FINAL REPORT

Integrated Health Care Information System based on RIA-Technology: Integration of an Anaesthesia specific Recommendation Engine at the BWH

by order of the

Austrian Marshall Plan Foundation

presented by:

Andreas Plank, BSc

Boston, June 2014

General Information

First Name and Surname:	Andreas Plank
Institution:	Salzburg University of Applied Sciences
Course of Studies:	Information Technology & Systems Management
Topic of Research:	Integrated Health Care Information System based on RIA-Technology: Integration of an Anaesthesia specific Recommendation Engine at the BWH
Catchwords:	integrated clinical portal, e-health, recommendation engine, Apache Mahout
Supervisor at University:	Dr. Joachim Steinwendner, MSc.

Abstract

This thesis deals with the conceptualization and implementation of a recommendation engine for the Integrated Clinical Portal/ePortal for Anesthesia Learning (ICP/ePAL) at the Brigham and Women's hospital (BWH) in Boston, Massachusetts. This portal records a high number of parameters and parameter types on which recommendation can be based. Also in an initial phase the search space of the recommendation engine is typically sparsely populated. A solution had to be found for these two issues. The very different use of the portal by every anaesthesiologist was also a complication that needed to be focused. The research led to a multi-part-recommender that scales with the usage of the system and the information the users provides. Therefore several frameworks, recommendation algorithms and similarity metrics were compared to match the requirements of the BWH. After the implementation several tests for computation time and response of the recommender were performed and interpreted.

Contents

List of Tables

List of Figures

List of Abbreviations

API	Application Program Interface
ARFF	Attribute-Relation File Format
BRI	Biomedical Research Institute
BWH	Brigham and Women's Hospital
CF	Collaborative Filtering
CFML	ColdFusion Markup Language
CSV	Comma-Separated Values
DOI	Document Object Identifier
ePal	ePortal for Anesthesia Learning
GUI	Graphical User Interface
HDFS	Hadoop Distributed File System
HMIS	Health Management Information Systems
HTML	HyperText Markup Language
ICP	Integrated Clinical Portal
J2EE	Java 2 Platform, Enterprise Edition
JSON	JavaScript Object Notation
LMA	Longwood Medical and Academic Area
OLAP	Online Analytical Process
PACU	Post Anaesthesia Care Unit
POM	Project Object Model
REST	Representational State Transfer
RIA	Rich Internet Application
ROC	Receiver Operating Characteristic
SSH	Secure Shell
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
WEKA	Waikato Environment for Knowledge Analysis
XML	Extensible Markup Language

1 Introduction

This chapter provides knowledge about the general problem of joining information of different sources. Furthermore the administrative difficulties of an anaesthesia department with reference to the research environment is elucidated and general information about the clinic presented.

1.1 Handling Distributed Information

The wealth of information is a blessing and a curse at the same time. The piece of information that someone needs is always available somewhere, the problem is to find it in the abundance. The pooling of resources, in case of this thesis involving scientific articles, papers, videos and sound recordings, is therefore indispensable. As a result the filtering of knowledge on a users profession, interests and other specifications is feasible. Since a straightforward approach can provide relevant information based on users metadata and the content of articles, the result is not always the best match. With heterogeneous data, this approach is obsolete since the data is not comparable. Therefore other indicators that represent the users taste must be considered to increase the relevance of predictions. Another problem is the consolidation of interfaces, which is difficult to overcome with proprietary software. Without proper access to the needed data, an information merge, specially in real time, is hard to fulfill [4].

1.2 Aim and Purpose

The aim of this thesis is the advancement of health management information systems through the integration of a recommendation service. The clinical information portal of the anaesthesia department at the Brigham and Women's hospital (BWH) serves as an example.

Given the fact that every department in a hospital has its own individual finances, also the decision for the information systems is independent. The speciality about the anaesthesia is that they work close together with several other departments and need to coordinate their work with them. This is not an easy task, since in most cases the different information systems cannot interact. Therefore, the staff has to take a look at several different portals and schedules to get the information about their next cases.

ICP/ePAL was designed to avoid these problems by joining all needed intelligence and also by integrating an information and learning portal, that meets the requirements of the anaesthesia department to save time for the high quality medical staff. To not only make this system unique for the department but even more for the individual users, an anaesthesia specific recommendation engine is conceptualized, implemented and integrated into ICP/ePAL.

1.3 The Brighman and Women's Hospital

The Brigham and Women's hospital arose in 1980 out of the merger of three of the most prestigious Harvard Medical School teaching hospitals - the Robert Breck Brigham Hospital, the Peter Bent Brigham Hospital and the Boston Hospital for Women. With its 793 beds it is the biggest hospital in the Longwood Medical and Academic Area (LMA) in Boston, Massachusetts. It is internationally leading in almost every medical area and is a pioneer in methodology and complex cases. Therefore it is interstitially known and ranked ninth in the 2013 U.S. News & World Report List of Top Hospitals out of nearly 5,000 hospitals nationwide, - not to mention the performance of the first full face transplant in the United States in 2011 and the accomplishment of five lung transplants in only 36 hours in 2004. Beside health care the BWH Biomedical Research Institute (BRI) with its funding from over $640 million and over 1,000 principal investigators is one of the world's largest institute in its profession[5].

2 Fundamentals

Since the middle of the 1990s, where the first papers about Collaborative Filtering (CF) have been published, recommendation systems became an important research area [32] [11]. At that time also the first reports about how Content-Based methods can guide users through the flood of information that the internet provides were published [18]. Over the last decade the academic area as well as the industry have done a lot of work in finding new approaches and algorithms that improve predictions and recommendations. A well-known company for using Collaborative Filtering is MovieLense, a recommendation platform for movies, that provides a big dataset that is mentioned and used in many publications [22]. Amazon published their approaches in extending classical CF-methode to meet their expenses as well [19].

Further approaches applied other techniques like clustering, Bayesian networks and Boltzmann machines to recommendation systems. Clustering techniques are fast after the pre-calculation, but usually produce less personal predictions than other methods. Beside that it was found out, that in some cases recommendations can be less accurate than nearest neighbour algorithms [14]. Asela Gunawardana and Christopher Meek from Microsoft research applied Boltzmann machines for the special case of the in section 4.5 elucidated cold-start problem, where it outperformed an expectation-maximization algorithm in all concerns [2]. Tong Zhang and Vijay S. Iyengar from the IBM Research Division proved in their work with Bayesian networks that the accuracy is always better than with memory-based collaborative filtering approaches [37]. But network based classifiers systems also have their drawbacks. Beside the potential problem of overfitting, a large training dataset is required, which in some cases takes hours to days to process. This means also, that the system is not suitable for rapidly or frequently changing environments [17].

There are also several studies that compare different hybrid approaches to improve the precision of recommendation systems. Nathaniel Good and his team tested a variety of different hybrids that included collabortive filtering, content-based algorithms and knowledge-based techniques in the field of movie recommendation. Their study proved, that hybrid systems performed better than single-algorithem based recommenders. Beside that they engaged testing of multi-part-hybrid systems, which they considered a success [10].

This thesis engages such a multi-part-hybrid system and expounds the scientific process of planing and implementation on the example of the anaesthesia department of the Brigham and Women's hospital.

3 Health Management Information Systems

This chapter will give a basic overview of how Health Management Information Systems (HMIS) can be applied to medical environments, which advantages they bring and which barriers are associated. HMIS are essential for health care organizations since they are the base for tactical and informational planning, administration and evaluation. They suppose to improve and enhance the processes that are necessary in a clinical environment for making products and services more affordable, available and accountable. The optimization of processes is conceived to gain a temporal relief for high quality staff in the medical area. Faster access to patient and case information and a better scheduling are the keys to increasing the efficiency of a medical organisation [16].

3.1 HMIS Components

An integrated HMIS system consists of five major components

- Data/information/knowledge component
- Hardware/software/network component
- Process/task/system component
- Integration/interoperability component
- User/administration/management component

The core of the HMIS system is built by the data/information/knowledge component. It includes the specification, organisation and interrelationships between data, information and knowledge elements that an integrated IMHS requires. The combination of all these different parts is made by intelligent data-mining algorithms, rule engines and online analytical processes (OLAP). From the storage and computation of raw data, this leads to the transformation into information to serve an output for HMIS end-users, in order to make informed and intelligent decisions. This feedback process is illustrated in figure 3.1.

The hardware/software/network is the basis for every information system. It plays a prominent role as it brings the choice of various information storing, handling and computing technologies to support HMIS applications and usage. This component connects the communication infrastructure with various hardware, user interfaces, associated devices and applications to achieve effective and efficient information services. Therefore it is important that all devices can access and communicate with the HMIS application. This is necessary

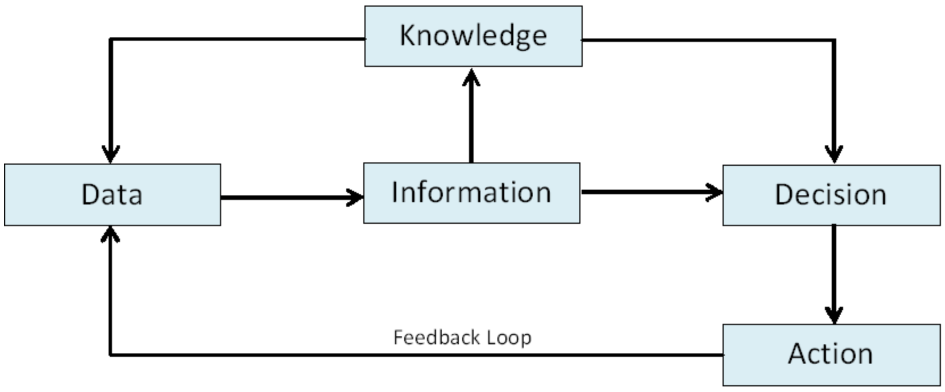


Figure 3.1: Overview over a Data/Information/Knowledge System

to support the people in a technology-driven environment, so they can benefit from increased performance in accomplishing their tasks.

