Department of Electrical Engineering and Information Technology
Institute for Media Technology
Distributed Multimodal Information Processing Group
Prof. Dr.-Ing. Eckehard Steinbach

Evaluating Socially Connected E-Learning

Auswertung der E-Learning-Nutzung im sozialen Verbund

Christina Obolashvili

Diploma Thesis

Author:	Christina Obolashvili
Address:	[redacted]
Matriculation Number:	[redacted]
Professor:	Prof. Dr.-Ing. Eckehard Steinbach
Advisor:	Prof. Dr. Matthias Kranz
	Dipl.-Medieninf. Andreas Möller
Begin:	01.10.2012
End:	28.03.2013

Department of Electrical Engineering and Information Technology
Institute for Media Technology
Distributed Multimodal Information Processing Group
Prof. Dr.-Ing. Eckehard Steinbach

Declaration

I declare under penalty of perjury that I wrote this Diploma Thesis entitled

Evaluating Socially Connected E-Learning
Auswertung der E-Learning-Nutzung im sozialen Verbund

by myself and that I used no other than the specified sources and tools.

Munich, March 28, 2013

Christina Obolashvili

Christina Obolashvili
[redacted]

Kurzfassung

In dieser Arbeit geht es um die Weiterentwicklung des Projektes MobiDics, einer E-learning Applikation für Lehrende. MobiDics erlaubt es den Teilnehmern ihr Wissen untereinander zu teilen und dadurch die Lernqualität zu verbessern. In den früheren Arbeiten wurden die Web- und Mobile Applikationen entwickelt. Im Rahmen dieser Arbeit wurde eine Studie durchgeführt um die Benutzerfreundlichkeit von MobiDics und das Gesamt-Nutzungserlebnis zu evaluieren.

Die Ergebnisse der Studie zeigen, dass die Nutzer MobiDics als benutzerfreundlich einstufen. Darüber hinaus, zeigt sich bei den Teilnehmern eine signifikante Steigerung in Bezug auf die Selbstsicherheit bei der Anwendung einer didaktischen Methode, sowie bei dem Wissen um die passende didaktische Methode für verschiedene Situationen.

Im zweiten Teil der Arbeit wurde MobiDics um ein Empfehlungssystem erweitert. Ein Empfehlunssystem sollte den Nutzern für sie interessante didaktische Methoden empfehlen. Für diesen Zweck wurden zwei Ansätze von Empfehlungsalgorithmen kombiniert. Bei inhaltsbezogenen Empfehlungssystemen werden die Vorlieben der Nutzer berücksichtigt und so die passenden didaktischen Methoden ausgewählt. Der Nutzer hat dabei die Möglichkeit die für sie relevanten Kriterien unterschiedlich zu gewichten, um die Resultate individuell anzupassen.

Das zweite Verfahren verwendet kollaboratives Filtern um Empfehlungen auf der Basis von Bewertungen anderer Nutzer zu generieren. Im Rahmen dieser Arbeit wurde dafür der sogenannte Slope-One Algorithmus eingesetzt. Um die begrenzten Resourcen der mobilen Applikation zu schonen, wurde der rechenintensive Teil, d.h. die Erzeugung der "Deviation Matrix" auf den Server verlagert.

Abschließend werden die Ergebnisse der zwei Verfahren kombiniert (ein sogenanntes "gewichtetes hybrides Empfehlungssystem"), um die Vorteile beider Systeme zu nutzen und um Schwächen auszugleichen.

Abstract

This thesis is a continuation of the project MobiDics that focuses on developing the e-learning application for teaching personnel in order to support the interchange of their experience and help improve the teaching quality. In the previous steps of this project the web and mobile applications were created. Within this thesis the features of the existing system were evaluated during a user study. The aim of the user study was to investigate the usability of MobiDics and study the user experience.

The results of the user study showed that overall the teaching personnel found MobiDics user friendly and helpful. Additionally, the study showed that participants' self confidence in applying the didactic methods rises significantly with the usage of the application. Moreover, the participants' self-assessed knowledge of didactic methods for different situations has enhanced significantly.

In the second part of the thesis the application was enhanced with a recommendation system. The recommendation system should suggest didactic methods to the users that could be interesting for them. For this purpose two recommendation algorithms were combined. The content-based approach takes the user's preferences into consideration and makes suggestions based on them. The user has the opportunity to change the weights for the different criteria in order to get tailor-made recommendations.

The second algorithm is based on collaborative filtering which takes other users' ratings into account to generate recommendations. In this thesis the so-called Slope-One algorithm was used. To save resources the computationally intensive part, i.e., the creation of the "deviation matrix" was implemented on the server-side.

A weighted combination of the two approaches result in a hybrid recommendation system that benefits from advantages of both approaches.

Contents

Chapter 1.

Introduction

Internet has become an ubiquitous service over the years. We all use it for many reasons, whether it is to check our e-mails, search for information, buy products, engage in social networking or simply browse the World Wide Web. There is hardly a major company or a service provider of any kind that does not have a web site. The growing number of web sites and provided services is hard to manage for the costumer. People easily risk being flooded with information. Users have to continually invest more time to categorize offered information and services to find exactly what they are looking for. This can frustrate users and they might decide to leave the web site and use other web sites that provide a similar service. As this can be a serious economic disadvantage for web site owners, there is a strong incentive to provide a system that supports a user in finding items and information that is relevant to her.

