number of solving steps int the history that we want to go back. There are further modifications of this model, factoring in for example the fact that students forget information in time.

The problems with the tensor/matrix factorization approach, as stated in [18] are mostly performance based. Computing recommendations using this technique can be time and memory consuming when done over a large data sets.

2.1.4 Association Rules Mining

Recommender systems use association rules mining techniques to, as the name implies, find association rules among the recommended items. These rules represent correlation between

the items in a database [8]. The rules consist of an antecedent (left side) and consequent (right side) and the intersection between the two must be empty. An association rule is defined as such:

$$X \implies Y, \tag{2.6}$$

where $X, Y \subseteq I$ and I is a set of database items and X, Y are sets of items, sometimes called *itemsets*.

Practical example of an association rule could be a rule from the electronics e-shop domain:

$$\{phone, USB\,cable\} \implies \{powerbank\} \tag{2.7}$$

This rule would mean, that when phones and USB cables are bought together, they tend to also buy a powerbank.

Usually, the association mining algorithms require the user to set at least two thresholds, one for minimum support, the other for minimum confidence [8]. *Support* in the context of association rules mining is defined as an indication of how often does an itemset appear in the dataset. The $confidence$ is then defined as a probability of how often is the rule satisfied, using the example rule above, confidence would be a measure of how often people who bought phone and USB cable also bought a powerbank.

The rules can be valuable in finding interesting or unexpected patterns in user behavior that could be of use in the recommendation process. In the context of e-learning recommender systems, these rules could provide valuable information about the ways students find information. The disadvantage of using this approach is that there is usually an overwhelming amount of rules found, and they are not guaranteed to be relevant.

2.2 Using ontologies to provide contextual information

In the area of e-learning recommendation systems, context is a very valuable and necessary information. It is important to know what the student is currently studying or interested in and recommend appropriate materials from that specific field. This also allows the use of content-based filtering in the context of e-learning since the topic of a material can be used as item attribute and then materials of similar attributes can be recommended. Ontologies help with this problem. Domain ontology can be considered as a set of logical axioms designed to account for the intended meaning of a vocabulary [7]. Informally it can be described as a „vocabulary“ providing definitions of terms of a selected domain. These domains can be for example physics, chemistry or machine learning.

In the area of e-learning recommender systems, ontologies provide a way to incorporate content-based and collaborative techniques into the systems, as they are a sufficient way of categorizing study materials. We can set attributes or tags of a specific document with the use of ontology classification and then work with it using standard content-based or collaborative techniques.

Chapter 3

Open University Analyse Project

3.1 Open University study plan

Education in Open University takes form of offered courses taught via distant learning. The individual course runs are called presentations and most courses have two presentations per year. Presentations are divided into blocks taught over the course of several weeks. Blocks are further divided into parts which can be taught one part per week or multiple parts per week depending on the course. Variant of a course study plan is visualized in figure 3.1.

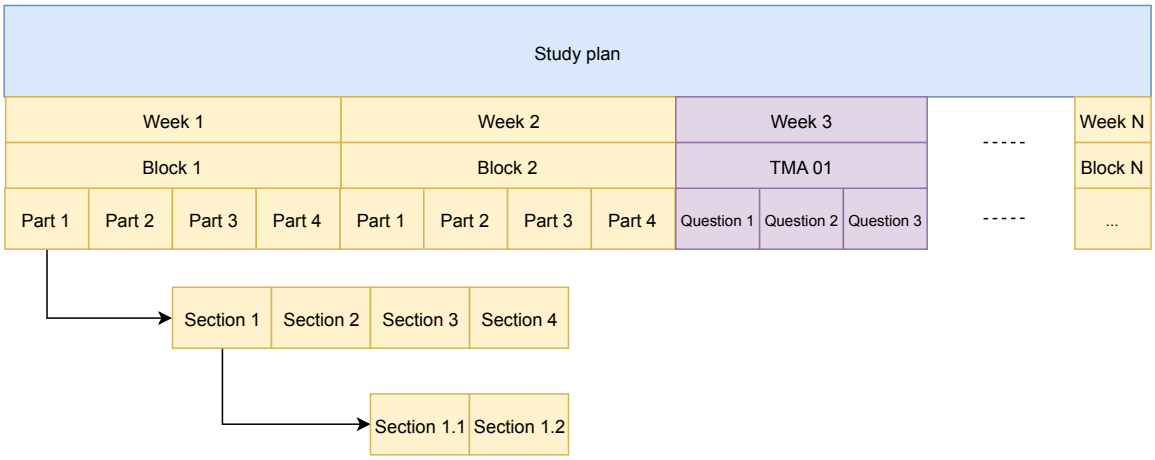

Figure 3.1: Visualisation of the study plan arrangement

The students are assigned tutors, members of the faculty, who are supposed to help them and guide them during their studies as well as evaluate them. Student evaluation is done mainly via Tutor Marked Assignments (TMA) which can be thought of as marked homework. Study materials are usually available through the Virtual Learning Environment and consist of text materials, media, quizzes or experiments. Students are classified into three groups, based on their average scores of all TMAs as such:

1. **Excellent**, with TMA scores over 75%, with the exception of students with TMA scores over 95% - those are considered either of genius level intellect or cheaters.

2. **Pass**, with TMA scores between 40% and 75%.

3. **Fail**, with TMA scores under 40%.

3.2 Virtual Learning Environment

The Virtual Learning Environment(VLE) is an university information system, which contains interface for courses. All of the course's materials can be accessed through the VLE, along with experiments, quizzes, assignments or optional study material. It also contains course forums, which are moderated by course tutors and is meant to be a place, where students can ask questions about specific materials or assignments.

It allows for tracking of user activity, which is used for the purpose of the Open University Analyse Recommender. The recommender system this work is about uses textual study materials from the VLE for recommendation.

3.3 Open University Recommender system

The Open University Analyse Recommender can be categorized, using categories as defined in [8], as one using Collaborative filtering and Association rules to recommend study materials, relevant to passing the selected course. It uses legacy data gathered from the VLE in the previous presentations for recommendation.

Currently, only clicks on study materials are tracked in the VLE. Each click has semantic label, called activity type, which identifies the interaction in VLE it was generated from, be it forum, ou-content (text material), resource, quiz or other. Ou-content represents key study materials, usually in the form of highly structured HTML content, therefore it is easy to track user effort in more detail. As for the other types of clicks, their relevance is not easy to estimate, mostly because resource type represents whole study texts in pdf and quiz and forum relevance can be questionable. Effort, required for understanding of block topic, is measured in terms of average number of clicks of passing students from the previous year.

For the recommendation itself, there are two defined measures, *Relevance* of study material and *effort* of the current student [17].

3.3.1 Relevance

Relevance is defined as a normalized difference of the average cumulative students activity a, measured by the cumulative number of clicks on a specific study activity, between two consecutive weeks *i-1* and i:

$$R(w,a) = \frac{\sum_{i=1}^{w} c_p(i,a) - \sum_{i=1}^{w-1} c_p(i,a)}{\sum_{i=1}^{N} c_p(i,a)}, \tag{3.1}$$

where $c_p(i,a)$ is the number of clicks for the activity a in week i, $\sum_{i=1}^{w} c_p(i,a)$, $\sum_{i=1}^{w-1} c_p(i,a)$ are cumulative clicks from week 1 to week w and *w-1*, respectively. $\sum_{i=1}^{N} c_p(i,a)$ is the cumulative sum of clicks through the whole course last year.

- *Relevance* is always non-negative.
- Sum of the *Relevance* of an activity over all weeks is 1.
- *Relevance* of an activity for a given week is the same for every student.