The process/task/system component illustrates the routines of the HMIS. A connection and interaction between all applications, processes and tasks, is the goal of this component. Therefore all clinical-based HMIS as well as administrative applications, such as information systems for human resources and finance as well as scheduling systems and facility utilization, must be conceptualized to collect relevant information. This is needed to optimise task-processing as well as decision-making activities. If every activity is organized into a fixed task-procedure, it can easily be changed or adjusted to new practices since methods are constantly changing. To achieve an optimal functionality between the various applications and task processes a systematic survey is absolute.

For the enterprise point of view the integration/interoperability component is the key to determine the outcome. To position health care services on the market, the interoperability of health care information services with the external environment is often the key to achieve success. It provides an edge over the competitors, if external organizational frameworks can be integrated to achieve efficient, effective and excellent delivery of healthcare services. This requires, beside the understanding of evolving technologies and the changes in organizational task processes, also a broad knowledge about the health care service industry, the market structure and changing characteristics. With this knowledge HMIS applications, which fit not only the current industrial standards but also future requirements, can be designed.

The final and most critical part is the user. The user/administration/management combines and coordinates all other components of the HMIS system. Every user in the system has their particular tasks and activities in the overall system to fulfill the goals of the organization -

provide service to the clients in the most effective, productive and efficient way. Through the interaction with all other components this creates a holistic approach that controls and steers the perceptions and interdependencies of the organization [35] [33] [16].

3.2 Advantages

Health management information systems help clinics to improve the quality and efficiency, while lowering the costs. It can take away administrative burden on high quality medical staff and create a temporal relief. Defined processes, standard practices and treatment guidelines increase the patient's safety, because the space for errors is minimized. This also simplifies the replacement not only of procedures but also of staff. Documentation can be directly integrated in the schedule, which makes it possible to identify mistakes or confirm correct execution in case of complications or litigations. The shown feedback process in figure figure 3.1 helps to improve the treatment methods for one particular person, but also helps other patients since the recorded data can be used for studies and research [13].

3.3 Drawbacks

Since information technologies in medical areas led to easy access of a huge amount of critical data, the protection of patient information is one of the most critical points. Where, in former times only rooms or containers needed to be locked to ensure sensible data to be safe, nowadays this is a much harder task. In most cases the data needs to be protected in several levels. What is the use of an information system that logs every access to patient records, if the data can undetectably be accessed in the database? This requires an ingenious security system that has logs, restriction and policies for all layers. Beside security concerns, medical information systems are expensive. Not only the software itself, but also the infrastructure, maintenance and people needed to keep the system running. Also training of the medical staff is required, an additional expense if the system is not easy to use [16].

4 Recommendation Engines

This chapter provides at first basic information about Recommendation Engines before it gets more detailed with the explanation of different algorithms. Finally it explains how to handle the common problems of unknown users and sparsed data.

4.1 Introduction

Recommending is one of the key parts of machine learning techniques. It is the most recognizable technique, since many websites use it for some reason. On the one hand, nearly every webshop has implemented a recommending system, showing you articles you are most likely to buy on your behavior or prior purchases. On the other hand there are social websites and dating pages which guide you to people you may know or like. There are some prominent examples for recommendation engines.

4.2 Basic Approaches

The first thing that is essential, is information about the user, needed for recommendation to same and data about the item, user, etc., which should be recommended. This section will explain three common ways for recommending.

4.2.1 Content-based Filtering

Every user and item have specific metadata. Information that lets us identify who he/she/it is and that makes it similar or unlike to another. This can be very different information, depending on the matter. There are basically two ways in which this data is can be collected.

- A user's preferences, which represents his interests, are stored in his profile. Which mostly happens while registering. Based on this data a prediction if a user likes or dislikes an item can be computed.
- A history of the user's activities is stored in the recommendation system. This can be for example purchases and views of items. Based on this information a user profile is created, on which a prediction for items can be calculated [27].

An extract of the User-/Item-Metadata that is relevant for recommendation of movies:

- User-Information:
  - Age
  - Language

  - Preferred genre
  - Preferred actors
- Film-Information:
  - Genre
  - Actors
  - Director
  - Age rating

Based on this information it can be predicted, which movie a user is most likely to watch. Either with a direct matching from user to items via decision algorithms or with a clustering algorithm to group similar items or users. The problem with recommendation only on metadata is, that subjective factors are not considered. Not every movie with one particular actor must be good, and one genre can cover a huge variety of different types. Also only the purchase of a movie does not mean the user likes the movie. So recommending only relying on metadata can be a complete flop. To solve this problem user ratings need to be considered.

4.2.2 Collaborative Filtering

As a user rates different items, regardless of which scale, there is a lot more information about taste available than that only from metadata. As it is possible to find other users with a similar taste, at best on many different items, it is possible to predict how likely it is, that the user also likes another item. Therefore it is first calculated how similar a user is to others before the predicted rating for other articles can be computed.

This process is illustrated in figure 4.1. The input is the table with ratings. Based on the circled items, where both users have rated, a similarity is calculated. This is done for user i to all others users from 1 to n. The CF-Algorithm than picks the most similar users to i, based on a fixed value or similarity threshold. Afterwards it computes a prediction for all articles, that the most similar users have read and i not. The `top n` predictions are then represented to the user as recommended. The algorithms to do so are explained in the next section.

4.2.3 Usage-Based Filtering

A Usage-Based recommendation system can use a huge variety of different data. It depends on which data is recorded and how this data can be combined. Most systems put the browser history into count and predict, based on this data, which sites the user is most likely to visit in the future.

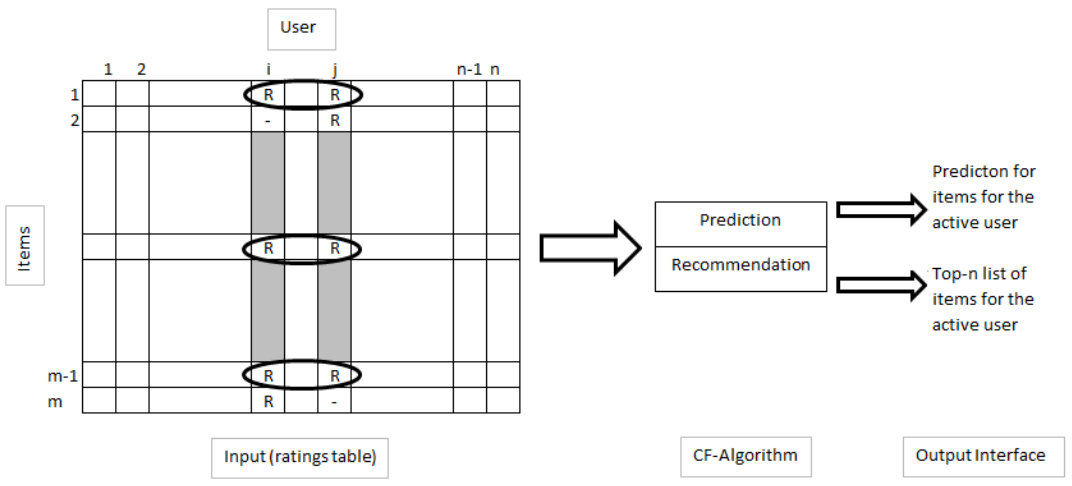


Figure 4.1: Overview over the Collaborative Filtering process

For example a user is searching for different cities in the United States of America and looks them up on a travel guide. The first section he is always heading to is "places of interests" and opens the photos. If he likes the pictures, he goes back to the search engine, looking up the place on the route planner and calculating the distance to an other destination.

If every site is separated from each other, everybody just has a little information.

- The search engine knows, the user is searching for cities in America
- The travel guide knows, that the user is opening photos of different sights in America
- The route planer knows, which cities the user is searching for and which way he will take.

Would all this information now be put together, a specific pattern could be recognized and provide recommendation for hotels on the way, cheap flights and other sites that are interesting for people who travel to the US. Since the patterns are very focused, an integration into a topic specific portal, in that case for travelling, is suggestive.

A much simpler example would be to log search terms on an e-commerce platform and measure the time a user stays on one particular item. This simple data can be used to draw a conclusion if a user liked an article, since he would have continued, if it would not have been the article he was searching for. So the platform shows the customer articles that are similar to the article they have watched for a long time [24].

4.3 CF Algorithm

Algorithms in Collaborative Filtering have the duty to find items that best match the users information. Therefore a similarity between users is calculated and out of it a prediction created. Listing 4.1 represents the pseudo code for the recommendation process.

```
for every item i that u has no preference for yet
 for every other user v that has a preference for i
  compute a similarity s between u and v
  incorporate v's preference for i, weighted by s, into a running average
return the top items, ranked by weighted average
```

Listing 4.1: Pseudo code for recommending process

The outer loop nominates every item, for which the user has no preference yet, as a potential candidate for recommendation. The inner loop takes every user who expressed a preference for this candidate and computes a similarity between the two users. In the end a weighted average based on the value and weight from every user is calculated. Since processing of all items with all users would take a very long time, a neighbourhood with the most similar users is computed first. Only these users are then taken into account when a prediction for the items is calculated.

The weighted average user in CF is calculated as followed:

$$p_{x,u} = \bar{r}_x + \frac{\sum_{i=1}^{n}(r_{i,u} - \bar{r}_i) * w_{x,i}}{\sum_{i=1}^{n} w_{x,i}} \tag{4.1}$$

$p_{x,u}$ is the prediction for the user x to the item u. n represents the number of neighbours and $w_{x,i}$ the weight between the user and its current neighbour u. The more similar the users, the stronger the influence on the result. The calculation of the weight is shown in the next section.

4.4 Similarity Metrics

One of the most important parts of item- and user-based recommendation is to determine the similarity. Knowledge of the most similar items or users is needed for further calculation, since only the top n are into consideration.

4.4.1 Pearson Similarity

The Pearson similarity is one of the most common algorithms in collaborative filtering system. It measures how similar two series of numbers, paired one-to-one, are to each other. The result of the calculation is a value between -1 and 1 and represents how similar two datasets are. Values near 1 mean that the two people are very similar, since 1 would indicate a perfect match. A little relationship is indicated with a value around 0, while -1 indicates an opposing relationship.