The answer lies in recommendation systems: programmable tools that collect variables from an individual's browsing/purchasing history which then provide tailored suggestions to the user based on their online browsing tendencies [1]. Internet service providers have to continually upgrade their features to draw users' interest to their products. Thus, there is a variety of such systems, each of which uses a different approach. As mobile computing gains popularity, recommendation systems have to be also provided for ubiquitous computing, albeit in a different way due to its limitations (storage capacities, small screen display) [2]. Mobile computing has its benefits as well, most notably knowing the exact location for each user. This information can be beneficial for suggesting items that are close to the user at a given moment [2]. Examples of this include receiving suggestions for the nearest seafood restaurant or the nearest cinema that is showing a certain film.

The intention of this thesis is to both evaluate an existing system, MobiDics ([3], [4]), a didactic tool that supports educators in a university setting, through a four-week user study and to enhance the functionality of MobiDics with a custom-made recommendation system. The focus of the evaluation is the user satisfaction with the MobiDics functionalities. For this purpose the overall usage of MobiDics is being studied to draw conclusions which can help to improve user experience.

The structure of the thesis is as follows:

Chapter 2 reviews the recommendation systems currently in operation as well as a brief description of their different classifications and uses. Chapter 3 introduces MobiDics and its functionality. In Chapters 4 and 5, a recommendation system algorithm is introduced and implemented, respectively.

A user study has been conducted to evaluate user satisfaction with already existing functionality of MobiDics. The results of the study are presented in Chapter 6, along with a discussion based on the findings of the user study. Finally, conclusions are drawn and possible aspects of future work are discussed in Chapter 7.

Chapter 2.

Related work

In this chapter we will describe the common types of recommendation systems, their advantages and disadvantages as well as provide examples of their functionality.

2.1. Recommendation Systems

Recommendation or recommender systems (RSs) have gained popularity over the last few years. More and more web pages other than e-commerce services offer user-specific recommendations. Further, non-commercial web pages use RS techniques to increase user satisfaction. For example, the known social media services such as Twitter, Inc.[1] and Facebook[2] contain features to suggest content that might be interesting for users. Online newspaper editorials, e.g., The Guardian[3] and blogs recommend related articles, which could be relevant for the reader and in turn increase the number of page views.

> "Recommender Systems (RS) are software tools and techniques providing suggestions for items to be of use to a user" [1].

When we speak about recommendations, a common name of an object that is being recommended is an item.

There are several reasons that make RSs interesting and important for service providers, such as selling more items, providing a user with recommendations which make her easily find and purchase diverse items, increase user's satisfaction by using the service, which may ultimately lead to her loyalty towards the web site [1]. On the other hand, the RSs can also be very helpful and important for users, supporting them by personalizing their profiles and wishes.

> "... a RS must balance the need of these two players and offer a service that is valuable to both" [1].

[1]`http://www.twitter.com` (Last visited: 24.03.2013)
[2]`http://www.facebook.com` (Last visited: 24.03.2013)
[3]`http://www.guardian.co.uk` (Last visited: 24.03.2013)

Depending on the service, a RS can facilitate different tasks itself. There are nine tasks that Herlocker et al. define in their paper and that are most cited among the related literature [5]:

- *Find good items:* This is perhaps the most common task of a RS that each one should implement. A RS provides a ranked list of items that could be interesting for a user.
- *Find all good items:* There are cases, where it is important to identify all items that may be interesting for a user. For example, this could particularly be very useful in the field of medicine, e.g., looking for a treatment plan. In this case, a person using the RS is willing to put in the extra time to closely study every item in a ranked list [1].
- *Recommend sequence:* Instead of a single item, recommend a sequence of items. The difference here is that the whole sequence has to be interesting. There is a possibility that some of the items of the sequence will be less important for the user. However, this is less problematic for a suggestion as whole than recommending a single inadequate item.
- *Just browsing:* A user usually will be interested in recommendations while having the intention to purchase a product. Nevertheless, it is possible that some of them will just be interested what a RS has to offer, even if there is no particular item that they are looking for. A RS should be able to help the user view the items that may be interesting for her for the particular session.
- *Find credible recommender:* Service providers have to deal with the idea that some users question the reliability of a recommendation system. Herlocker et el. observed that some users changed their preferences or profiles, just to see if a RS was "clever enough" to notice the changes and make sensible recommendations based on changed profile data information. Providing thoughtful additional services to make a recommendation system trustworthy could be helpful [5]. For example, explaining why the item was recommended to the user could help her see if her changed preferences are adopted by the RS while suggesting the items.
- *Improve profile:* In order to make personalized recommendations, a RS has to have some information about the user. In fact, the more a system knows about a user, the higher the chance of getting an accurate result. For this purpose, a user should have the opportunity to add information about her likes or dislikes, in addition to having the opportunity to rate items. Most of the service providers consider this task as a crucial one. For example, Goodreads [4] asks its users to personalize their profiles by adding a list of their favorite authors and genres. This information can be taken into consideration for making suggestions, meaning, a user is more likely to be interested in books from a favorite genre or from favorite authors.
- *Express self:* During the conducted studies, Herlocker et al. noticed that some of the users

[4] http://www.goodreads.com (Last visited: 24.03.2013)

gave feedback to items "because it felt good", rather than wanting a suggestion themselves [5]. Thus, the authors suggest, that giving a user an opportunity to give feedback can be beneficial, having in mind that this action leads to more data.