3.3.2 Effort

Effort measures the activity of a student in VLE and serves as an approximation of the student's progress for given activity and is defined as such:

$$E(w,a) = \frac{\sum_{i=1}^{w} c_c(i,a) - \sum_{i=1}^{w-1} c_c(i,a)}{\sum_{i=1}^{N} c_p(i,a)}, \tag{3.2}$$

where $c_c(i,a)$ is the number of clicks for the activity a in week i from the current student, $c_p(i,a)$ is the number of clicks for the activity a in week i from the previous year, $\sum_{i=1}^{w} c_c(i,a)$, $\sum_{i=1}^{w-1} c_c(i,a)$ are cumulative clicks from week 1 to week w and *w-1*, respectively, from the current student. $\sum_{i=1}^{N} c_p(i,a)$ is the cumulative sum of clicks through the whole course last year.

- *Effort* is calculated for each student.
- *Average Effort* is measured as an average of *Effort* of all students.

3.3.3 Recommendation

As a result, *Importance* of activity a in week w is defined as:

$$I(w,a) = R(w-1,a) - E(w-1,a), \tag{3.3}$$

where $R(w-1,a)$ is the *Relevance* of given activity in a previous week and $E(w-1,a)$ is the *Effort* for the given activity in the previous week.

The *Effort* and *Relevance* for an activity should be similar. If the *Relevance* of an activity is higher than the *Effort* of the student for the activity the system recommends the student to focus on the activity.

3.3.4 Critical Recommendation

Critical Recommendations identify the most important study materials for passing the next Tutor-Marked Assignment (TMA). They recommend to the student the bare minimum necessary study materials for him to pass the TMA. *Critical Recommendations* are evaluated using the difference between activities of students in the *passed* group and the students in the *fail* group from the last year's course.

3.4 Possible Modifications to Relevance and Effort Computation

The main problem of the current implementation of OURecommender is the fact that it does not take activities of type *resource* into consideration when calculating relevance. The reason for this is that they usually represent long texts in pdf format, so their relevance cannot be measured in terms of clicks. There needs to be another way of calculating relevance and effort for use with these pdf files.

3.4.1 Using Time for Calculating Relevance and Effort

Since clicks are not an usable metric for determining relevance of long texts a different one must be used. One possibility is to use time that the student spent on the text to determine

the relevance. Let's consider $t_p(i, a)$ as the collective time spent on activity a in week i and $t_c(i, a)$ as time spent on activity a in week i by the current user. This would allow for modifying the relevance equation 3.1 as such:

$$R(w, a) = \frac{\sum_{i=1}^{w} t_p(i, a) - \sum_{i=1}^{w-1} t_p(i, a)}{\sum_{i=1}^{N} t_p(i, a)}, \tag{3.4}$$

and the effort equation 3.2 as such:

$$E(w, a) = \frac{\sum_{i=1}^{w} t_c(i, a) - \sum_{i=1}^{w-1} t_c(i, a)}{\sum_{i=1}^{N} t_p(i, a)}, \tag{3.5}$$

These modified equations still retain all properties of their non-modified versions and allow for calculating the relevance and effort in the context of *resource* type activities.

Calculating the time spent will not be as easy as summing the time though. The system must take into consideration characteristics of the specific student. This is because the time spent to reach the same results can differ between students significantly. Some students can be fast learners, while others may need to read the text several time to fully understand it. Research would be needed to properly consider the relevance of the difference between students in the context of VLE study materials for relevance and effort calculation.

3.4.2 Using Student Forum Activity for Effort Computation

The student effort calculation could be enhanced by tracking what each student talks about on the course forums. These forums are a place for students to discuss and ask questions about course and are moderated by the tutors themselves. An attempt to find what materials are talked about by the students using document similarity computation as discussed below was made. However since the forums are often too heavily moderated and threads can often only be started by tutors, the amount of student discussion is minimal and the only interactions are between a student and tutor in the form of simple questions and answers. This does not provide enough information to aid in computing either student effort or relevance of discussed material. Still, the forums could prove useful for the recommendation process in the future if fitting method of information mining can be implemented.

3.4.3 Linking Study Texts Together

This modification would aim to improve the user experience by linking study materials together. The goal is to allow students to easily find information about terms used in specific study text and provide link to other study materials explaining the text. For example, if a student reads a text about machine learning and the text uses certain term, say, naive bayes classifier and does not explicitly explain it's meaning, the system can provide link to another study text that explains it.

This could be done by building domain ontologies for taught disciplines which could then be represented as a tree, that would allow for searching the proper documents explaining a specific term. [19] proposes such a tree. Let R be the root of the domain tree and node C_i the representation of a concept under R, then:

$$R = \cup_{i=1}^{n}, \tag{3.6}$$

where n is the number of concepts in the domain. Each of the concepts C_i consists either of sub-concepts, which can be children or leaves representing the actual study material.

Chapter 4

Document Vectorization Methods

The developed recommender system relies on document similarity measures to recommend study materials. This process requires the documents to be represented in vector space, for which many methods already exist, most notably, Term Frequency - Inverse Document Frequency(TF-IDF). Five different methods of document representation were compared in the process of developing this recommender - TF-IDF, Latent Semantic Analysis(LSA), Latent Dirichlet Allocation(LDA), Okapi BM25 and pre-trained Word Embedding vectors. Similarity between two documents was them computed using cosine similarity.

LSA and LDA were only used for reference as they are both widely used for semantical categorization of large texts and will only be touched upon briefly.

For the purpose of this work, Gensim[1] implementation of LSA, LDA and BM25 models and similarity computation was used.

4.1 Term Frequency - Inverse Document Frequency

TF-IDF is a term weighting scheme that assigns weight to the term based on frequency of a term in specific document and its inverse frequency in the whole corpus of documents. The motivation behind this process is that the more the term occurs in text, the more integral it is to the text's meaning. The IDF then assures that the terms with high frequency actually convey meaning and are not just common language constructs such as and, it, then, etc. Generally if there is a high frequency of a word that is uncommon for the language in the text, then that word is integral to the meaning of the text as a whole.

Formally, let $tf(i,j)$ be the frequency of a word w_i in document d_j and $df(i)$ be the document frequency of word w_i. N, being the total number of documents, the inverse document frequency is defined as:

$$idf(i) = log_2(N/df(i)), \tag{4.1}$$

finally, Tf-idf is defined as:

$$tfidf(i,j) = tf(i,j) \cdot idf(i) \tag{4.2}$$

There are some variations of Tf-idf where the tf and idf components are normalized to reduce the influence of document size [2].

To represent a document as a vector using TF-IDF, the TF-IDF weight is computed for every token in the document and the document's vector is then a vector of all the TF-IDF weights. To compute similarity of two documents using these vectors and cosine similarity,

[1] Gensim python library: `https://radimrehurek.com/gensim`

the vectors have to have the same number of dimensions. For this purpose a dictionary of all terms in corpus is built and the document vectors contain TF-IDF values of all terms in the corpus dictionary whether or not the term itself is present in a given document.

4.2 Latent Semantic Analysis

Latent Semantic Analysis (LSA) operates on the premise, that the contexts in which a given term appears or doesn't appear provides a set of constraints that determine the semantical similarity of words to each other. After processing a text corpus, LSA represents its terms or sets of terms as points in a very high dimensional semantic space [6]. This resulting matrix can be reduced using *singular value decomposition*.

4.3 Latent Dirichlet Allocation

Latent Dirichlet allocation (LDA) is a generative probabilistic model of a corpus. The basic idea is that documents are represented as random mixtures over latent topics, where each topic is characterized by a distribution over words [3]. The resulting matrix is consistent of topics instead of terms unlike with other methods.