After isolating the items that both users, i and j, have rated, the Pearson Similarity is calculated as followed:

$$sim(x,y) = \frac{\sum_{i \in I}(r_{x,i} - \bar{r}_x)(r_{y,i} - \bar{r}_y)}{\sqrt{\sum_{i \in I}(r_{x,i} - \bar{r}_x)^2}\sqrt{\sum_{i \in I}(r_{y,i} - \bar{r}_y)^2}} \tag{4.2}$$

In figure 4.2 three different datasets are computed with Pearson similarity. The result of the left statistical series is 1.0, since both persons have given the same rating for every item. The opinions in the middle plot differ crucially, still there is an overlap as the result of the calculation is 0.0. The last figure shows two people with a very opposing position. The computation results in a value of -0.59. For visualization of the data trend, a linear stripline was inserted, which lets suggest the result value.

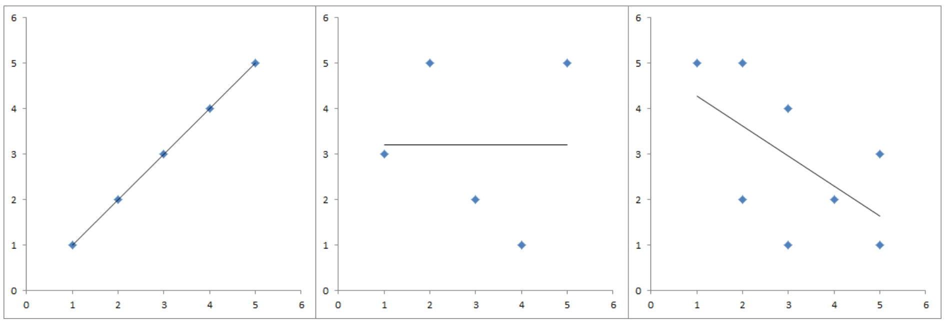


Figure 4.2: Pearson Correlation Demonstration

One problem of Pearson Similarity is, that it can only take items into count that both users have rated. The second problem is, that the correlation cannot be computed, if the users only overlap in one item rating. This is an issue for sparse datasets, where only a little amount of users are overlapping. The last issue is that if a user is rating all of the items with the same value, the Pearson correlation in undefined [31][34].

4.4.2 Spearman Correlation

The Spearman correlation is a variant of the Pearson similarity that is calculated the same, but the data is first brought into relative rank. This means that the least-preferred item gets a value of 1. The next-least-preferred item gets a 2 and so on. Through this process information is lost, but the essence is kept and the user profiles get more harmonized to each other, since it is irrelevant if a user rates all items between 1 to 4 or 3 to 5. [31].

4.4.3 Euclidean Distance

$$sim(x,y) = \frac{1}{1 + \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}} \tag{4.3}$$

The idea is to see a user as a point in a n-dimensional space. The number of dimensions is defined by the quantity of items, and the coordinates by its preference values. To define how similar two users are, the euclidean distance between these user points is computed. The smaller the number, the more similarity is there between the two users. Since in that calculation a larger number would mean less similar, the result needs to be inverted by 1/(1+d). The outcome is always a positive number with a value between 0 and 1 [31][34].

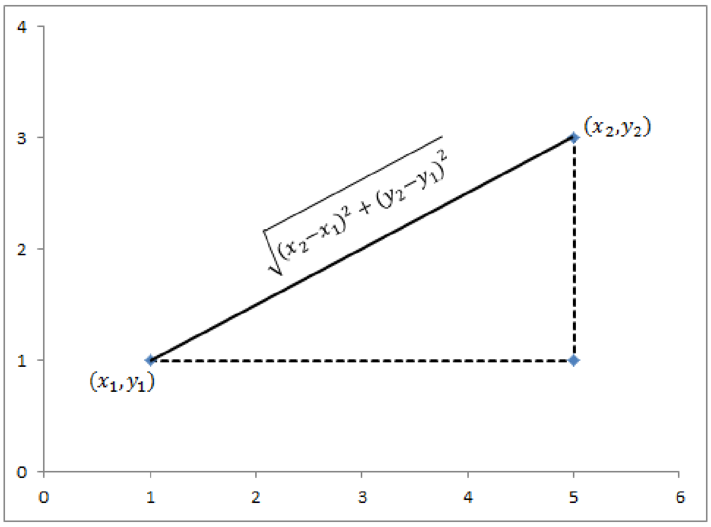


Figure 4.3: 2D Euclidean Distance Demonstration

Figure 4.3 illustrates the calculation of a two dimensional euclidean distance. Every further dimension is only an addition of an other variable $\sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \ldots (x_n - y_n)^2}$, which makes the computation very inexpensive for high dimensional matrices.

4.4.4 Cosine Measure

$$sim(x,y) = \cos(\vec{x}, \vec{y}) = \frac{\vec{x} * \vec{y}}{\|\vec{x}\|_2 * \|\vec{y}\|_2} \tag{4.4}$$

Like the Euclidean distance, also the Cosine measure treats users as points in an n-dimensional space. But in contrast to a distance an angle between the two points is calculated. The smaller the angle between the two users measured from the origin, point(0,0,....0), the more similar they are. Since the cosine of a small angle approaches 1 and a 180 degree angle has a value of -1 no conversion is needed. [31][34].

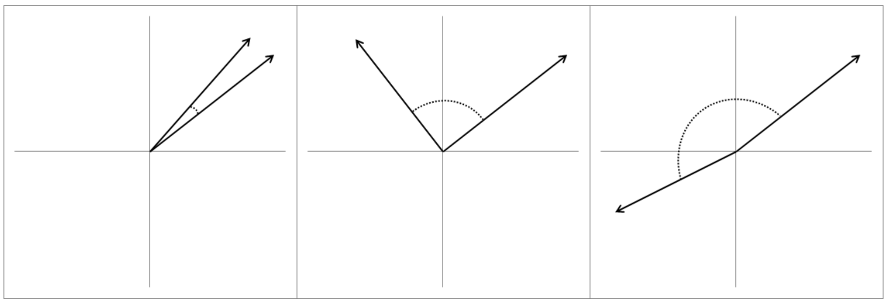


Figure 4.4: Cosine Measure Demonstration

Figure 4.4 illustrates three different two dimensional Cosine models. The angle of the left graph amounts to 12 degrees which indicates a similarity value of 0.978. With a right angle the cosine is zero shows a medium similarity between the two datasets. The right angle is close to 180 degrees which refers to a value close to -1.

4.4.5 Tanimoto Coefficient

$$sim(x,y) = \frac{\sum_{i=1}^{n} x_{di} x_{dj}}{\sum_{i=1}^{n}(x_{di})^2 + \sum_{i=1}^{n}(x_{dj})^2 - \sum_{i=1}^{n}(x_{di})(x_{dj})} \quad (4.5)$$

As it can be seen in figure 4.5 result of the Tanimoto coefficient, also called Jaccard coefficent, represents the overlap between two data samples. The two circles represent the rated articles of the users. The ratio between the overlapping area (intersect) to the whole area (union) is the measure of the similarity. The Tanimoto coefficient does not take the value of the rating into count. It just takes all items, for which the user expressed an opinion. The result varies between 0 and 1, where 1 indicates that both circles are completely overlapping and 0 if they have nothing in common. The big difference to most other similarity models is, that items that not both of the users have rated are taken into count. Most likely this similarity metric is used when only boolean preferences are available.

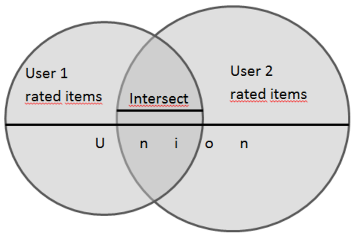


Figure 4.5: Tanimoto coefficent result representation[31]

4.5 Challenges and Limitations

This section shows the challenges and limitations of recommendation systems and how they can be handled. In some points it creates connection to ICP/ePAL.

4.5.1 Scalability

Time is a major factor for recommendation, since it is most likely done in real time. Therefore the algorithm should be fitted to the requirements for time and be appropriate to the size of the dataset. For example: If a dataset only has a small amount of users compared to the items it is most likely to base the similarity model on the user, since the calculation with less variables is faster. Beside the code optimization another solution is to use a framework that can work on a cluster and can be extended when needed. In general there are five questions that should be answered before a system is chosen.

- How many training examples are there?
- What is the batch size for classification?
- What is the required response time for classification batches?
- How many classifications per second, in total, need to be done?
- What is the expected peak load for the system?

The answers to these questions give a basic overview of the requirements the system needs to meet and which algorithms are possible. While some points can be handled with upgrade of the hardware, others cannot be treated that easily [31].

4.5.2 Sparsity Problem

In collaborative filtering systems the user is represented by a matrix of data that is known about the user. This data typically represents ratings, purchases or viewed articles. In most applications the number of items as well as users is fairly large and a user expresses an opinion for only very few items. This means most of the space stays untouched and is represented by zeros. This empty space has a significant negative influence on the quality and effectiveness of collaborative filtering. The less data available, the more decreases the probability to find a neighbourhood of people with similar ratings. The sparsity most likely leads to a result of zero, if any calculation is possible. CF systems do rely on that neighbourhood to predict which articles a user likes. This problem occurs most likely in new systems and if there is a high item-to-user ratio. To counter this problem, domain specific knowledge can be taken into account. [28][6].

4.5.3 Cold-start Problem

One of the most significant challenges for a recommendation system are new users and items. It can be seen as a special case of the sparsity problem, where only one row or line is sparsely filled.

Collaborative Filtering systems are not able to generate useful recommendations to a user about whom they have no information. Therefore some sites request new users to rate some basic articles directly after the registration procedure. The other problem occurs with new items. As long as nobody has rated them, no neighbourhood would refer to this new product. Therefore a content-based approach can be integrated, since it takes the attributes into account and does not rely on ratings in the same way a collaborative filtering approach does [28][6].