- *Help others:* Some users contribute their feedback solely for the reason to help other users to make their choice. For example, the web site "Stack Overflow" [5], where programmers can ask questions and get answers from professional users, depends entirely on such users.
- *Influence others:* Reasons to rate an item and thus contribute feedback sometimes have additional and/or hidden motives. Some users rate a particular item they want to advertise or receive other benefits from doing so, thus distorting the natural ranking of items.

It is obvious that RSs depend on data. To deal with data in a correct way, there has to be some kind of classification, because not every kind of available data should be handled the same way. For this matter, Ricci et al. classify data into the following three categories [1]:

- *Items*: As mentioned above, items are the subject of a recommendation. An item can be a book for a RS, that recommends books, or a song for a RS, that recommends music. Each of these items have their own characteristics, e.g., books have authors, or belong to particular genre. These characteristics can be used for making suggestions. Usually, an item is stored in a database, represented by listing its characteristics and most commonly having an additional identifier [1].

  An item itself can have a positive or a negative value for a user. Typically, an item that is useful for a user, is of a positive value for her and vice versa [1].
- *Users:* Users are an invaluable source of information for every RS. Firstly, they are the reason a RS is needed and secondly, they provide personal data by means of which a RS can make suggestions, whether or not an item is suitable for her [1].

  The behaviour of a certain user may also be considered by a RS. The browsing pattern or a searching pattern of a user are examples of how a RS could benefit from the given information [1].
- *Transactions:* "are log-like data that store important information generated during the human-computer interaction and which are useful for the recommendation generation algorithm that the system is using" [1]. In other words, if a user bought one item during one session, it is valuable to know the characteristics of this item because it is likely to happen, that a user will be interested in suggestions which include similar items [1].

Considering different approaches and goals of a given service, there are several techniques of realizing a RS which will be discussed in the next sections.

[5]http://www.stackoverflow.com (Last visited: 17.03.2013)

2.2. Content-Based Recommendation Systems

Content-based (CB) recommendation systems make suggestions relying only on user profiles and characteristics of items that a user has found interesting before. A good example of content-based filtering is last.fm[6] an online radio station. The web site provides users with a list of songs that may be interesting for her, taking into account songs a user has listened to previously.

In order to make a meaningful suggestion, the characteristics of items are being considered. These can be of a binary, nominal, or a textual value. For example, Meteren et al. use content-based filtering in their personalized recommendation system (PRES) to suggest contents of textual data to users, who are interested in do-it-yourself home improvements [6].

User profiles are also of a high value for RSs. The more a RS knows about the preferences of a user, the better the provided suggestions will get. For this matter, most of the service providers give the user an opportunity to give feedback about an item, e.g., by rating it. Explicit rating can be of a textual type, meaning an actual review for an item, but the most common way of a rating is on some kind of scale ([1], [7]). This can have several effects: It is easier to work with the data represented in scales, such as applying statistical formulas to them [7]. Additionally, it is more likely for a user to have time to provide a simple rating via scale, than to write a review about an item. De Gemmis et al. also talk about privacy issues that arise with a non-anonymous textual type of feedback, as many users may not be willing to share their intentions to a broader community of users [7].

Another important source is the history of a user's interaction with the provided service: gathering information about items that a user has viewed, or storing search queries that a user has typed can be used by a RS algorithm to draw conclusions about her preferences [8]. This type of information is also known as "implicit rating" [7].

The main advantage of using content-based filtering is the personalized suggestion. Further, a user is not dependent on other users' preferences and ratings. The CB approaches consider all items equally. It does not matter whether an item has not yet been rated (e.g. new items) or is popular among other users. This means, that so called *new item problem* is not an issue for CB RSs ([7], [9]).

However, CB RSs also have disadvantages. The most common one is the *new user problem*: It is nearly impossible to make meaningful recommendations for users when their preferences and previous activities are unknown [9]. Furthermore, some users do not give sufficient feedback, or their preferences change quickly, thus, the learning-data is fairly scarce [9].

According to de Gemmis et al. two more aspects can be viewed as drawbacks of CB RSs [7]:

[6] `http:www.last.fm` (Last visited: 25.03.2013)

- *Limited content analysis:* If a system does not provide enough specific information about items, it is very hard to categorize items in recommendable and non-recommendable lists.
- *Over-specialization:* content-based RSs can only make suggestions similar to what a user has already found interesting. This approach is not able to recommend new items that correspond to new ideas that a user may find interesting.

2.3. Collaborative Filtering Recommendation Systems

Collaborative filtering (CF) uses a different approach than described in Section 2.2: characteristics of an item that is being recommended are not considered. The suggestions are made depending on the feedback of other users that have interacted with the item. A usual indicator of users' opinions is a rating, whereas a rating can be given by a user explicitly by providing feedback about the item, or implicitly, in which case user interaction is being studied [10].

Schafer et al. describe the three most important functionalities a CF system should have [10]:

1. *Recommend items:* a CF system should make suggestions that are meaningful for a user.
2. *Predict for a given item:* predict how it could be rated by a user.
3. *Constrained recommendations:* Sometimes the suggestions need to be made based on some characteristics that are desired by the user. A CF system should first identify a set of items with such characteristics and then find suitable suggestions from this set. [10].