4.4 Okapi BM25

BM25 is a term weighting scheme expanding on the TF-IDF scheme. It uses a parametrized tf component tf^* depending on parameters k and b with default values of $k = 1.75$ and $b = 0.75$ which were settled on after a long process of academic iteration [2]. Formally, tf^* is defined as such:

$$tf^* = \frac{tf(k+1)}{k(1-b+\frac{b \cdot DL}{AVDL}) + tf}, \tag{4.3}$$

where DL is the document length and $AVDL$ is the average document length in the corpus. Finally, $BM25$ is defined as such:

$$BM25 = tf^* \cdot idf, \tag{4.4}$$

where *idf* is inverse document frequency as defined in 4.1

This modification to the standard TF-IDF formula helps alleviate the problem of evaluating corpora with large variance of length of texts between documents.

4.5 Word Embedding

Word Embedding is a name for a multitude of methods of representing terms as vectors of real numbers. Generally, these vectors are computed using deep learning approaches by analyzing contexts in which the terms are used. The vector values represent relationship to other vector-represented terms in the same vector space and as such convey no meaning on their own.

Word Embedding vectors are available in the form of pre-trained models, which are often trained on colossal amount of text data, often on wikipedia dumps or Google News data. Usage of these pre-trained vectors alleviates the need for training a model for each

application. These pre-trained models often yield better results that models trained on in-house data simply because of the sheer amount of text they are trained on which allows them to represent term meaning more precisely.

A very well known example of a word embedding model is *Word2Vec*. It is a method of computing vector representations of words introduced by a team of researchers at Google. Currently, *Word2Vec* is widely used and is regarded as the state-of-the-art in word embedding. However, there are newer methods for word embedding computation, which claim better results, which is why the author decided to not use the *Word2Vec* model and explore the alternatives.

For the purpose of this work, following pre-trained Word Embedding models were used:

1. Stanford GloVe Wikipedia 2014 + Gigaword 5[2]
2. Stanford GloVe Common Crawl 840B
3. ConceptNet Numberbatch[3]

[2]Stanford GloVe: `https://nlp.stanford.edu/projects/glove`
[3]ConceptNet Numberbatch: `https://github.com/commonsense/conceptnet-numberbatch`

Chapter 5

Fulltext Recommender

The Open University Analyse Fulltext Recommender is a system for recommending text study materials to students of Open University, Milton Keynes. This system's primary use is to find the most suitable study materials that will help students to complete course's Tutor Marked Assignments (TMA). The system operates on the idea, that the documents which are semantically similar to a TMA question will be dealing with the same problem as the question itself and hence can yield the solution to it. These closest documents are recommended to the querying user.

The developed recommender is user-agnostic, it only uses the relationships between the recommended items for the recommendation process. The reason for this design decision is that there is already a recommender system online at the Open University that uses student activity and demographic data. As the original system aims at underperforming students to help them pass the course, this one is designed to be a general study aid for all the students or a tool for tutors to reflect on their TMA and see, for example, if the information they demand from students is actually readily available in their course materials or not. The recommender interface will be available to students and staff via the Open University Analyse Dashboard, where they can query for recommendation for any prepared course TMA.

It only works with text data that is available to student via the Virtual Learning Environment (VLE) - an information system containing study texts, forums, assignments, quizzes, etc. The material type this work is concerned with is called *oucontent* and is available in the database in the form of highly structured XHTML pages. The TMAs are also available in the VLE as *oucontent*, which allows the recommender to treat them the same as the actual study texts when it comes to document similarity computation.

The recommender also allows users to find similar documents to ones they provide, be they a document from an OU course they found interesting or an external document provided in text form for which they want to find any other materials available in the course that handles this given problematic.

The recommendation process consists of preparation of course materials, building a matrix of cosine similarities between all of the course materials, which allows the user to request recommendation for a particular course TMA. All of these steps will be detailed below.

5.1 Study Material Preprocessing

The study materials in question are available to students in the VLE as XHTML pages. As these pages are not available in the database in plaintext form, they first need to be stripped of all XHTML tags, which was done mainly using Python's BeatifulSoup library[1]. The few nonstandart tags and embedded pieces of code in these pages were stripped using regular expressions.

After converting all of the course pages to plaintext, they had to be split into singular terms (tokens) through tokenization. There are multiple approaches to tokenizing texts for similarity computation and there is some debate on which approach has the best results. Tokenization can be simple, when the text is simply split on spaces or more complex forms of tokenization which take into account hyphenation, multi-word terms, etc. For this work, the NLTK word_tokenizer[2] was used, which splits text on any non-period punctuation. This resulting list of tokens was then stripped of inflections using the NLTK wordNetLemmatizer[3], which uses the online lexical database for english *WordNet* to find the base form of all terms.

The lemmatized terms are sufficient for document similarity computation using TF-IDF. For the purpose of similarity computation with word embedding, the word vectors of all the prepared terms are saved. The pre-trained word embedding models consist of a vocabulary, mapping terms to their vector representation. Using this vocabulary, the document's terms are mapped to vectors, from which a vector representing the document as a whole is computed using their mean. Using an aggregation function on document's word vectors to create a vector representation of a document is a technique commonly used for representation of small texts, such as short abstracts or tweets, ex. [4], but it can work even for larger texts, albeit not as effectively as some more complicated methods, such as *Word Movers Distance* [9] The reason for choosing the vector aggregation approach was that it is a much simpler to implement and, while not as sophisticated as some other methods, the results are still sufficient for the purpose of recommending study materials in the scope of a single course. Arithmetical mean was used as a vector aggregation method.

The course material data is saved in a DataFrame[4], which is then saved to database in a binary format.

5.2 Similarity Matrices

The system uses precomputed document similarities for recommendation purposes. These similarities are stored in the form of binary files containing document similarity matrices. These matrices contain document similarity scores between each and every text material in the form of cosine similarity of the two documents. The cosine similarity is a measure calculating the cosine of the angle between two vectors. In the context of word vectors this represents overlap between the features of the two word vectors and as such, cosine similarity is widely used for this purpose. Since the documents here are represented in the same format as the words, we can treat them as such and use the cosine similarity without the need to modify the method.

[1] BeatifulSoup: `https://www.crummy.com/software/BeautifulSoup/`

[2] NLTK tokenizer: `http://www.nltk.org/api/nltk.tokenize.html`

[3] NLTK lemmatizer: `http://www.nltk.org/_modules/nltk/stem/wordnet.html`

[4] A tabular data structure from the Pandas library: `https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.html`

Cosine similarity is derived from the Euclidian dot product formula and represented as such:

$$cos(\theta) = \frac{\sum_{i=1}^{n} A_i B_i}{\sum_{i=1}^{n} A_i^2 \sum_{i=1}^{n} B_i^2}, \tag{5.1}$$

This measure represents the similarity score between material 1, represented by vector A, and material 2, represented by vector B. Higher the score, more similar the materials. A slice of a similarity score matrix can be seen in figure 5.1

	S111-2017J|W16|1179344:20144660:2.2|TMA 03/Question 1 (20 marks)	S111-2017J|W16|1179344:20144661:2.3|TMA 03/Question 2 (20 marks)	S111-2017J|W16|1179344:20144662:2.4|TMA 03/Question 3 (25 marks)
S111-2017J|W12|1170045:18226972:6.1|Topic 5: Part 2 Reactivity of metals/2.5.1 Displacement reactions	0.821262	0.613466	0.428014
S111-2017J|W2|1169998:18226490:1|Topic 1: Part 1 What is water?/Introduction	0.760114	0.656222	0.494077
S111-2017J|W2|1169998:18226497:4|Topic 1: Part 1 What is water?/1.3 Chemical symbols	0.757886	0.643725	0.510478
S111-2017J|W12|1170045:18226974:8|Topic 5: Part 2 Reactivity of metals/2.7 Practical 2 The ice-tray battery	0.752159	0.628721	0.441493
S111-2017J|W12|1170045:18226966:3.2|Topic 5: Part 2 Reactivity of metals/2.2.2 Alkali metals in water	0.749872	0.478567	0.350970

Figure 5.1: A slice of course similarity matrix

5.3 System Interface and Recommendation

The recommender system was designed to handle two types of tasks. Firstly, it should be able to preprocess textual materials for any given course and compute similarity score matrix for this course. This type of task was named *long running tasks*. The other type of task is the recommendation itself, which is done on the basis of a user's query. For both of these types of tasks, a separate interface was implemented to fit to the use case. The specifications and documentation of the two interfaces can be found at **??**

5.3.1 Long Running Tasks

The *long running tasks* interface design had to take into account the fact, that it requires a large amount of time to complete the task, reason being the large corpora of texts that had to be processed and prepared for recommendation. An interface using message queues was used, precisely, a *RabbitMQ*[5] was implemented to handle the requests for course preparation. This message queue interface works by having a task worker running on the server that acts as a queue listener and processes all the requests that has been sent by the user from the OUA Dashboard. This allows any user with sufficient privileges to request preprocessing and matrix building for any number of courses. The worker prepares each requested course and logs the result to the OU database. A simple diagram of this process can be seen in figure 5.2.