5 Analysis and Conceptualization

This chapter will deal with the planning of the implementation. Therefore all the available data will be analysed, proper algorithms defined and potential problems handled. Beside that the framework and programming language will be determined.

5.1 Data Analysis

This section will give an overview over the data that is available for processing. The metadata and usage data has only a unique numeric ID in its individual database table, as with the ItemID. By contrast the UserID is alphanumeric and unique, since it is used for identification in all information systems.

5.1.1 Item Metadata

The item metadata that is relevant for the recommendation engine is defined by the categories, topics and tags that are associated with it. They can be defined during the upload of the article or added and modified in the abstract view. Beside that also the title, source, origin category and author are stored as far as they are available.

5.1.2 User Metadata

The relevant user Metadata contains the level of training, the current rotation and the interests of the anaesthesiologist. The interests are predefined by the categories, topics and tags so they can be directly matched.

5.1.3 Usage Data

The system records several data while a user is using the system. Some they are aware of, some they are not. The logged data is viewed and described in table 5.1.

Since there is a huge amount of usage data stored, one needs to define which data should be used for collaborative filtering and which should not. Rating and Like/Dislike has the same background and is therefore almost a redundant data. The problem is that one is a boolean value, the other on a scale from 1 to 5 and so they can not be directly merged. `Favourite` also gives a strong statement if a user likes an item, while `bookmark` only indicates if a user wants to read the article later.

A very interesting usage data is the "article read". Since a user does not have to explicitly select something, even the users who do not rate, share, bookmark, etc., they give information

Rating	User rating on articles in a scale from 1 to 5.
Like/Dislike	Like and dislike both stored a boolean value.
Bookmark	Bookmark for articles the user wants to read later or again.
Share	Sharing of an article with an other user.
Favourite	Marks the item as favorite so that the user can see it in his favorite-list.
Article Read	Stores if a user has viewed an abstract, or if he has viewed or downloaded the whole article.

Table 5.1: Overview stored Usage Data

on which articles they prefer. It is even better that one can differ if the user has only viewed the abstract or if they have read or downloaded the whole article.

5.2 Cold-start and Sparsity Problems

Since the system was introduced in march 2014 there has not been a lot of available data. This leads necessarily to a cold-start and sparsity problem, explained in section 4.5. Nevertheless, at least there is no anonymous user in the system. They are very basically defined by their metadata that allow an initial prediction for items the user may like.

5.3 The five scalability Questions

As mentioned in section 4.5, there are five questions to answer to give a general basis for all further deliberations. These questions are answered in this section.

5.3.1 How many training examples are there?

At the start of the project there was only very little data, most likely from test scenarios. Until the first of March 215 users have accessed the system, which does not reflect the numbers logged from the portal. Beside the data, that was inserted by a single user, there were only nine ratings from five different users and 79 article views from 27 different users available. Only four persons uploaded 290 items, which represented 94% of data content. The meta data about the users was very limited. Only the core data from the management such as name, birthday, position was accessible. The first article read was logged at the 17th of March. Because of a serious misplanning countless information gets lost in the system. All data that is gathered in the resource section is only saved if the user pushes the back-button after they have viewed, downloaded, rated, etc. an article. Otherwise the information is lost, which, according to the report of users, happens often.

5.3.2 What is the batch size for classification?

The batch size was defined to be ten. After the recommendation a predefined selection of articles will be shown to the user. This list will also compensate the case that a user has not filled out their user profile yet, which should not occur. Beside that, the user, as far as he is a resident, will also receive questions and answers out of an exam dictionary according to the constant review tests.

5.3.3 What is the required response time for classification batches?

The response time should be as low a possible. Since the last question, which deals with the awaited peak, is related to the needed response time, they were treated in combination. In agreement with the department an expected response time from under 300 ms qualifies as a success. This time only defines the response of the recommender, the time of the flow through the middleware and database query is not included.

5.3.4 How many classifications per second, in total, need to be done?

Based on the specified response time a maximum of three recommendations per second need to be made. This number will of course be much lower on average expected throughout the day, but at the peak times, this value will be achieved entirely.

5.3.5 What is the expected peak load for the system?

The highest peak of ICP/ePAL was awaited in the morning and during the lunch hour. Since the cases of the anaesthesiologist do not start at the same time and the lunch breaks are depending on cases, there is no particular peak time. The use of the portal does not mean a load for the recommender, since it is only related to one particular part to the resource module. This theory was reflected in the further course through logs of the read articles. The access is distributed irregularly over the whole day and also night.

5.4 Defining the Recommender

The huge amount of different usage data, the availability of item and user metadata and the Cold-start and Sparsity Problems led to a hybrid recommendation system.

Most of the hybrid system contain two different prediction systems, but in this case it used to be three. The reason for that is that all the recorded data can be divided into three categories.

- Attentive Usage
- Unconscious Usage
- Metadata

The thought behind that is that after speaking to several people of the anaesthesia department, it turned out that some use the system and read articles, but do not want to do anything else like rating, tagging, bookmarking, sharing etc. To provide recommendations on their preference as well, it is possible to take the "article reads" into account.

Figure 5.1 illustrates the three steps from a basic recommendation on Metadata to a recommendation based on the users taste. Therefore we will now talk about a 3-Step-Recommender.

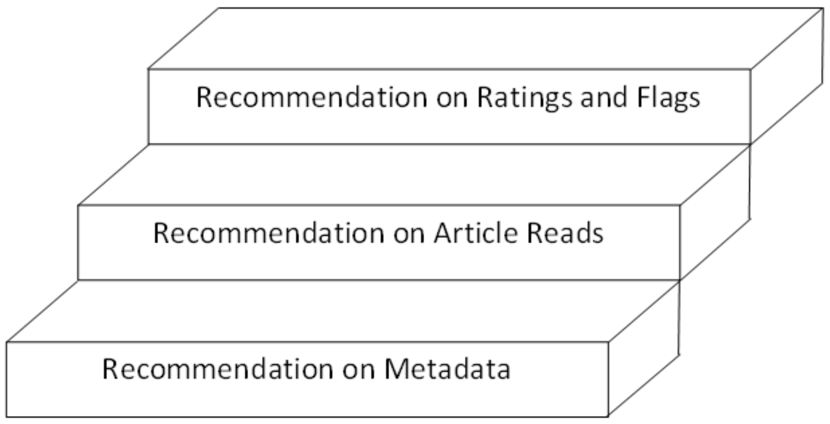

Recommendation on Ratings and Flags
Recommendation on Article Reads
Recommendation on Metadata

Figure 5.1: Three Steps Recommendation

The more data available about the user, the more specific the recommendation will be.

5.5 Defining the Algorithms

This section shows which algorithms for the three steps are used based on the information that is provided.

5.5.1 First Step: Metadata

Filtering based on Metadata is usually handled by machine learning techniques such as Bayesian Classifiers, decision trees or neural networks. However, therefore information about the taste for at least one item is again necessary. Since this step should handle the Cold-Start- and Sparsity-Problem it has to recommend without any further knowledge than the basic user profile. As users share the same property as the items in the system, they both will be treated equally. The User-Profile contains the current rotation, qualification and up to three interests, which are the same as the categories of the articles. In addition, there are topics and tags which further describe an article. The best fitting algorithm for this similarity computation is the Tanimoto coefficient, described in subsection 4.4.5. The reasons why this similarity metric was chosen are:

- The data is only boolean
- Can always compute, even if no property matches
- Each user and item is compared with all its information, not only with information they both share.

A clustering algorithm would also have been able to handle this step, but because of the fact that a whole new computation is necessary when an article is added, this option needed to be eliminated.

The third reason and its underlying fact guards against the problem, that a user would be perfectly matched to an item, just because an item has a huge amount of tags. This problem would occur with every algorithm that just computes on matching preferences.

5.5.2 Second Step: Article Reads

The second step will be handled by a Collaborative Filtering algorithm. The available information, abstract show or article viewed/downloaded, will be represented as values one and two. That means algorithms which are most likely to handle boolean data can already be omitted. Also the Spearman correlation is not suitable since a relative rank on a value difference of two is meaningless. That means the candidates for this step's similarity computation are

- Pearson similarity,
- Euclidean distance and
- Cosine measure.

All three algorithms will be integrated and evaluated to find out which one fits best.

5.5.3 Third Step: Ratings and Flags

As in the second step also the third step is processed by CF. But since there are five parameters left, it needs to be decided which ones are used and how they contribute. The facts that are stored are shown and explained in the subsection 5.1.3. The two parameters that absolutely need to be used are rating and like/dislike. As a third parameter favorite was chosen. Bookmark and share have no real statement about if a user likes or dislikes an article. This information is more or less already covered with the second step. The various information is merged before the recommendation process. The rating is the first information that is taken with its original rating. Afterwards like/dislike is merged, with a five for like and a one for dislike, for the articles that do not have a rating already from the same user. At least the list of favorite items is taken and added with a rating of five. The algorithms which are used to process this data are

- Pearson similarity,
- Spearman correlation,
- Euclidean distance and
- Cosine measure.

5.6 Merging of the three Steps

The merging of the three steps will be implemented and evaluated in four different ways. First of all, each step calculates its recommendation for the particular user. This list will contain, as far as it is possible, double the amount of items that need to be recommended.