It is easy to notice that this type of RSs depends highly on a number of users, the number of ratings they have given to items and lastly, on the number of items itself. Hence, there are some challenges that a CF system can face. These are described in literature as follows [7]:

1. *New User Problem*: When a user has just started to use the given service, it is impossible to recommend items because her preferences are not yet known.
2. *Sparsity Problem*: Users do not tend to rate everything they have liked or disliked. Thus, a recommendation system has to make suggestions based on a small set of data, which in many cases is insufficient.
3. *Unusual User*: The CF recommendation systems depend on users who have a similar taste. Of course, it is not the case that two or more individuals like exactly the same things, but we can draw some conclusions depending on the items they both have liked in the past. A provider using recommendation services has to assume that there are more individuals with similar taste than people with no or very little similarity to other users. The last is called "unusual user", or a "grey sheep" [7]. To put it simply, there are always people who have unusual taste and thus differ from other users. It is therefore difficult to predict which items

they could like. This problem can be partially solved by a rising number of users, because the higher the number of users, the higher the chance that two or more of them have a similar taste [7].

4. *Scalability Problem*: The amount of data a CF system requires to make a meaningful recommendation can become a hazard. For example, if there are many users, finding similar users for each user and then investigating which items these users have liked can require serious amount of computational resources [7].

5. *Lack of Transparency*: Most of the time users have to accept recommendations without knowing anything about how these were formed [7].

CF systems can further be categorized into *neighbourhood-based* and *model-based* systems.

Neighbourhood-based CF systems identify users that are similar to a current user. This set of users is called the *neighbourhood* of a current user. The system then makes suggestions based on item ratings of the users from the given neighbourhood ([9],[11]).

Model-based CF systems: In this case statistical models are used to build a learning-data and to estimate statistical parameters. These parameters are then used on the existing user ratings to make recommendations [9].

A good known example for a web site which uses collaborative filtering in order to make recommendations is Amazon.com, Inc[7].

2.4. Hybrid Recommendation Systems

So far we have discussed content-based RSs in Section 2.2 and collaborative filtering RSs in Section 2.3 and have distinguished between their advantages and shortcomings. There are services where content-based RSs work well, but CF RSs do not yield good results with regards to user recommendations and vice versa.

Hybrid recommendation systems combine two or more RS approaches to minimize their individual problems and take advantage of their strengths. The most common reason why hybrid RSs are used is the *cold-start problem*. This problem arises when a new user sings up to use a service, or when a new item is added to a list of items [12].

Of course two or more of the same types of RSs can be combined to build a hybrid RS, but we will concentrate on the seven types of hybrid RSs here, that are defined in [12] as:

- *Weighted hybrid RS:* In this type of a hybrid RS, different approaches are used to determine the recommendation score of an item. The final score is then just the weighted linear

[7]`http://www.amazon.com` (Last visited: 26.03.2013)

combination of all scores.

- *Mixed hybrid RS:* Recommendation scores are not combined, rather the user is shown ranked lists of recommendations that are derived from both approaches separately.
- *Switching hybrid RS:* Depending on the characteristics of a user, one RS approach is chosen and recommendations are made without use of the other approaches.
- *Feature Combination hybrid RS:* Typically every type of RS is designed according a special algorithm. Feature Combination hybrid RS does exactly the opposite. Here different features that should be processed by one algorithm are included in another one. Thus, technically, the result is a sole recommendation system, other than the hybrid systems discussed above.
- *Feature Augmentation hybrid RS:* Here the result feature of one approach serves as an input for the another approach which for its part, makes further calculations based on its own algorithm.
- *Cascade hybrid RS:* There is one main approach according to which the recommendation scores are calculated. The other approach serves as an additional refiner, but it can not overturn the results of the dominating one.
- *Meta-Level hybrid RS:* Same as in feature augmentation hybrid RS, this approach also uses ready features provided from one approach as an input for the second approach, but "the difference is that in a meta-level hybrid, the contributing RS completely replaces the original knowledge source with a learned model that the actual RS uses in its computation"[12].

There are ample number of scientific papers about hybrid recommendation systems. Melville et al. present content-boosted collaborative filtering they have used for movie recommendations. In their case, a content-based approach makes it possible to predict ratings for movies for each user (even if there are no real ratings from a user regarding the item in the database). These ratings are subsequently used as an input in a collaborative filtering approach [13].

In another paper, Balabanovic et al. present "Fab": a project of Stanford University digital library, a hybrid approach to recommend specific articles from world wide web that could be interesting for a user [14]. Here the content-based RS is used to create user profiles which are based on the user's preferences. The user profiles are then the basis for finding similar users to use their ratings and preferences to make recommendations in a classic collaborative filtering approach.

2.5. Other Types of Recommendation Systems

Demographic Recommendation Systems make suggestions based upon a user's demographic data. Basically, it means that a service provider makes suggestions based on users' preferences, who belong to the same demographic groups.

A simpler approach of using demographic RSs is for example, recommending age-appropriate items that consider the user's age, or displaying the content in the user's native language [1]. In this case, ratings of other users or of the user herself are not considered, thus, it is always possible to make a meaningful recommendation, provided the user is not reluctant to share personal and thus sensitive information about herself. This may be the main reason why not very many service providers use this type of recommendation system [7].

Knowledge-based Recommendation Systems make recommendations by considering whether or not an item is useful for a user, based on user's interest or preferences [7]. Therefore, information about user's needs is crucial. Two types of Knowledge-based RSs are to be mentioned: case-based and constraint-based systems.

- *Case-based Reasoning* (CBR): A suggestion is made based on the suggestions that were made for similar situations in the past [15]. In [16] the author examines the two most popular RSs that are used by travel and tourism sites. Although, none of them are pure knowledge-based RSs (they combine content-based and collaborative filtering approaches), they do use a case-based approach, to ask users about their needs and constrains and try to make recommendations based on gained information.
- *Constraint-based Systems:* In this case, a recommendation system chooses a solution that is explicitly defined for the type of problem [1].