This preparatory step must be done before any recommendation can proceed, as the recommendation works only with saved course files in binary format and does not communicate with the database itself.

[5]Rabbit Message Queue: `https://www.rabbitmq.com`

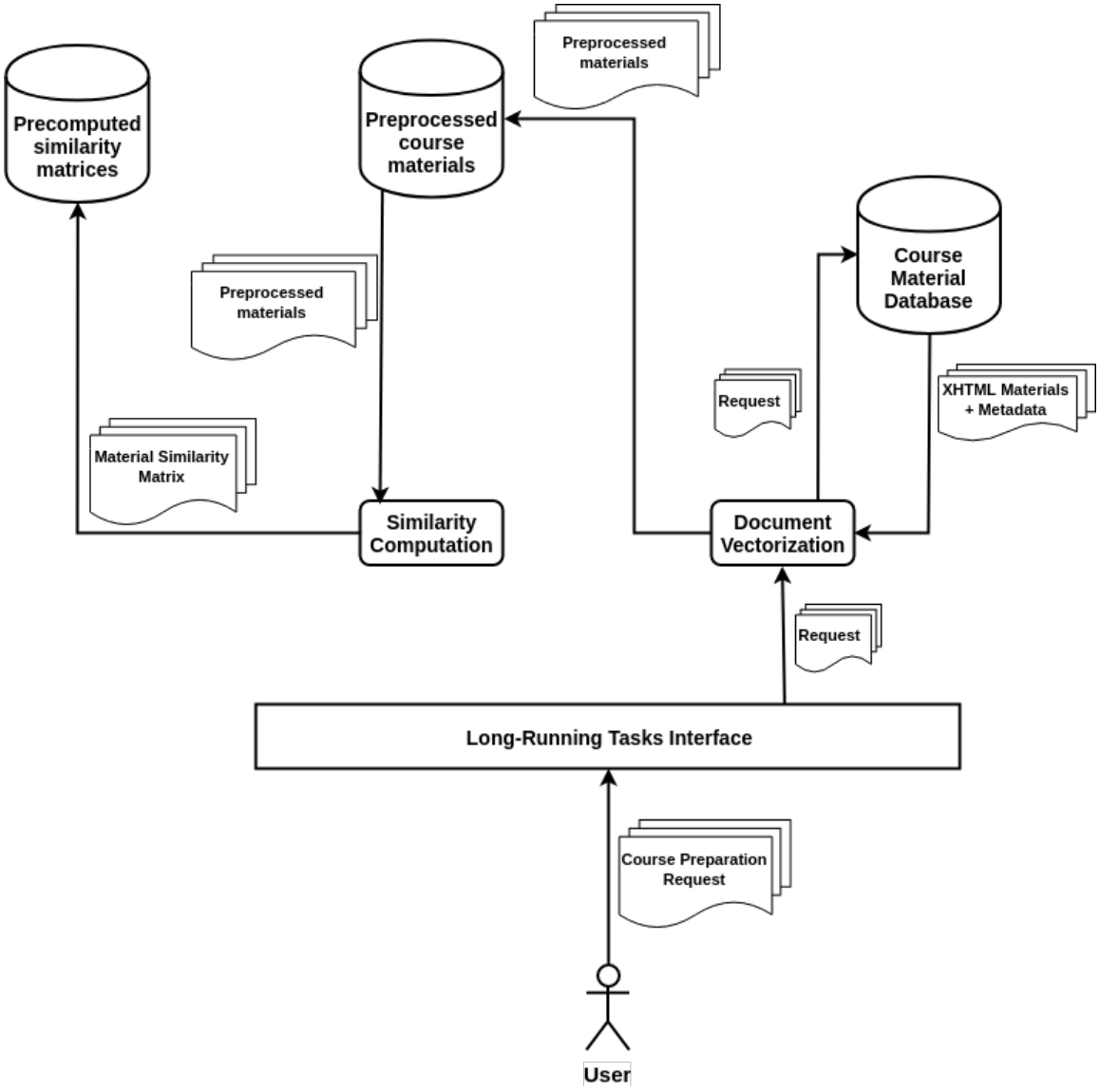

Figure 5.2: A long running task processing diagram

5.3.2 Recommendation

This system provides two basic types of recommendations. The first and the primary type is recommending study materials which are supposed to yield answers to or help with completing a given TMA. This process requires user to specify which of the course's TMAs should the recommendation be made for. Either a recommendation for all questions of a TMA can be made, or a single question can be specified, should the user request it and the TMA questions be present as separate pages in the VLE. The user can further specify the number of text materials to be recommended to him for every single TMA question. After the request is made, the system extracts a slice of the course's similarity matrix containing the specified TMA and presents the requested number of closest VLE text materials to the user.

The other type of recommendation, provided to the user by the system, is finding closest course materials to a material provided by student. This material can either be a VLE material from the course or an external material in text format. This allows the user to easily find any similar materials to ones he may be interested in. If the user uses a VLE material as a query, only materials from the same course can be recommended and the system uses matrix slicing as with the TMA recommendation. But when the user requests recommendation for an external document, he can choose a course from which the closest materials can be extracted. When a course is chosen a small temporary similarity matrix is built with similarity scores between all the course materials and the query document. Requested number of materials is then chosen from this matrix and presented as with TMA recommendation.

For the purpose of this recommender a simple RESTful APi was devised, which is accessible through the OUA Dashboard. A simple diagram of this process can be seen in figure 5.3.