5.6.1 Weighted Merge

For a weighted merge all results have to be scaled in the same range first. Since ratings have the highest range, the other two recommendations will be scaled from zero to five. The next step is the definition of the weighting coefficients. For a hybrid system with two different algorithms several papers and books with advices for weighting percentages can be found. However, also these references differ for every use-case and algorithms used. As an initial weight 50%/30%/20% will be used for step three, two and one. If a recommender has no prediction on an article, it is assumed to be neutral and a value of three is assigned.

$$\begin{split} Step1\_rank(3) \times Weight1(0.2) + Step2\_rank(4) \times Weight2(0.3) + \\ Strep3\_rank(4) \times Weight3(0.5) = Weighted\_rank(3.8) \end{split} \tag{5.1}$$

5.6.2 Mixed Merge

Also for this merge technique the ratings of the different steps will be scaled to a range from zero to five. Afterwards the lists are merged together through an addition of the recommendation scores of the three steps.

$$Step1\_rank(3) + Step2\_rank(4) + Strep3\_rank(4) = Mixed\_rank(11) \tag{5.2}$$

Step3_pos(1)
Step2_pos(1)
Step1_pos(1)
Step3_pos(2)
Step2_pos(2)
Step1_pos(2)
Step3_pos(3)
Step2_pos(3)
....
Step1_pos(n)

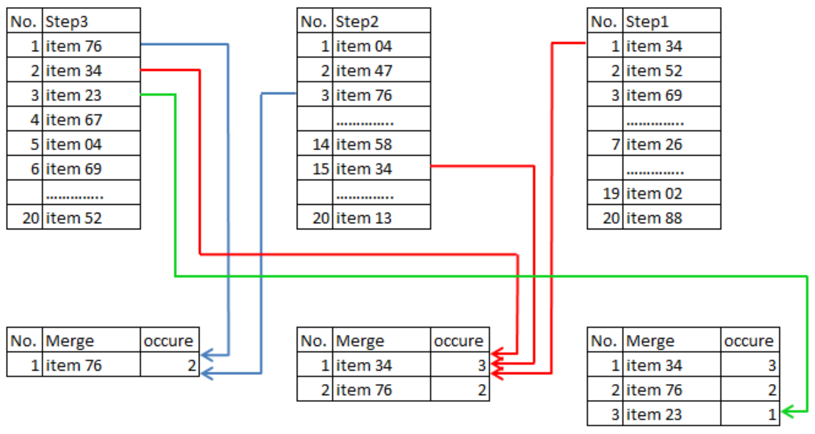
Table 5.2: Recommendation after Top-n Merge

5.6.3 Top-n Merge

This merge technique takes a fixed amount of recommendations from each step into account, which are put together in a round in one list depending on their prior position as shown in table 5.2

5.6.4 Occurrence Merge

After calculating twice as many predictions than recommendations are needed, it is looked up if one item occurs in more than one step. Starting from the highest step it is compared if an item occurs more than once and is written with its occurrence value on the merge table. If the next item now has a higher occurrence it is pinned over the first item with lower value. If it has a lower value it is pinned at the end. This procedure is illustrated in figure 5.2.



No.	Step3
1	item 76
2	item 34
3	item 23
4	item 67
5	item 04
6	item 69
	....
20	item 52

No.	Step2
1	item 04
2	item 47
3	item 76
	....
14	item 58
15	item 34
	....
20	item 13

No.	Step1
1	item 34
2	item 52
3	item 69
	....
7	item 26
	....
19	item 02
20	item 88

No.	Merge	occure
1	item 76	2

No.	Merge	occure
1	item 34	3
2	item 76	2

No.	Merge	occure
1	item 34	3
2	item 76	2
3	item 23	1

Figure 5.2: Imaging of Occurrence Merging

6 Implementation

This chapter illustrates the implementation of the recommendation system conceptualized in chapter 5. First of all it treats the setup of the development environment of Apache Mahout, followed by the implementation of the recommender and webservice and finally the integration into the ColdFusion server.

6.1 UserID Conversion

As mentioned in section 5.1, the UserID contains numbers and letter. Since the Mahout Engine can only work with Long-Numbers, they needed to be converted first. Therefore the Mahout `MemoryIDMigrator` class was used. To translate the LongID back to the original UserID a mapping would be necessary. Since such a back conversion is not needed, this point is skipped. Here are two examples for a converted ID.

- ap291 : -6324286982661262164
- ze136 : 3936981354976206448

The fact, that all IDs are translated to a 19 digit positive or negative number, makes it easy to differ, which of the return results of step one are users and which ones are items.

6.2 Business Data

To minimize the data transfer between the recommender and the database, all in section 5.1 listed business data is downloaded to comma separated files in the format needed for Mahout‘s FileDataModel. The data that is called to separate files from the database includes the following information:

1. UserID, ItemID, Rating
2. UserID, ItemID, Liked/Disliked
3. UserID, ItemID, Favourite
4. UserID, ItemID, Bookmark
5. UserID, ItemID, Shared
6. UserID, ItemID, ArticleRead
7. UserID/ItemID, TagID/TopicID, 1

The first three datafiles are used for the collaborative filtering of step three. Number four and five only give a very little statement if a user has a preference for the item. This information will be used during the evaluation process or later advancements to gain more intelligence. The sixth datafile contains the information needed for the CF of the second step. This data contains the "read" of an abstract as a one and the view/download of the article as a two. The last file contains all users and items and their tags and topics for the similarity measure of the first basic step. The class diagram for `dbDataDownload` is illustrated in figure 6.1.

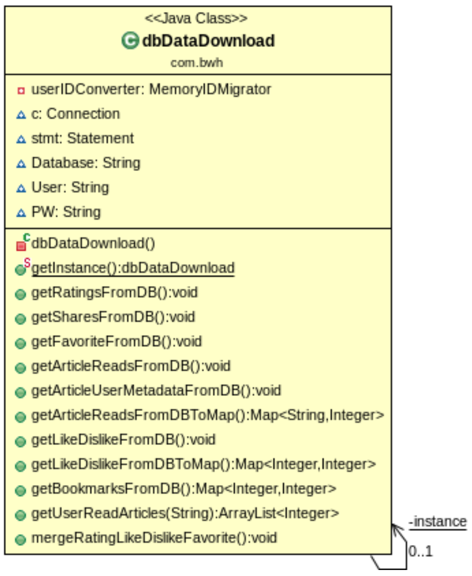

<<Java Class>>
dbDataDownload
com.bwh
userIDConverter: MemoryIDMigrator
c: Connection
stmt: Statement
Database: String
User: String
PW: String
dbDataDownload()
getInstance():dbDataDownload
getRatingsFromDB():void
getSharesFromDB():void
getFavoriteFromDB():void
getArticleReadsFromDB():void
getArticleUserMetadataFromDB():void
getArticleReadsFromDBToMap():Map<String,Integer>
getLikeDislikeFromDB():void
getLikeDislikeFromDBToMap():Map<Integer,Integer>
getBookmarksFromDB():Map<Integer,Integer>
getUserReadArticles(String):ArrayList<Integer>
mergeRatingLikeDislikeFavorite():void
-instance
0..1

Figure 6.1: Class Diagram of `dbDataDownload`

The data frequently updates through a http-call `update`. Mahout reorganizes its data automatically as soon as it determines a change in the data-file.

6.3 General program structure

The general structure of the recommender is kept as modular as possible. The class `recommend` is implemented as a singleton and represents the controller. It manages the programme flow and serves the `recommenderServlets` class, which in this case is the view.

The full class diagram can be seen in the appendix in figure 9.1.

6.4 Implementation of the three Steps

This section gives an overview of the implementation of the three steps, as defined in section 5.5. Every step was implemented in a separate class, that reads its configuration from the file while constructed. The class diagrams of the three recommenders can be seen in figure 6.2.

Since the `Recommender` Class was implemented for easier handling, several unused functions

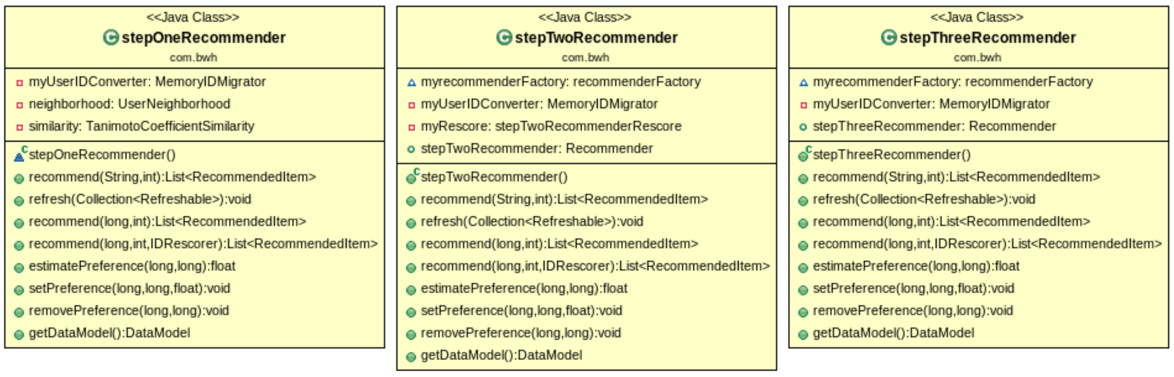

<<Java Class>>
stepOneRecommender
com.bwh
myUserIDConverter: MemoryIDMigrator
neighborhood: UserNeighborhood
similarity: TanimotoCoefficientSimilarity
stepOneRecommender()
recommend(String,int):List<RecommendedItem>
refresh(Collection<Refreshable>):void
recommend(long,int):List<RecommendedItem>
recommend(long,int,IDRescorer):List<RecommendedItem>
estimatePreference(long,long):float
setPreference(long,long,float):void
removePreference(long,long):void
getDataModel():DataModel
<<Java Class>>
stepTwoRecommender
com.bwh
myrecommenderFactory: recommenderFactory
myUserIDConverter: MemoryIDMigrator
myRescore: stepTwoRecommenderRescore
stepTwoRecommender: Recommender
stepTwoRecommender()
recommend(String,int):List<RecommendedItem>
refresh(Collection<Refreshable>):void
recommend(long,int):List<RecommendedItem>
recommend(long,int,IDRescorer):List<RecommendedItem>
estimatePreference(long,long):float
setPreference(long,long,float):void
removePreference(long,long):void
getDataModel():DataModel
<<Java Class>>
stepThreeRecommender
com.bwh
myrecommenderFactory: recommenderFactory
myUserIDConverter: MemoryIDMigrator
stepThreeRecommender: Recommender
stepThreeRecommender()
recommend(String,int):List<RecommendedItem>
refresh(Collection<Refreshable>):void
recommend(long,int):List<RecommendedItem>
recommend(long,int,IDRescorer):List<RecommendedItem>
estimatePreference(long,long):float
setPreference(long,long,float):void
removePreference(long,long):void
getDataModel():DataModel

Figure 6.2: Class Diagram of the three recommenders

are present. These untapped functions are `refresh`, `estimatePreference`, `setPreference` and `removePreference`. The underlying datasets that are used by the different recommenders were explained in the planning phase in section 5.5.