Community-based Recommendation Systems: A RS suggests items that users' friends find interesting. Sinha et al. conducted a study on six RSs (three book RSs and three movie RSs) and compared the recommendations made by RSs to friends' recommended items. The study suggests that a user tends to value a recommendation of a friend higher than one of a user who might be similar to her, but who she doesn't know personally. However, recommendations made by RSs also were of a significant value for users [17].

Community-based recommendation systems are becoming very popular with the rise of social networks. For instance, Last.fm[8] provides a list of "friends' loved tracks", or the book recommendation service Goodreads[9] lists books that user's friends have found interesting.

[8] http://www.last.fm (Last visited: 22.03.2013)
[9] http://www.goodreads.com (Last visited: 22.03.2013)

2.6. Recommendation Systems for Mobile Devices

As ubiquitous computing takes an important place in people's everyday lives, the interest towards RSs that can be easily used by mobile applications also rises. Users having the opportunity of using recommendation services via mobile devices can have many benefits. Mobile computing provides advantages other than its rising popularity, that cannot be overlooked. Ricci et al. talk about the advantage of knowing where exactly the user is at a given moment and the advantage of not being dependent on a PC, thus, being able to provide information exactly where and when a user needs it [2]. But, there are also obstacles that have to be considered that are not a subject for RSs providing a service for PC users. The nature of mobile computing, storage capacities, and displaying the information are some of these obstacles. The benefits of RSs in mobile computing are highly acknowledged in tourism. Some of the services offered by such RSs are: recommending attractions in a given city, tourist-services, tours through the city etc. [18].

Two important types of RSs have emerged in the area of mobile computing:

- *Location-Aware Recommendation Systems:* The location of a user or of an item is valuable [19]. For example, a recommendation should take into account the distance between a user and an item (e.g. the closest restaurant). Furthermore, Stiller et al. argue that the distance between an item and a user also contains some sort of implicit rating. For instance, the longer the user was willing to travel to reach a (rated) item, the higher its ranking should be [19].
- *Context-Aware Recommendation Systems:* this type of RS suggestions are based on a user's current context state. A context in this particular situation can be the user's location, the individuals or objects near the user, date, time, temperature etc. ([20], [21]).

Some examples of mobile recommendation systems are:

- Tumas et al. present PECITAC, which helps a user to get to a certain place in a city using public transport [22]. Rather than providing a user with a simple path to a chosen destination, this application takes user preferences into account by asking additional questions and provides a personalized solution based on the answers.
- Vico et al. present a two-phase proactive model of RS for Android application. This widget-based application, "Proactive Restaurant Recommender", provides recommendations for restaurants based on the current context, such as current activity of a user, time, location and whether the user is alone. Based on the context, it is estimated if a recommendation is meaningful and only after that a second phase of the RS, namely calculating which restaurants to recommend and displaying them, takes place [23].
- Ko et al. describe a movie recommender mobile application that suggests movies of a user's

preferred genre currently playing in cinemas close to the user [24].

2.7. Slope One- An Example of a Collaborative Filtering Recommendation System

In 2005 a new collaborative filtering approach was introduced by Lemire et al. [25]. The main aspect of this rating-based CF is to predict the user's preferences based on preferences of other users. The estimated rating $f(x)$ is calculated according to *f(x)* $= x + b$, where x represents the rating that is being predicted and b is a constant, representing the average difference in preference values between two given items by other users.

The Slope-One algorithm basically closes the gap between an item i (which was *not* yet rated by the user) and an item j (which was rated by the user) by taking into account the pairwise average rating differences between i and j (given by other users). For example, if the rating of j by the user is 2 and all the other users that have rated i *and* j think that i is $+1.0$ better than j, than the algorithm would expect that the user will rate the item i with a $3.0 = 2.0 + 1.0$.

To generalise the above assumption, *b* is set to an average difference between all ratings which two users have given to same items: $b = \frac{\sum_{i=1}^{n} u_i - v_i}{n}$, where u_i and v_i are ratings, users u and v have given to items, respectively. In this set, only the items that both users have rated are considered.

So, if given a set X of ratings of all items and any two items j and i with ratings u_i and u_j by a user u, the average deviation is as follows:

$$\mathrm{dev_{j,i}} = \sum_{u \in S_{j,i}(X)} \frac{\mathrm{u_j - u_i}}{\mathrm{card(S_{j,i}(X))}} \tag{2.1}$$

where $card(S_{j,i})$ is the number of elements in $S_{i,j}(X)$ and where $S_{i,j}(X)$ represents all users that have rated both items [25]. The created matrix is anti-symmetric (i.e. $dev_{i,j} = -dev_{j,i}$).

Finally, the prediction for an item j given a rating for item i is calculated as follows, suggesting that a predictor should be calculated as the average out of all given predictors: $dev_{i,j} + u_i$:

$$P(u)_j = \frac{1}{card(R_j)} \sum_{i \in R_j} (dev_{j,i} + u_i) \tag{2.2}$$

where R_j is the set of all relevant items [25].