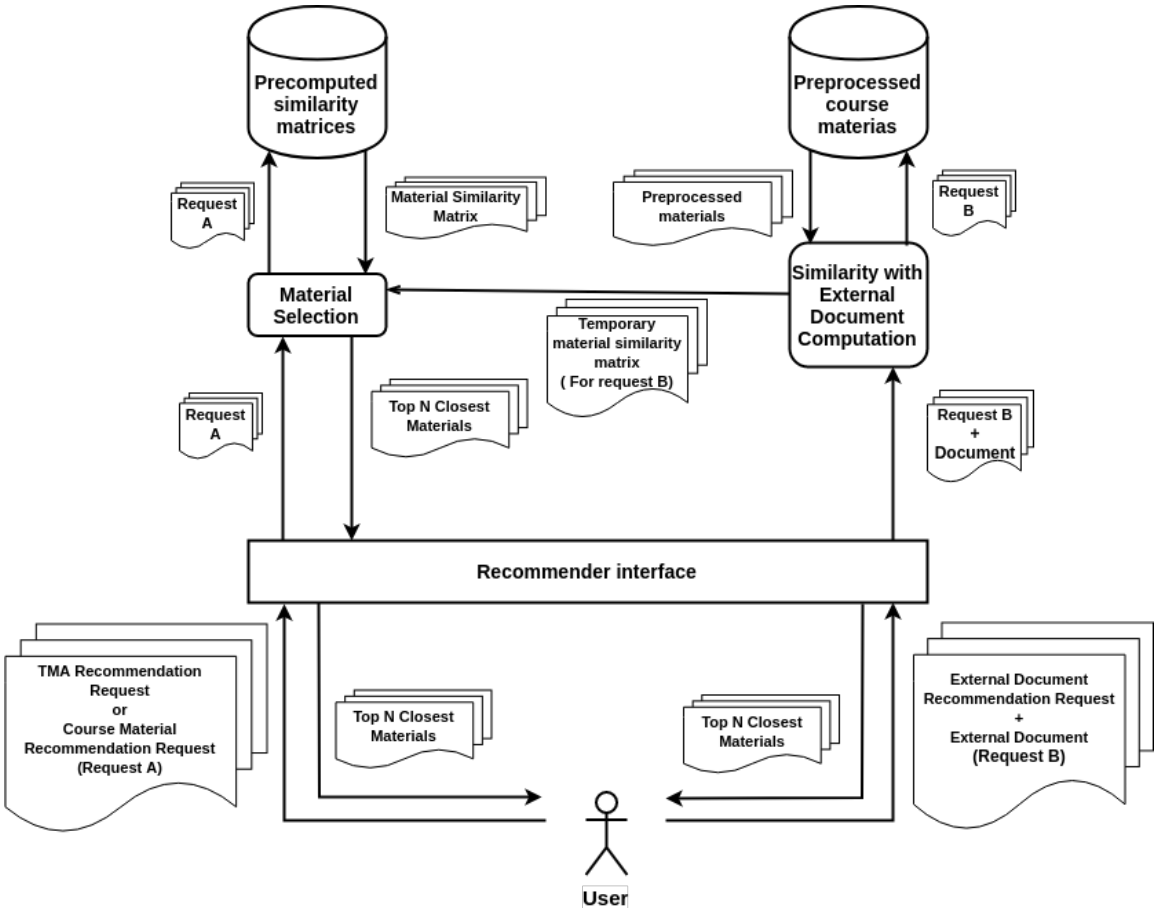

Figure 5.3: A recommendation query processing diagram

Chapter 6

Evaluation

This recommender system was evaluated offline by measuring it's performance on one selected course using metrics which will be specified further.

The system was evaluated using OU course S111-2017J, which is a beginner science course. This specific course has separate VLE pages for each TMA question, which allows the system to make recommendations for each question separately. Each of this course's TMA questions also contains a list of text materials that will help solve the question, which was used as the materials this system should recommend for the questions. This detailed tutor-prepared material recommendation list is only found in this course, which is why it was chosen for evaluation.

This evaluation determines if the document similarity approach to recommendation produces expected and usable results. This evaluation compares TF-IDF, BM25, Word Embedding, LSA and LDA methods for measuring document similarity and scores their performance in the recommendation for this course.

6.1 Evaluation Metrics

For the purpose of determining if the recommendation made is actually useful, *precision*, *recall* and *F Measure* metrics were used [12]. *Relevant documents* in the following definitions are the materials recommended by the tutors and *retrieved documents* are documents recommended by the system.

Precision is the fraction of documents that are relevant. Formally:

$$Precision = \frac{\#(Relevant\ documents\ retrieved)}{\#(Documents\ retrieved)} \tag{6.1}$$

Recall is the fraction of relevant documents that are retrieved. Formally:

$$Recall = \frac{\#(Relevant\ documents\ retrieved)}{\#(Relevant\ documents)} \tag{6.2}$$

F Measure is the weighted harmonic mean of precision and recall. Formally:

$$F\ Measure = 2 * \frac{Presicion * Recall}{Precision + Recall} \tag{6.3}$$

As additional measures for evaluating the methods used for document similarity computation, *Mean Rank* and *Median Rank* were used.

Mean Rank is defined as a mean of *Relevant document* positions in TMA question similarity matrices and *Median Rank* is the median of these positions. Formally:

$$Mean\ Rank = \frac{\sum_{i=0}^{n} Dpos_i}{n}, \tag{6.4}$$

where $Dpos_i$ is the row number of *Relevant document* in an ordered question similarity matrix and n is the number of *Relevant documents* across all TMA questions of a course.

6.2 Evaluation Process

The evaluation consisted of recommending the same number of text materials as were recommended by the tutor. This specific course has 6 TMAs with 5 questions each, fifth of which always requires student to reflect on his studies and his progress in the course. For that reason, only the first four questions of each TMA were taken into account if they had the tutor-recommendations.

Each recommendation was scored using *precision*, *recall*, *F Measure*, *Mean Rank*, *Median Rank*. A mean of the scores over the course was used to evaluate each method of similarity computation in regard of its viability for recommendation.

6.3 Evaluation Results

This section contains evaluation results for all methods listed. Only means of scores are listed here, for the full evaluation result tables for each method, see appendix A.

A visualization of the results of *rank* evaluation can be seen in figure 6.1.

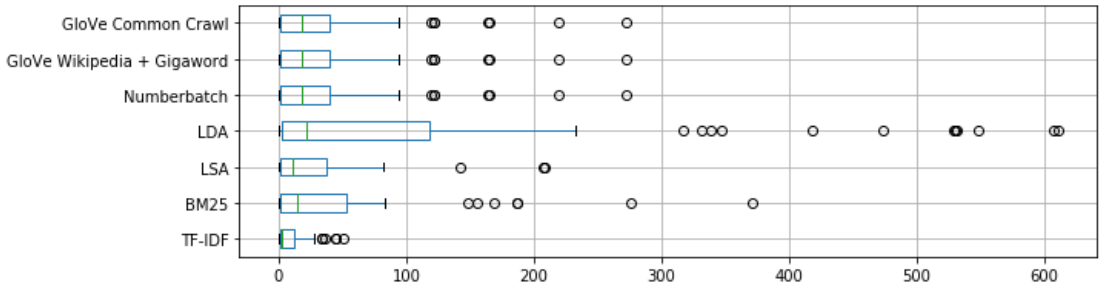

Figure 6.1: A boxplot of tutor-recommended material positions in the respective similarity matrix for each method

6.3.1 TF-IDF

The TF-IDF method, even though it is the simplest method of the ones used, ended up with the second best results. This could be attributed to the features of the TMA questions. As they are very information dense short texts with little to no off-topic terms they are ideal for TF-IDF. There is an important thing to consider when looking at the results though. One of the reasons TF-IDF scored so well on this evaluation may be because this course's TMAs use the same terms when asking the questions as are used in the materials answering them. However this is seldom the case, as few tutors let the author know when asked about this. In some cases, the tutors explicitly try to use different terminology for TMAs than

	Mean Score
Precision	0.31
Recall	0.62
F Measure	0.38
Mean Rank	9.36
Median Rank	3.0

Table 6.1: TF-IDF evaluation using mean scores

for other course materials. This could mean that this kind of result for TF-IDF could be an outlier so more experiments are required.

Nonetheless, TF-IDF shows promise for this kind of task.

6.3.2 BM25

	Mean Score
Precision	0.23
Recall	0.54
F Measure	0.31
Mean Rank	44.16
Median Rank	15

Table 6.2: BM25 evaluation using mean scores

Even though the method should be an improvement over the TF-IDF, since it aims to alleviate problems of varying text length in corpus which this application suffers from, it's results are significantly worse. BM25 shows high variance in the *rank* metric, so the results when using this method for recommending are unpredictable. Even manual examination of recommended materials showed that they are not very usable.

6.3.3 Word Embedding

	Mean Score
Precision	0.25
Recall	0.66
F Measure	0.32
Mean Rank	37.05
Median Rank	18

Table 6.3: Word Embedding evaluation using mean scores

Word embedding evaluation resulted in average scores across all used metrics. However, upon manual inspection of the recommendations made, it was observed, that the method manages to identify subtler connections between texts. Among the recommendations made, some advanced optional texts concerning the question's subject were found, which was not the case with the other methods. It also had better results for the TMA questions that concerned experiments, which other methods had problems with because of the different

form of text used. This implies that the method could be able to work around the problem of varying terminology throughout the course.