6.4.1 Realization of the first Step

Step one has a very simple implementation because of the Apache Mahout framework. After generation the `FileDataModel` and handing it over to the `TanimotoCoefficientSimilarity` only the neighbourhood needs to be defined. Since after the computation a filtering of the return values is necessary a neighbourhood of 100 was chosen. After deleting the other users in the `Long`-Array and the rescore to a range between zero and five, the predictions are ready for the merge process.

6.4.2 Realization of the second Step

In its first part step two is very similar to the first step, but here it is only a precomputation for the Collaborative Filtering process. All similarity metrics mentioned in subsection 4.4 were implemented and can be selected through the configuration file. The outcome of this step is a value between zero and two, since the input data is in the same range. The recommenders are created from a factory method, that produces a recommender by its given parameters. This way also temporary recommenders for evaluation can be created, which is necessary for testing and evaluation beside the productive use of the system. For the actual use of the system, the configuration is loaded from the config file. For the rescore of this step a `IDRescorer` class, that can be seen in figure 6.3, was created to recomputes the predicted value. Therefore the `rescore` function, that in some cases can be very complex, is called to return the new value.

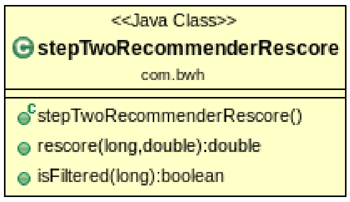

<<Java Class>>
stepTwoRecommenderRescore
com.bwh
stepTwoRecommenderRescore()
rescore(long,double):double
isFiltered(long):boolean

Figure 6.3: Class Diagram of `stepTwoRecommenderRescore`

6.4.3 Realization of the third Step

The third step only uses the basic recommender without rescore, since the predictions are already in the defined range. Like in step two, also in three, all similarity metrics mentioned in subsection 4.4 can be selected through the configuration file. The creation of the recommender is done by the in figure 6.3 shown `stepTwoRecommenderRescore` class. Therefore the `getNewRecommender` function is fed with

- `basedOn` - user or item,
- `similarityMetric` - pearson, euclidean, spearman, tanimoto or cosine,
- `neighborhoodQuantity` - neighborhood quantity of similar users and
- `DataModel` - fileDataModel the recommender recommends on.

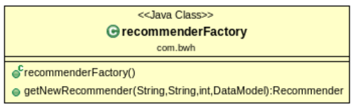


Figure 6.4: Class Diagram of `recommenderFactory`

6.5 Merge of the three Steps

The merge algorithms that were described in section 5.6 are implemented in a separate class. The class diagram is shown in figure 6.5.

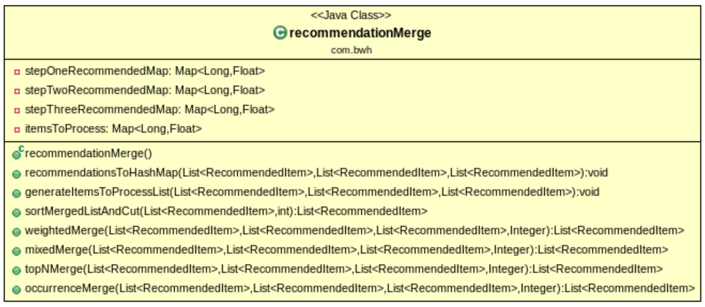

<<Java Class>>
recommendationMerge
com.bwh
stepOneRecommendedMap: Map<Long,Float>
stepTwoRecommendedMap: Map<Long,Float>
stepThreeRecommendedMap: Map<Long,Float>
itemsToProcess: Map<Long,Float>
recommendationMerge()
recommendationsToHashMap(List<RecommendedItem>,List<RecommendedItem>,List<RecommendedItem>):void
generateItemsToProcessList(List<RecommendedItem>,List<RecommendedItem>,List<RecommendedItem>):void
sortMergedListAndCut(List<RecommendedItem>,int):List<RecommendedItem>
weightedMerge(List<RecommendedItem>,List<RecommendedItem>,List<RecommendedItem>,Integer):List<RecommendedItem>
mixedMerge(List<RecommendedItem>,List<RecommendedItem>,List<RecommendedItem>,Integer):List<RecommendedItem>
topNMerge(List<RecommendedItem>,List<RecommendedItem>,List<RecommendedItem>,Integer):List<RecommendedItem>
occurrenceMerge(List<RecommendedItem>,List<RecommendedItem>,List<RecommendedItem>,Integer):List<RecommendedItem>

Figure 6.5: Class Diagram of `recommendationMerge`

Beside the different merge algorithms, also three helping functions were integrated. For easier handling, `recommendationsToHashMap` converts the list of `RecommendedItem` to a `HashMap` so that every item can be called by its ID. As the name `generateItemsToProcessList` suggests, this function is generating a `HashMap` with all items to process out of the three steps. This is needed since one item can occur up to three times, but only should be processed once. The last helping function is `sortMergedListAndCut`, which sorts `RecommendedItem` based on their value and cuts the `ArrayList` afterwards to the length defined by `howMany`. For the sorting a class that implements `Comparator` was generated, which compares the value of the recommended items for `java.util.Collections.sort()`.

7 Results

This chapter shows the results of the recommendation engine. First the available data will be analysed, before the execution is explained and the data shown. The last section of this chapter then deals with the interpretation of the results.

7.1 Available Data

During the answering of the five questions for a basic vision of the recommendation system in section 5.3, the data was collected for the first time. After almost four months the collected data grew to the following values:

- 48 ratings by 20 different users
- 25 likes and 12 dislikes by 24 different users
- 2 favorites by 2 different users
- 396 article reads by 56 different users

Beside these values the number of categories raised to 14 with a total of 166 subcategories. The quantity of tags grew to 77. On the basis of data that is associated with the third step no evaluation is possible. Also the data from article reads is very limited since more than a third of the reads are from one user. 14 of the users only have read one article, only seven users have read more than ten articles and are likely to get predictions. After computing all users that have more than five article reads, only user ps939 got two predictions. At the moment an evaluation of the whole system is therefore not possible. The adjustment of the CF algorithms for step two and three is no doable due to the same cause. For this reason this chapter will deal with the analysis of the first step and the peculiarities of the tanimoto similarity.

7.2 Test Setup and System

To understand the behavior of the tanimoto coefficient similarity for a basic recommendation five datasets with two to ten matches are generated and computed. Afterwards the response will be analysed by adding mismatches to several already matching items. Finally we will take a look at the computation time of datasets with different sizes and number of attributes.

All tests are performed on a notebook with an i7-3687U, a two core 2.1 GHz CPU with Hyper-threading, and 8 Gb of memory. The host as well as the virtual system are running on a 64 bit architecture. The testdata that is used for the performance test of the first step is self-generated and contains between two and twelve preferences per user and item.

7.3 Results and Interpretation

Figure 7.1 illustrates the response of the tanimoto coefficent similarity on various quantities of matches. Each line, standing for a different amount of attributes an items has, proceeds linearly when overlapping with a user property.

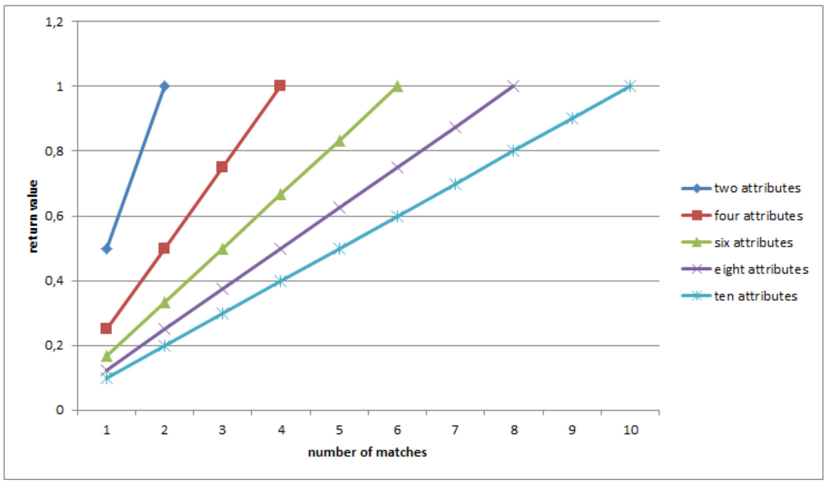


Figure 7.1: Response of tanimoto coefficent similarity on match

Figure 7.2, by contrast, shows a non-linear response on the mismatch of an item. Every line stands for a count of matching articles. The trend shows then the return value, which accrues when attributes that do not concur are added.

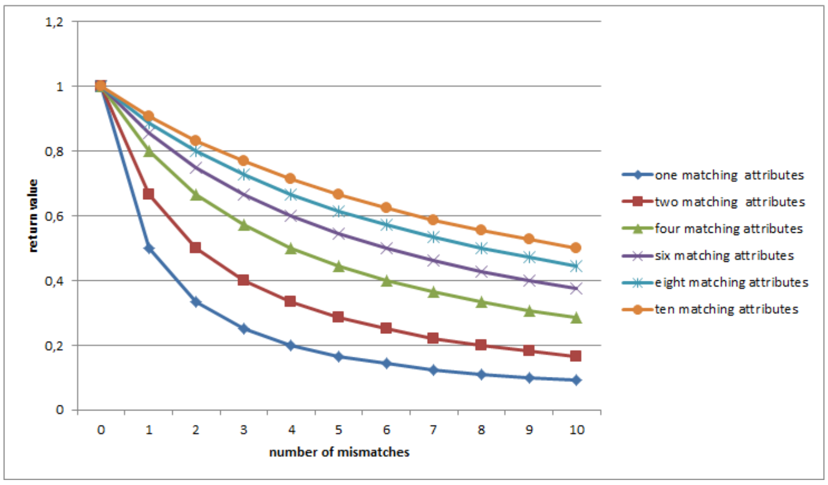


Figure 7.2: Response of tanimoto coefficent similarity on mismatch

It is shown that the return value rises quickly on a match with an article with a small amount of attributes. By contrast, overloading of an item leads to a lower response as desired, since the specificity drops. As expected the tanimoto coefficent similarity meets exactly the requirements for the basic recommendation on the first step. This behavior is also wanted for the match of cases with the user's attributes to articles, videos and voice records.