The Authors argue that the Slope-One algorithm meets five goals that they have set at the beginning, such as simplicity of implementation, easily up-datable deviation matrix, resulting fast queries, handling the *"new user"* problem and being competitive to other CF algorithms. The authors also note the drawbacks of the algorithm, such as expensive storage costs, which they justify with fast queries [25]. Further, the number of overall ratings for a given item is also of a high value. The more ratings an average can be built from, the more accurate the prediction will be. In the plain Slope-One approach this problem is not considered. To address this problem, the authors provide a Weighted-Slope-One scheme as a solution, where the number of pairwise rated items that appear more frequently is being considered.

The Slope-One recommendation algorithm is used by "Value Investing News[10]" a web site providing news in stock market.

[10] http://www.valueinvestingnews.com (Last visited: 20.03.2013)

Chapter 3.

MobiDics

In this chapter the MobiDics web- and mobile applications and their features are briefly described.

Motivation

After analyzing the varied recommendation systems currently being employed and reviewing examples of each, the following will discuss the benefits of using a RS for mobile and web applications that support the improvement of teaching/learning quality.

Numerous didactic methods can be applied while teaching and each one of them can be used in different situations. Based on a person's preferences, conclusions can be drawn about which didactic methods are important to her. A RS can then provide tailored suggestions to the user about similar didactic methods, offering several additional benefits: (i) providing a person with recommendations may help her learn about more didactic methods that are well suitable for her purposes. Thus, the person will have an opportunity to vary the didactic methods during the class and keep students motivated and interested; (ii) the person can apply several didactic methods and identify which of them she is more confident to use during the class.

It may also be helpful to recommend didactic methods to the less experienced theaching staff that the more experienced users have already found useful.

3.1. MobiDics - An Online Didactic Tool

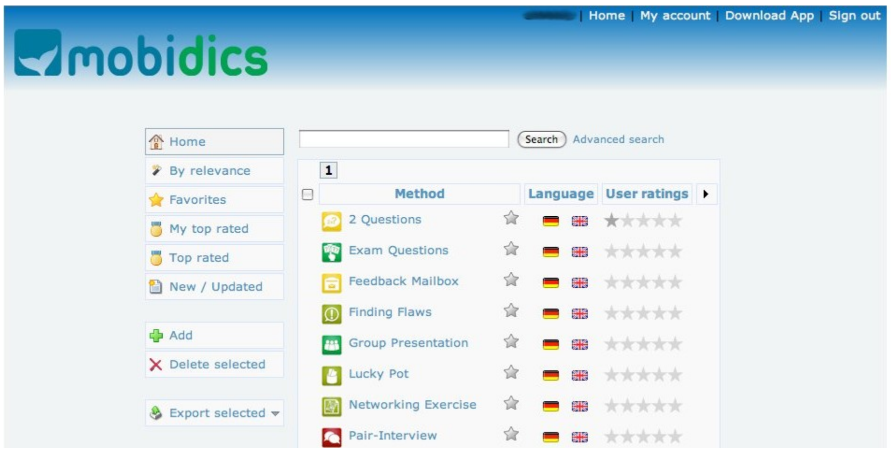
MobiDics (short for "Mobile Didactics") is an online service which supports educators at universities to improve their teaching techniques. This especially applies to personnel that have little experience in teaching, such as PhD students or teaching assistants [3]. MobiDics has been developed during the previous two stages and is currently available as both mobile and web applications [26]. Möller et al. argue that the benefits of MobiDics mobile application are: the ability to use it everywhere (*Everywhere Use*), additional multimedia contents that are provided to describe didactic methods and their usage (*Better Understanding*), the ability to filter the methods depending on a given context and/or location (*Context Sensitivity*), the opportunity for users to leave feedback, e.g., rate didactic methods, mark them as favorite, or leave comments (*Pervasive Cooperation*) [3]. Furthermore, MobiDics can be viewed as a "social network of teaching personnel", where teaching personnel can share their knowledge with each other, receive feedback and advice from experienced colleagues [3].

To make MobiDics' functionality easier to understand for a reader, a more detailed description of MobiDics web and mobile applications is provided in the following sections.

Didactic methods: The core idea of MobiDics is to provide teaching personnel at the university with the set of didactic methods which can be used to improve the teaching quality [4]. There are numerous didactic methods that a user can read, rate, mark as favorite, or leave a comment. The didactic methods have their properties, such as a suitable group size, or materials, used during a particular didactic method.

3.2. Web Page

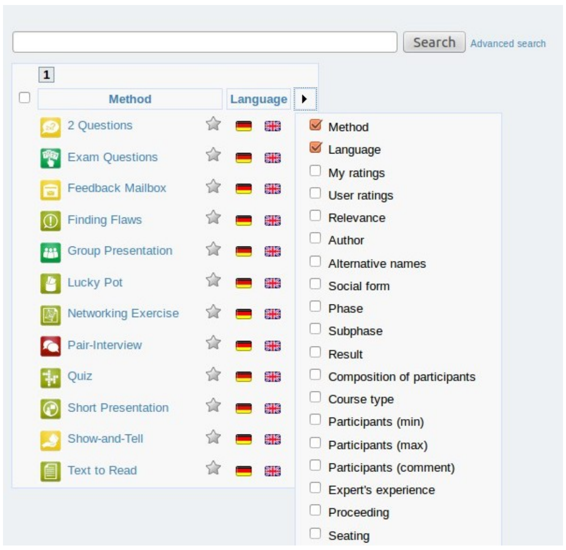
As shown in Figure 3.1, upon logging in a user can view a list of didactic methods. After selecting one of these methods, a detailed description is displayed. The left sidebar provides further categories, such as methods sorted by their relevance, favorite methods, methods that a user has rated, methods that have the highest ratings, or new methods. A user can also add a new method or delete one from her view. Further, chosen didactic methods can be exported into a csv file.