All three of the evaluated word embedding models yielded the exact same scores. This shows that the used model does not change the result of recommendation in a significant way. As all of the models used were trained on very large amounts of data, this is to be expected.

The Word Embedding method shows great promise in the context of recommending study materials and with some modifications could prove to be very useful.

6.3.4 LSA

	Mean Score
Precision	0.31
Recall	0.74
F Measure	0.39
Mean Rank	27.54
Median Rank	11

Table 6.4: LSA evaluation using mean scores

LSA results were the overall best among the methods explored. This is not surprising, since it is a very refined method for calculating document similarities. There were no obvious problems with the method even when manually checking the results of recommendation. This method could to an extent identify subtler connections between texts using modified terminology, such as the experiment assignments, but those recommendations were not as precise as with word embedding.

One problem of ths method is significant calculation length when using more precise settings for the method. This is however not a significant drawback since the matrix computations are made in advance anyway.

6.3.5 LDA

	Mean Score
Precision	0.23
Recall	0.48
F Measure	0.27
Mean Rank	122.18
Median Rank	22

Table 6.5: LDA evaluation using mean scores

The LDA scored lowest among the methods evaluated. These were the results obtained when generating 200 topics with 20 iterations, which should be enough to provide accurate results. The bad results could be explained by the nature of the questions, which, being only short texts could have yielded not precise enough topics, resulting in ambiguity.

6.4 Evaluation Conclusion

The recommendations made by TF-IDF, LSA and Word Embedding were ranked best among the methods with the recommended materials being very accurate in the most cases. Word embedding ranked lower than TF-IDF and LSA, but manual evaluation of the recommendations made by this method showed promise. Even though the tutor-recommended materials ranked slightly lower than with the previous two methods, the materials that ranked high often concerned the required problematic and yielded the solution to questions provided. Another interesing result of word embedding is that it often recommended optional study materials that other methods did not. While this behavior of the word embedding method could prove detrimental to recommending materials to help with TMA questions, as that requires the recommended materials to ideally not be of too advanced level, it could prove useful when making recommendations for closest materials to user-supplied materials.

For this reason, TF-IDF and Word Embedding were both deemed a viable methods of document similarity computation. Further experiments will be done to better evaluate the two methods chosen. As the *precision* and *recall* metrics are very limited and evaluating a recommender system without users is not very precise, a further testing of the system on tutors is necessary, as the tutors are best suited to evaluate the quality of recommendations made for assignment they themselves devised.

The results of offline evaluation however were promising and deemed satisfactory, which brings motivation to continue with this approach and hints that with further refining of the methods applied, using document similarities could prove viable for recommendation purposes.

Chapter 7

Conclusion

The purpose of this work was to design and implement a system or a module for an existing system for recommending study materials to students at the Open University, Milton Keynes as part of the project Open University Analyse (OUA). The author explored the most common types of recommender systems and their variations for use in the field of education. The existing recommender system developed by OUA, using student activity metrics for the purpose of recommendation, was analysed, alternative metrics of student activity and modifications to existing metrics were explored.

Author decided to design and implement a new recommender system instead of expanding on the existing one. This recommender system uses semantical similarity between textual study materials to identify the most suitable study materials for completing the Tutor Marked Assignments (TMA) of any course.

This was done by representing the textual materials, including TMA questions, in vector space and recommending the closest documents to user's query document - the TMA question. Techniques considered for the purpose of document vecotrization were Term Frequency - Inverse Document Frequency, Okapi BM25, Word Embedding, Latent Semantic Analysis and Latent Dirichlet Allocation. These methods were evaluated using *precision*, *recall* and *rank* metrics, using one course which has tutor-recommended materials available for every question, which were used as *relevant documents* for the purpose of these metrics.

The LSA, being a widely used method for computing document similarities, was used as a baseline. It had the best results in the evaluation metrics of the methods. The method however lacked the more nuanced recommendations of the word embedding method.

LDA was used to compare other methods to a topic modelling one. It scored lowest in the evaluation. This could be caused by the nature of the TMA questions, them being rather short and often composed of a number of subquestions. This may have lead to LDA not being able to characterize the question with precise enough topics.

BM25 ended with sub-par results during the evaluation. It showed very high variance between quality of recommendations made as was deemed not suitable for the purpose of this recommender system.

TF-IDF method showed very good results in the evaluation. This, however is attributed to the nature of TMA questions used in the evaluation process. The questions used the same terminology as the text materials of the course, which does not happen very often, as some tutors informed the author. Some tutors explicitly try to use different terminology in their TMAs. TF-IDF will be subject to more experiments to determine the impact of different terminology in TMA and in text materials.

The word embedding method yielded average results in the evaluation metrics. This was offset however by manual investigation of recommended materials. These showed that word embedding is able to find subtler similarities in documents, because it was able to find optional materials that concerned the subject of the question using different terminology. It also boasted better results than other methods when the TMA question was an experiment assignment. All this reveals word embedding as a promising method that could yield very precise results with further modifications, such as using weighted document vectors.

The resulting recommender system shows promise with its use of document similarity as a metric for recommendation as the evaluation results were satisfying and show that with modifications, this approach to recommending study materials could be used to great merit.

Author's work also served as a basis for a research paper submitted to EC-TEL[1] which was rejected, but will be corrected and submitted for the next conference.

7.1 Possible Future Work

As the results of the evaluation were promising and the OUA has demonstrated interest in continuing the development of this system, further evaluation is proposed. As it is very difficult to evaluate a recommender system offline, an evaluation using the course tutors is planned to determine the actual usefulness of the system's recommendations.

Further modifications to the recommender system could marginally improve the quality of provided recommendations. Most notably, the methods for measuring document similarity could be more refined and other could be tested, for example World Movers Distance shows promise in this area. Other modifications, like weighted document vectors or topic models could prove useful.

Other attributes of the study materials could be considered. For example, the length of a document could be used to modify relevancy or use could be made of more material metadata, like if the material contains a video. Other types of VLE content may prove useful, quizzes, forums and experiments are currently not used by either of the OU recommenders.

Another modification to the system could be cooperation with the already live OUA recommender, which uses student activity for recommendation purposes. Using these two recommender systems together could yield interesting results, but there are problems regarding different granularity of study materials and few other problems that would have to be resolved first.

[1] European Conference on Technology Enhanced Learning: `http://www.ec-tel.eu/`

Bibliography

[1] Adomavicius, G.; Tuzhilin, A.: *Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions* . 2005.

[2] Alvarez, J. E.: A review of word embedding and document similarity algorithms applied to academic text. 2017.

[3] Blei, D. M.; Ng, A. Y.; Jordan, M. I.; et al.: Latent dirichlet allocation. *Journal of Machine Learning Research*. vol. 3. 2003: page 2003.

[4] Boom, C. D.; Canneyt, S. V.; Demeester, T.; et al.: Representation learning for very short texts using weighted word embedding aggregation. *Pattern Recognition Letters*. vol. 80. 2016: pp. 150 – 156. ISSN 0167-8655.
doi:https://doi.org/10.1016/j.patrec.2016.06.012.
Retrieved from:
`http://www.sciencedirect.com/science/article/pii/S0167865516301362`

[5] Davidson, J.; Liebald, B.; Liu, J.; et al.: The YouTube Video Recommendation System. In *Proceedings of the Fourth ACM Conference on Recommender Systems*. RecSys '10. New York, NY, USA: ACM. 2010. ISBN 978-1-60558-906-0. pp. 293–296. doi:10.1145/1864708.1864770.