The self generated data sets have a size of 20.000, 100.000 and 1.000.000 entries that are distributed as mentioned in section 7.2. It was found out that the computation time is independent from the size of the dataset. Figure 7.3 illustrates the duration of the calculation for two to 24 parameters. How these parameters are distributed between items and users does not matter. From the 300 ms that were defined as a success this step does not even take 10 % of the maximum computation time.

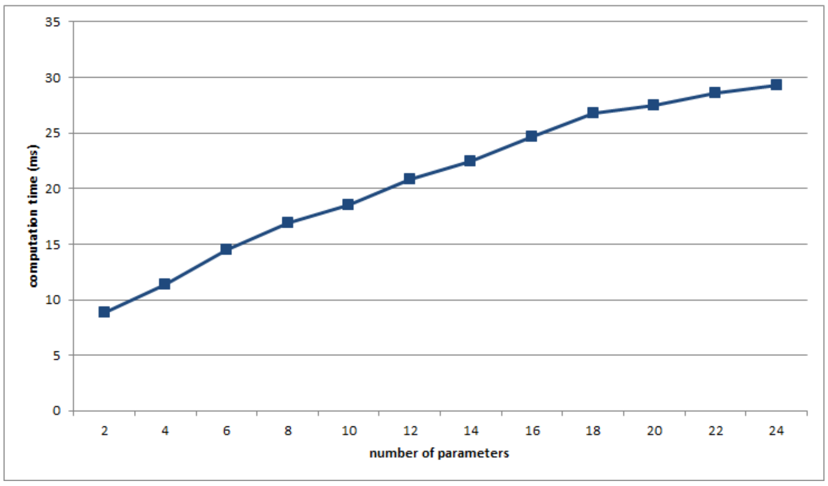


Figure 7.3: Computation time of tanimoto coefficent similarity

8 Summary

This thesis illustrates the usage of a multi-part-hybrid recommendation system on the basis of the business data and specialities of the Brigham and Women's hospital. It has been shown that through structured breakdown of the available information, recommendations to a variety of different users is possible, as well as the bypass of an existing cold-start and sparsity problem. Since there are many different programmes and frameworks that are suitable for this task, external circumstances determine the fitting to a large extent. The algorithms behind the various stages can be varied according to each information type and easily adjusted with the appropriate framework. Therefore it is useful to first optimize the individual steps, before the merge takes place in an overall system. Mahout proved to be very efficient due to it's modular structure and because of its already implemented parts. This resulted in a rapid prototype, which was further developed step by step to the final system.

8.1 Conclusion

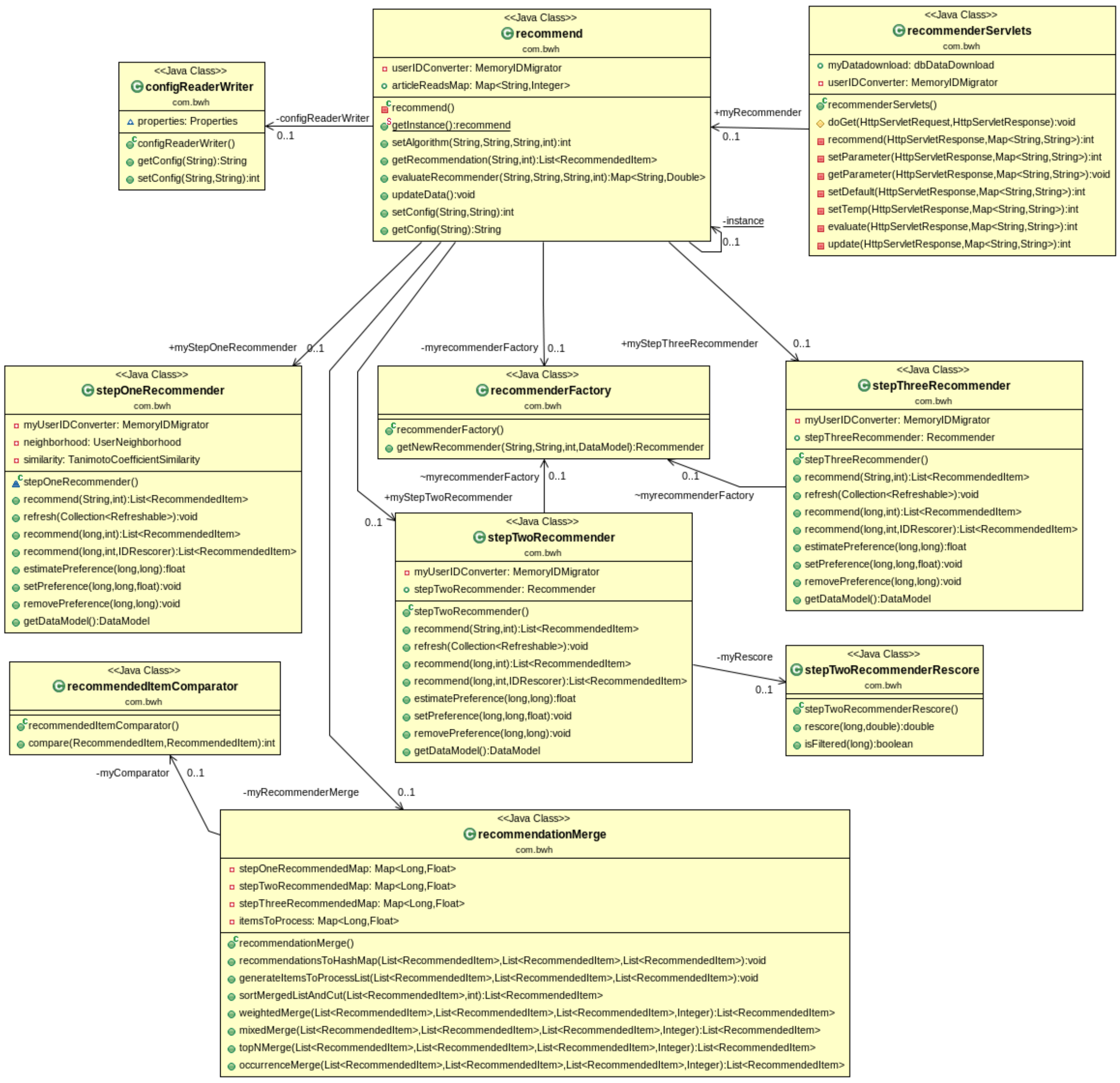
The analysis of the available data, over the evaluation of several different frameworks and recommendation algorithms led to the implementation of a solution, that fits the requirements of the anaesthesia department of the Brigham and Women's hospital. The several peculiarities this department has brought with it, could be considered and used for a scalable recommendation system. Unfortunately it was not possible to test the system in all its complexity due to the lack of data. The use of the tanimoto similarity metric for the recommendation on metadata however seems to be a complete success. For later evaluation and configuration various precautions have been taken to make this work as easy as possible. It was also made sure, that functions and classes in the future can take over other tasks that they were not specifically designed for. Because of its modular implementation the system additionally offers several possibilities for extension and alternations to also fit future requirements of the department.

8.2 Outlook

First of all the acquisition of data will be the main focus before any further work can be done. The evaluation and configuration of the similarity metrics for the collaborative filtering is the primary goal. Therefore a concept for a rewarding system exists, as well as the code for the computation on the ColdFusion server. Only the integration at the front end and publication needs to be done, of which an improvement in the willingness of reviews is expected. Afterwards the different merge algorithms can be tested on their performance and results. It is also planned that the third set of the recommendation system will be split up in three different independent computations for the parameters `rating`, `like/dislike` and `favorite`. The merge algorithms will be reused for this enhancement. For the extension of ICP/ePAL it is planned that users can directly jump to recommended items from cases they are working on. Therefore an additional call of the recommendation system will be implemented, which adds a temporary user, based on case and user data, to find matching articles. Since a database with exam questions and answers will be integrated into the portal, this would also be a possibility to recommend questions on the current rotation and therefore the focus of the test. A documentation of the answers can furthermore help the improvement by repeating questions that were answered incorrectly. This would be a useful addition to a learning portal.