Method		Language	User ratings
2 Questions	☆	🇩🇪 🇬🇧	★☆☆☆☆
Exam Questions	☆	🇩🇪 🇬🇧	☆☆☆☆☆
Feedback Mailbox	☆	🇩🇪 🇬🇧	☆☆☆☆☆
Finding Flaws	☆	🇩🇪 🇬🇧	☆☆☆☆☆
Group Presentation	☆	🇩🇪 🇬🇧	☆☆☆☆☆
Lucky Pot	☆	🇩🇪 🇬🇧	☆☆☆☆☆
Networking Exercise	☆	🇩🇪 🇬🇧	☆☆☆☆☆
Pair-Interview	☆	🇩🇪 🇬🇧	☆☆☆☆☆

Figure 3.1.: A screen shot of the home page of the MobiDics web application. A property "User ratings" is added to the view. The categories in the left sidebar show the didactic methods sorted according specific characteristics.

As a default, each of the categories list methods by their names and additionally show two more properties: whether or not a user has marked these methods as favorites and in which languages their descriptions are available. A user, however, can configure the settings and add more properties of a didactic method to the view. This is possible through the right sidebar, which is designed as a drop-down menu. The user has the opportunity to check all properties that are to be added to a view (see Figure 3.2).



Method	Language
2 Questions	🇩🇪 🇬🇧
Exam Questions	🇩🇪 🇬🇧
Feedback Mailbox	🇩🇪 🇬🇧
Finding Flaws	🇩🇪 🇬🇧
Group Presentation	🇩🇪 🇬🇧
Lucky Pot	🇩🇪 🇬🇧
Networking Exercise	🇩🇪 🇬🇧
Pair-Interview	🇩🇪 🇬🇧
Quiz	🇩🇪 🇬🇧
Short Presentation	🇩🇪 🇬🇧
Show-and-Tell	🇩🇪 🇬🇧
Text to Read	🇩🇪 🇬🇧

- [x] Method
- [x] Language
- [ ] My ratings
- [ ] User ratings
- [ ] Relevance
- [ ] Author
- [ ] Alternative names
- [ ] Social form
- [ ] Phase
- [ ] Subphase
- [ ] Result
- [ ] Composition of participants
- [ ] Course type
- [ ] Participants (min)
- [ ] Participants (max)
- [ ] Participants (comment)
- [ ] Expert's experience
- [ ] Proceeding
- [ ] Seating

Figure 3.2.: A screen shot of the home page of MobiDics web application showing the drop-down menu to display additional properties.

Two sorts of search mode are provided: first, the Simple Search Mode makes it possible to search for a given term and display methods, which contain the searched term in their description, whereas the second, the Advanced Search Mode, enables to search for terms in specific properties. These properties can be added one by one to the Advanced Search.

Under "My account", users can add personal information about themselves such as date of birth, name, occupation, university, where they teach or how many years of teaching experience they have.

3.3. MobiDics Mobile Application

The MobiDics application is provided for Android mobile phones. After a user has downloaded the application to install it, a setup screen appears, where a user can configure settings according to her wishes. When the configuration is complete, a user can login and view the didactic methods. The main screen provides all available methods, just as on the web site (see Figure 3.3). A user can navigate between other views by sliding the finger along the screen. All views that are available via the web site are also represented here. The personal settings can be changed via preference menu, where additional features (e.g. preferred internet connection) can be chosen. Actions such as rating a method, leaving a comment, and marking a method as favorite are also possible via mobile application. The data is also locally stored on the mobile device so that the application usage is possible even if there is no internet connection.

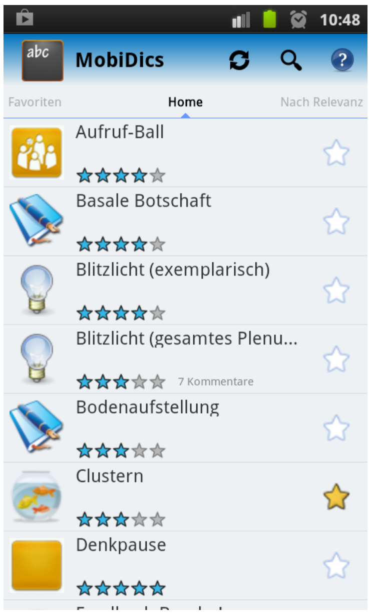


Figure 3.3.: A screen shot of the main screen of the MobiDics mobile application. The blue stars show the average user rating of a didactic method. The yellow star indicates that a user has marked the didactic method as favorite.

3.4. Server

The didactic methods are stored in a SQL database on a central server. Since it is important to have the same representation of MobiDics data on both the web and mobile applications, the changes that a user makes on a mobile device are automatically uploaded to the server so that they are mirrored on the web application. For example, if a user rates a didactic method or marks one as favorite, these actions should be also displayed on the web site. Should a user make changes while operating the application without an internet connection, the changes will be saved and uploaded upon the next login [4]. Synchronisation works in the opposite direction: new didactic methods and comments made by other users are downloaded on a mobile device and the data is transported via XML files.

Chapter 4.

Recommendation Algorithm for MobiDics Application

In the previous chapters the state-of-the-art recommendation systems and the existing features of MobiDics were described. Furthermore, the possible benefits of enhancing the MobiDics functionality with a RS were discussed.