[6] Folz, P.; Laham, D.; Landauer, T.: An Introdution to Latent Semantic Analysis. 1998.
Retrieved from: `http://lsa.colorado.edu/papers/dp1.LSAintro.pdf`

[7] Guarino, N.; et al.: Formal ontology and information systems. In *Proceedings of FOIS*, vol. 98. 1998. pp. 81–97.

[8] Klašnja-Milićević, A.; Ivanović, M.; Nanopoulos, A.: *Recommender systems in e-learning environments: a survey of the state-of-the-art and possible extensions* . 2015.

[9] Kusner, M.; Sun, Y.; Kolkin, N.; et al.: From Word Embeddings To Document Distances. In *Proceedings of the 32nd International Conference on Machine Learning*, *Proceedings of Machine Learning Research*, vol. 37, edited by F. Bach; D. Blei. Lille, France: PMLR. 07–09 Jul 2015. pp. 957–966.
Retrieved from: `http://proceedings.mlr.press/v37/kusnerb15.html`

[10] Lops, P.; de Gemmis, M.; Semeraro, G.: *Content-based Recommender Systems: State of the Art and Trends*. Boston, MA: Springer US. 2011. ISBN 978-0-387-85820-3. pp. 73–105. doi:10.1007/978-0-387-85820-3_3.

[11] Lu, J.; Wu, D.; Mao, M.; et al.: *Recommender system application developments: A survey* . 2015.

[12] Manning, C.; Raghavan, P.; Schütze, H.: *Introduction to Information Retrieval*. Cambridge University Press. 2008. ISBN 0521865719.

[13] Miller, B. N.; Albert, I.; Lam, S. K.; et al.: MovieLens Unplugged: Experiences with an Occasionally Connected Recommender System. In *Proceedings of the 8th International Conference on Intelligent User Interfaces*. IUI '03. New York, NY, USA: ACM. 2003. ISBN 1-58113-586-6. pp. 263–266. doi:10.1145/604045.604094.

[14] Pennington, J.; Socher, R.; Manning, C. D.: GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. 2014. pp. 1532–1543.
Retrieved from: `http://www.aclweb.org/anthology/D14-1162`

[15] Ricci, F.; Rokach, L.; Shapira, B.: *Introduction to Recommender Systems Handbook*. Boston, MA: Springer US. 2011. ISBN 978-0-387-85820-3. pp. 1–35. doi:10.1007/978-0-387-85820-3_1.

[16] Speer, R.; Chin, J.; Havasi, C.: ConceptNet 5.5: An Open Multilingual Graph of General Knowledge. 2017.
Retrieved from: `http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14972`

[17] Zdráhal, Z.; Huptych, M.; Bohuslávek, M.; et al.: *Measures for recommendations based on past students' activity* . 2017.

[18] Žemaitis, M. L. V.: Analysis of the recommendation systems based on the tensor factorization techniques, experiments and the proposals. 2011.

[19] Zhuhadar, L.; Nasraoui, O.; Wyatt, R.; et al.: Multi-model Ontology-Based Hybrid Recommender System in E-learning Domain. In *2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, vol. 3. Sept 2009. pp. 91–95. doi:10.1109/WI-IAT.2009.238.

Appendix A

Evaluation Result Tables

This appendix contains full result tables for recommender system offline evaluation. The tables contain *precision*, *recall* and *F Measure* scores for each TMA question evaluated. Column *#Recommendations* contains the number of recommendations made by the system. The amount of recommended materials corresponds to the number of recommendations made by tutors for the same question.

A.1 Results

		#Recommendations	Precision	Recall	F Measure
TMA 1	**Question 1**	2	0.0	0.0	0.0
	Question 2	5	0.4	0.4	0.4
	Question 3	2	0.4	1.0	0.57
	Question 4	1	0.0	0.0	0.0
TMA 2	**Question 1**	1	0.2	1.0	0.33
	Question 2	3	0.2	0.33	0.25
	Question 3	2	0.0	0.0	0.0
TMA 3	**Question 1**	4	0.4	0.75	0.52
	Question 2	5	0.2	0.2	0.2
	Question 3	2	0.0	0.0	0.0
TMA 4	**Question 1**	1	0.2	1.0	0.33
	Question 3	3	0.4	0.67	0.5
	Question 4	1	0.0	0.0	0.0
TMA 5	**Question 1**	2	0.2	0.5	0.29
	Question 2	8	0.4	0.25	0.31
	Question 3	6	0.4	0.33	0.36
	Question 4	3	0.0	0.0	0.0
TMA 6	**Question 1**	1	0.2	1.0	0.33
	Question 2	2	0.0	0.0	0.0
	Question 3	3	0.4	0.67	0.5

Table A.1: TF-IDF evaluation results

		#Recommendations	Precision	Recall	F Measure
TMA 1	**Question 1**	2	0.2	0.5	0.29
	Question 2	5	0.2	0.2	0.2
	Question 3	2	0.4	1.0	0.57
	Question 4	1	0.0	0.0	0.0
TMA 2	**Question 1**	1	0.2	0.0	0.0
	Question 2	3	0.2	0.33	0.25
	Question 3	2	0.2	0.5	0.29
TMA 3	**Question 1**	4	0.2	0.25	0.22
	Question 2	5	0.2	0.2	0.2
	Question 3	2	0.2	0.0	0.0
TMA 4	**Question 1**	1	0.4	1.0	0.33
	Question 3	3	0.2	0.67	0.5
	Question 4	1	0.0	1.0	0.33
TMA 5	**Question 1**	2	0.2	0.0	0.0
	Question 2	8	0.4	0.0	0.0
	Question 3	6	0.4	0.0	0.0
	Question 4	3	0.0	0.0	0.0
TMA 6	**Question 1**	1	0.2	0.0	0.0
	Question 2	2	0.2	0.5	0.29
	Question 3	3	0.2	0.33	0.25

Table A.2: BM25 evaluation results

		#Recommendations	Precision	Recall	F Measure
TMA 1	**Question 1**	2	0.0	0.0	0.0
	Question 2	5	0.0	0.0	0.0
	Question 3	2	0.2	0.5	0.29
	Question 4	1	0.2	1.0	0.33
TMA 2	**Question 1**	1	0.2	1.0	0.33
	Question 2	3	0.0	0.0	0.0
	Question 3	2	0.0	0.0	0.0
TMA 3	**Question 1**	4	0.2	0.25	0.22
	Question 2	5	0.2	0.2	0.2
	Question 3	2	0.0	0.0	0.0
TMA 4	**Question 1**	1	0.2	1.0	0.33
	Question 3	3	0.4	0.67	0.5
	Question 4	1	0.2	1.0	0.33
TMA 5	**Question 1**	2	0.0	0.0	0.0
	Question 2	8	0.2	0.125	0.15
	Question 3	6	0.6	0.5	0.55
	Question 4	3	0.0	0.0	0.0
TMA 6	**Question 1**	1	0.2	1.0	0.33
	Question 2	2	0.0	0.0	0.0
	Question 3	3	0.0	0.0	0.0

Table A.3: Numberbatch results

		#Recommendations	Precision	Recall	F Measure
TMA 1	**Question 1**	2	0.0	0.0	0.0
	Question 2	5	0.0	0.0	0.0
	Question 3	2	0.2	0.5	0.29
	Question 4	1	0.2	1.0	0.33
TMA 2	**Question 1**	1	0.2	1.0	0.33
	Question 2	3	0.0	0.0	0.0
	Question 3	2	0.0	0.0	0.0
TMA 3	**Question 1**	4	0.2	0.25	0.22
	Question 2	5	0.2	0.2	0.2
	Question 3	2	0.0	0.0	0.0
TMA 4	**Question 1**	1	0.2	1.0	0.33
	Question 3	3	0.4	0.67	0.5
	Question 4	1	0.2	1.0	0.33
TMA 5	**Question 1**	2	0.0	0.0	0.0
	Question 2	8	0.2	0.125	0.15
	Question 3	6	0.6	0.5	0.55
	Question 4	3	0.0	0.0	0.0
TMA 6	**Question 1**	1	0.2	1.0	0.33
	Question 2	2	0.0	0.0	0.0
	Question 3	3	0.0	0.0	0.0