9 Appendix

9.1 Class Diagram of the whole Recommender



<<Java Class>>
recommend
com.bwh
userIDConverter: MemoryIDMigrator
articleReadsMap: Map<String,Integer>
recommend()
getInstance():recommend
setAlgorithm(String,String,String,int):int
getRecommendation(String,int):List<RecommendedItem>
evaluateRecommender(String,String,String,int):Map<String,Double>
updateData():void
setConfig(String,String):int
getConfig(String):String
<<Java Class>>
recommenderServlets
com.bwh
myDatadownload: dbDataDownload
userIDConverter: MemoryIDMigrator
recommenderServlets()
doGet(HttpServletRequest,HttpServletResponse):void
recommend(HttpServletResponse,Map<String,String>):int
setParameter(HttpServletResponse,Map<String,String>):int
getParameter(HttpServletResponse,Map<String,String>):void
setDefault(HttpServletResponse,Map<String,String>):int
setTemp(HttpServletResponse,Map<String,String>):int
evaluate(HttpServletResponse,Map<String,String>):int
update(HttpServletResponse,Map<String,String>):int
<<Java Class>>
configReaderWriter
com.bwh
properties: Properties
configReaderWriter()
getConfig(String):String
setConfig(String,String):int
-configReaderWriter
0..1
+myRecommender
0..1
-instance
0..1
+myStepOneRecommender
0..1
-myrecommenderFactory
0..1
+myStepThreeRecommender
0..1
<<Java Class>>
stepOneRecommender
com.bwh
myUserIDConverter: MemoryIDMigrator
neighborhood: UserNeighborhood
similarity: TanimotoCoefficientSimilarity
stepOneRecommender()
recommend(String,int):List<RecommendedItem>
refresh(Collection<Refreshable>):void
recommend(long,int):List<RecommendedItem>
recommend(long,int,IDRescorer):List<RecommendedItem>
estimatePreference(long,long):float
setPreference(long,long,float):void
removePreference(long,long):void
getDataModel():DataModel
<<Java Class>>
recommenderFactory
com.bwh
recommenderFactory()
getNewRecommender(String,String,int,DataModel):Recommender
~myrecommenderFactory
0..1
+myStepTwoRecommender
0..1
~myrecommenderFactory
0..1
<<Java Class>>
stepThreeRecommender
com.bwh
myUserIDConverter: MemoryIDMigrator
stepThreeRecommender: Recommender
stepThreeRecommender()
recommend(String,int):List<RecommendedItem>
refresh(Collection<Refreshable>):void
recommend(long,int):List<RecommendedItem>
recommend(long,int,IDRescorer):List<RecommendedItem>
estimatePreference(long,long):float
setPreference(long,long,float):void
removePreference(long,long):void
getDataModel():DataModel
<<Java Class>>
stepTwoRecommender
com.bwh
myUserIDConverter: MemoryIDMigrator
stepTwoRecommender: Recommender
stepTwoRecommender()
recommend(String,int):List<RecommendedItem>
refresh(Collection<Refreshable>):void
recommend(long,int):List<RecommendedItem>
recommend(long,int,IDRescorer):List<RecommendedItem>
estimatePreference(long,long):float
setPreference(long,long,float):void
removePreference(long,long):void
getDataModel():DataModel
-myRescore
0..1
<<Java Class>>
stepTwoRecommenderRescore
com.bwh
stepTwoRecommenderRescore()
rescore(long,double):double
isFiltered(long):boolean
<<Java Class>>
recommendedItemComparator
com.bwh
recommendedItemComparator()
compare(RecommendedItem,RecommendedItem):int
-myComparator
0..1
-myRecommenderMerge
0..1
<<Java Class>>
recommendationMerge
com.bwh
stepOneRecommendedMap: Map<Long,Float>
stepTwoRecommendedMap: Map<Long,Float>
stepThreeRecommendedMap: Map<Long,Float>
itemsToProcess: Map<Long,Float>
recommendationMerge()
recommendationsToHashMap(List<RecommendedItem>,List<RecommendedItem>,List<RecommendedItem>):void
generateItemsToProcessList(List<RecommendedItem>,List<RecommendedItem>,List<RecommendedItem>):void
sortMergedListAndCut(List<RecommendedItem>,int):List<RecommendedItem>
weightedMerge(List<RecommendedItem>,List<RecommendedItem>,List<RecommendedItem>,Integer):List<RecommendedItem>
mixedMerge(List<RecommendedItem>,List<RecommendedItem>,List<RecommendedItem>,Integer):List<RecommendedItem>
topNMerge(List<RecommendedItem>,List<RecommendedItem>,List<RecommendedItem>,Integer):List<RecommendedItem>
occurrenceMerge(List<RecommendedItem>,List<RecommendedItem>,List<RecommendedItem>,Integer):List<RecommendedItem>

Figure 9.1: Class diagram of the whole recommender

Literaturverzeichnis

[1] Adobe Systems Incorporated. ADOBE® COLDFUSION?8 ColdFusion Developer's Guide. http://livedocs.adobe.com/coldfusion/8/cf8_devguide.pdf. [Online; accessed 05-May-2014].

[2] Asela Gunawardana, Christopher Meek. Tied boltzmann machines for cold start recommendations, 2008.

[3] Machine Learning Group at the University of Waikato. Machine learning group at the university of waikato. http://www.cs.waikato.ac.nz/ml/weka/, 2012. [Online; accessed 06-June-2014].

[4] Chumki Basu, William W. Cohen, Haym Hirsh, and Craig G. Nevill-Manning. Technical paper recommendation: A study in combining multiple information sources. *CoRR*, abs/1106.0248, 2011.

[5] Brigham and Women's Hospital. Brigham and Women's Hospital. http://http://www.brighamandwomens.org/, 2013. [Online; accessed 09-June-2014].

[6] Yibo Chen, Chanle Wu, Ming Xie, and Xiaojun Guo. Solving the sparsity problem in recommender systems using association retrieval. *JCP*, 6:1896–1902, 2011. [Online; accessed 16-May-2014].

[7] Dhoha Almazro, Ghadeer Shahatah, Lamia Albdulkarim, Mona Kherees, Romy Martinez, William Nzoukou. A Survey Paper on Recommender Systems, 2010.

[8] Dhruba Borthakur. HDFS Architecture Guide, 2014. [Online; accessed 05-May-2014].

[9] Facebook. Facebook. https://www.facebook.com/, 2014. [Online; accessed 27-June-2014].

[10] Nathaniel Good, J. Ben Schafer, Joseph A. Konstan, Al Borchers, Badrul Sarwar, Jon Herlocker, and John Riedl. Combining collaborative filtering with personal agents for better recommendations. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence and the Eleventh Innovative Applications of Artificial Intelligence Conference Innovative Applications of Artificial Intelligence*, pages 439–446. American Association for Artificial Intelligence, 1999.

[11] Will Hill, Larry Stead, Mark Rosenstein, and George Furnas. Recommending and evaluating choices in a virtual community of use. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 194–201. ACM Press/Addison-Wesley

Publishing Co., 1995.

[12] Netflix Inc. Netflix prize. http://www.netflixprize.com/, 2014. [Online; accessed 27-June-2014].

[13] J.M. Hook, E. Grant, A. Samarth. Health Information Technology and Health Information Exchange Implementation in Rural and Underserved Areas: Findings from the AHRQ Health IT Portfolio, 2010.

[14] John S. Breese, David Heckerman, Carl Kadie. Empirical analysis of predictive algorithms for. collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 43–52. Morgan Kaufmann Publishers Inc., 1998.

[15] Jonas Partner, Aleksa Vukotic, Nicki Watt. *Neo4j in Action*. Manning Publications, 2014.

[16] Joseph Tan. E-Health Care Information Systems: An Introduction for Students and Professionals, 2005.

[17] Taghi M. Khoshgoftaar, Kehan Gao, and Nawal H. Ibrahim. Evaluating indirect and direct classification techniques for network intrusion detection. *Intell. Data Anal.*, pages 309–326, 2005.

[18] Henry Lieberman. Letizia: An agent that assists web browsing. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 1*, pages 924–929. Morgan Kaufmann Publishers Inc., 1995.

[19] Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, pages 76–80, 2003.

[20] Marcel Caraciolo, Bruno Melo, Ricardo Caspirro. Crab: A Recommendation Engine Framework for Python, 2011.

[21] The NASDAQ Stock Market. The nasdaq stock market. http://www.nasdaq.com/, 2014. [Online; accessed 27-June-2014].

[22] Bradley N. Miller, Istvan Albert, Shyong K. Lam, Joseph A. Konstan, and John Riedl. Movielens unplugged: Experiences with an occasionally connected recommender system. In *Proceedings of the 8th International Conference on Intelligent User Interfaces*, pages 263–266. ACM, 2003.

[23] Justin J. Miller. Graph database applications and concepts with neo4j. Proceedings of the Southern Association for Information Systems Conference, 2013.

[24] Bamshad Mobasher, Xin Jin, and Yanzan Zhou. Semantically enhanced collaborative filtering on the web. *Web Mining: From Web to Semantic Web*, 3209:57–76, 2004.

[25] Pankaj Sarin, Andreas Plank. Conference Protocol 3 April 2014.

[26] Pankaj Sarin, Rodney Gabriel. Integrated clinical portal and electronic portal for anesthesia learning. `https://icp.bwhanesthesia.org`, 2014. [Online; accessed 28-May-2014].

[27] Michael J. Pazzani and Daniel Billsus. Content-based recommendation systems. In *The Adaptive Web: Methods and Strategies of Web Personalization Series: Lecture Notes in Computer Science, Vol. 4321*, pages 325–341. Springer-Verlag, 2007.

[28] Prem Melville, Vikas Sindhwani. Recommender Systems, 2010.

[29] Research Studios Austria Forschungsgesellschaft mbH. easyrec, 2014. [Online; accessed 06-May-2014].

[30] Roy Sutton. *Enyo: Up and Running.* O'Reilly & Associates, 2013.

[31] Sean Owen, Robin Anil, Ted Dunning, Ellen Friedman. *Mhout in Action.* Manning Publications, 2011.

[32] Upendra Shardanand and Pattie Maes. Social information filtering: Algorithms for automating &ldquo;word of mouth&rdquo;. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 210–217. ACM Press/Addison-Wesley Publishing Co., 1995.

[33] Jack Smith. *Health Management Information Systems: A Handbook for Decision Makers.* Open University Press, 1st edition, 1999.

[34] Sung-Hyuk Cha. Comprehensive Survey on Distance/Similarity Measures between Probability Density Functions, 2007.

[35] Joseph Tan and Fay Cobb Payton. Adaptive Health Management Information Systems: Concepts, Cases, & Practical Applications, 2009.

[36] The Apache Software Foundation. Apache Solr, 2014. [Online; accessed 06-June-2014].

[37] Tong Zhang, Vijay S. Iyengar. Recommender systems using linear classifiers. *The Journal of Machine Learning Research 2*, pages 313–334, 2002.

[38] Tray Grainger, Timothy Potter. *Solr in Action.* Manning Publications, 2014.