Table A.4: GloVe Wikipedia + Gigaword evaluation results

		#Recommendations	Precision	Recall	F Measure
TMA 1	**Question 1**	2	0.0	0.0	0.0
	Question 2	5	0.0	0.0	0.0
	Question 3	2	0.2	0.5	0.29
	Question 4	1	0.2	1.0	0.33
TMA 2	**Question 1**	1	0.2	1.0	0.33
	Question 2	3	0.0	0.0	0.0
	Question 3	2	0.0	0.0	0.0
TMA 3	**Question 1**	4	0.2	0.25	0.22
	Question 2	5	0.2	0.2	0.2
	Question 3	2	0.0	0.0	0.0
TMA 4	**Question 1**	1	0.2	1.0	0.33
	Question 3	3	0.4	0.67	0.5
	Question 4	1	0.2	1.0	0.33
TMA 5	**Question 1**	2	0.0	0.0	0.0
	Question 2	8	0.2	0.125	0.15
	Question 3	6	0.6	0.5	0.55
	Question 4	3	0.0	0.0	0.0
TMA 6	**Question 1**	1	0.2	1.0	0.33
	Question 2	2	0.0	0.0	0.0
	Question 3	3	0.0	0.0	0.0

Table A.5: GloVe Common Crawl evaluation results

		#Recommendations	Precision	Recall	F Measure
TMA 1	**Question 1**	2	0.0	0.0	0.0
	Question 2	5	0.0	0.0	0.0
	Question 3	2	0.4	1.0	0.57
	Question 4	1	0.0	0.0	0.0
TMA 2	**Question 1**	1	0.2	1.0	0.33
	Question 2	3	0.0	0.0	0.0
	Question 3	2	0.0	0.0	0.0
TMA 3	**Question 1**	4	0.4	0.5	0.44
	Question 2	5	0.0	0.0	0.0
	Question 3	2	0.0	0.0	0.0
TMA 4	**Question 1**	1	0.2	1.0	0.33
	Question 3	3	0.6	0.67	0.63
	Question 4	1	0.2	1.0	0.33
TMA 5	**Question 1**	2	0.0	0.0	0.0
	Question 2	8	0.2	0.125	0.15
	Question 3	6	0.4	0.33	0.36
	Question 4	3	0.0	0.0	0.0
TMA 6	**Question 1**	1	0.2	1.0	0.33
	Question 2	2	0.0	0.0	0.0
	Question 3	3	0.0	0.0	0.0

Table A.6: LSA evaluation results

		#Recommendations	Precision	Recall	F Measure
TMA 1	**Question 1**	2	0.0	0.0	0.0
	Question 2	5	0.0	0.0	0.0
	Question 3	2	0.2	0.5	0.29
	Question 4	1	0.0	0.0	0.0
TMA 2	**Question 1**	1	0.2	1.0	0.33
	Question 2	3	0.0	0.0	0.0
	Question 3	2	0.0	0.0	0.0
TMA 3	**Question 1**	4	0.2	0.25	0.22
	Question 2	5	0.0	0.0	0.0
	Question 3	2	0.0	0.0	0.0
TMA 4	**Question 1**	1	0.0	0.0	0.0
	Question 3	3	0.2	0.33	0.25
	Question 4	1	0.2	1.0	0.33
TMA 5	**Question 1**	2	0.0	0.0	0.0
	Question 2	8	0.2	0.125	0.15
	Question 3	6	0.4	0.33	0.36
	Question 4	3	0.2	0.33	0.25
TMA 6	**Question 1**	1	0.0	0.0	0.0
	Question 2	2	0.0	0.0	0.0
	Question 3	3	0.0	0.0	0.0

Table A.7: evaluation results

Appendix B

Recommender REST API

For the purpose of the communication with the system, a RESTful API is implemented. This appendix lists the available API calls.

B.1 Query for TMA Recommendation

- **Description:**
  - Fetches a list of recommended materials for all pages of specified TMA
- **Request Method:**
  - GET
- **URI:**
  - /tmarec/module/<module_code>/presentation/<presentation_code>/tma/<tma_number>
- **Parameters:**
  - *n:* Number of recommendations to list, default is 10
  - *sites:* If *true*, returns list of VLE sites instead of singular text materials as recommendation
  - *preceding_only:* If *true*, only considers materials from week preceding TMA for recommendation
  - *method:* One of *tfidf/embedding*, specifying similarity computation method
- **Response:**

```
{
    "Question 1": [
        {
            "module": "SXX",
            "presentation": "20XXX",
            "week": "1",
            "sas_id_site": "11111",
            "id_page": "22222",
            "section": "1.2",
            "name": "Introduction to SXX",
            "title": "1.2 Assessment in SXX",
            "similarity_score": "0.7101828932762146",
            "rank": 1
        },
        ...
}
```

B.2 Query for TMA Question Recommendation

- **Description:**
  - Fetches a list of recommended materials for a specific TMA question
- **Request Method:**
  - GET
- **URI:**
  - /tmarec/module/<module_code>/presentation/<presentation_code>/tma/<tma_number>/question/<question_number>
- **Parameters:**
  - *n:* Number of recommendations to list, default is 10
  - *sites:* If *true*, returns list of VLE sites instead of singular text materials as recommendation
  - *preceding_only:* If *true*, only considers materials from week preceding TMA for recommendation
  - *method:* One of *tfidf/embedding*, specifying similarity computation method
- **Response:**

```
[
    {
        "module": "SXX",
        "presentation": "20XXX",
        "week": "1",
        "sas_id_site": "11111",
        "id_page": "1924222229546",
        "section": "3.4",
        "name": "Forms of Water",
        "title": "3.4 Water",
        "similarity_score": "0.69839015007019043",
        "rank": 1
    },
    ...
]
```

B.3 Query for Similarity With Internal Document

- **Description:**
  - Fetches a list of recommended materials for a specific VLE material
- **Request Method:**
  - GET
- **URI:**
  - /docxcourse/module/<module_code>/presentation/<presentation_code>/page/<page_id>
  - /docxcourse/module/<module_code>/presentation/<presentation_code>/cmid/<sas_id_s
- **Parameters:**
  - *n:* Number of recommendations to list, default is 10
  - *sites:* If *true*, returns list of VLE sites instead of singular text materials as recommendation
  - *method:* One of *tfidf/embedding*, specifying similarity computation method
- **Response:**

```
[
    {
        "module": "SXX",
        "presentation": "20XXX",
        "week": "1",
        "sas_id_site": "1169995",
        "id_page": "19249546",
        "section": "3.4",
        "name": "Introduction to SXX",
        "title": "3.4 Forms",
        "similarity_score": "0.29839015007019043",
        "rank": 1
    },
    ...
]
```

B.4 Query for Similarity With External Document

- **Description:**
  - Fetches a list of recommended materials for a specific user-provided text
- **Request Method:**
  - POST
- **URI:**
  - /docxcourse
- **Parameters:**
  - *n:* Number of recommendations to list, default is 10
  - *sites:* If *true*, returns list of VLE sites instead of singular text materials as recommendation
  - *method:* One of *tfidf/embedding*, specifying similarity computation method
- **Request Body:**
  - *module*
  - *presentation*
  - *doc:* provided text
- **Response:**

```
[
    {
        "module": "SXX",
        "presentation": "20XXX",
        "week": "1",
        "sas_id_site": "1169995",
        "id_page": "19249546",
        "section": "3.4",
        "name": "Introduction to SXX",
        "title": "3.4 Forms",
        "similarity_score": "0.29839015007019043",
        "rank": 1
    },
    ...
]
```