In this chapter, a recommendation algorithm for MobiDics is presented. Shortcomings of the different approaches are discussed and possible solutions are provided. It was decided that the best choice in this instance was to implement a “weighted hybrid system” (cf. Section 2.4), that means to combine content-based and collaborative filtering approaches. This way possible shortcomings (such as the “new user problem” / “new item problem”) can be better avoided ([9], [7]).

4.1. Content-Based Approach

As mentioned in Section 2.2, a content-based recommendation system considers a user’s previous activity and learns about her preferences to find similar items, in our case, didactic methods. It does not consider the behaviour of other users. Thereby, we focus on an item itself and its characteristics, to be able to build a model of an item. This model will be compared to other items a user has not yet shown a preference to, to determine whether they should be recommended to a user.

4.1.1. Characteristics of an Item

As mentioned above, every didactic method is characterised by a number of properties. For example, this could be the group size or time needed for using the didactic method. These properties are stored in a database. Not all of these characteristics are taken into consideration, but a subset is defined that is more relevant and important in this case. In the end eight properties were selected to calculate the recommendation scores for each method. These are represented in table 4.1.

Table 4.1.: Method properties

field	**data type**	**description**
group size	interval	size of the group a method is suitable for
time	interval	time needed to perform the method
course type	nominal	suitable for seminar/ exercise/ lecture
phase	nominal	a learning phase during which a didactic method is suitable
sub-phase	nominal	a learning sub-phase during which a didactic method is suitable
social form	nominal	the form of socializing while using the didactic method
material	nominal	materials needed
group type	ordinal	homogeneous to heterogeneous

Note, that for ease of usage, all values of properties that are of a nominal data type are represented as integers in a database. A property of a didactic method can have more than one value. In particular, this applies to data of a nominal type. For example, a didactic method can be used during a seminar, a lecture or an exercise. In this case the values are stored in a database divided by colons.

4.1.2. The Concept of a Reference Average

We now present our approach of building a learning model based on the user's preferences. To be able to provide recommendations, a user has to already have rated at least one method. Further, we also consider didactic methods that a user has stored as favorite ones. As a result, we have more didactic methods and thus more information to work with. It can be assumed that this step enhances the accuracy of a recommendation.

Based on rated and favorite methods we create an "abstract didactic method", i.e., a didactic method that has the average properties of all methods the user has rated (at least a three-star rating) or marked as a favorite. Obviously, there is most likely not an actual didactic method in the database that matches exactly with the properties of the "abstract" method. This abstract method will be used as a reference to compare it with other methods, that have not been rated by the user yet.

The "abstract method" is constructed as follows: (i) for the interval data type we calculate an average value for the upper and lower bounds of an interval, (ii) for the ordinal data type we calculate the average value and (iii) for the nominal types we determine the most frequently used value. The calculation for (i) is as follows:

$$\begin{aligned} \text{AverageLowerBound} &= \frac{1}{n}\sum_{i=1}^{n} \text{lowerBound}_i \\ \text{AverageUpperBound} &= \frac{1}{n}\sum_{i=1}^{n} \text{upperBound}_i \end{aligned} \tag{4.1}$$

where n represents the number of methods that a user "showed interest" in. Methods that are interesting for a user are defined as the union of all methods that were rated (and at least have a three-star rating) by the user with the methods that are the user's favorites.

If the property is of a nominal type, we count the frequency of appearance of every value and choose the one that appears most frequently. The algorithm to build the "abstract didactic method" is shown in algorithm 1.

Algorithm 1 Create "abstract didactic method" based on user's preferences.

```
 1: for every rated (rating $\geq$ 3)/favorite item for a given user do
 2:     for every property do
 3:         if property is of a nominal type then
 4:             increment counter for each nominal value;
 5:         end if
 6:         if property is of an interval type then
 7:             sum up lower and upper bound separately;
 8:         end if
 9:         if property is of an ordinal type then
10:             sum up value;
11:         end if
12:     end for
13: end for
14: Calculate the average values for interval types (according to (4.1))
15: Calculate the average for ordinal values
16: Select the value that occurs most often (for nominal type)
```

This way, we have created a method that represents the average of methods a user has shown interest in. The next step is to compare every method a user has not yet rated or marked as favorite with the "abstract didactic method" and calculate the recommendation scores. Every property will have its own sub-score to determine how it affects the final score.

For every method that a user has not yet shown a preference for, we calculate the sub-score for every property as follows:

- Nominal types: if the value matches with the "abstract method" set the sub-score to 0.8 and to 0.0 otherwise (the reason why 0.8 is chosen will be explained in more detail in sub-section 5.1.1.

- Ordinal types: we calculate the sub-score based on the difference between a current value and the “abstract method”.
- Intervals: we calculate the sub-score based on the Manhattan distance as explained below.

For the interval data type we need to determine how well two given intervals match. Therefore, we have to use some type of a distance function. For our calculations we use the normalized Manhattan Distance. If we have two items $a = (x_1, y_1)$ and $b = (x_2, y_2)$, then the Manhattan Distance between the items is defined as [27].

$$\mathrm{ManhattanDistance(a,b)} = |x_1 - x_2| + |y_1 - y_2| \tag{4.2}$$

In order to have a sub-score in the range [0,1], we need to normalize the distance function [27]. This is achieved by dividing (4.2) by the range, which is :

$$\mathrm{range} = \max_i y_i - \min_i x_i \qquad (i = 1, \dots, n) \tag{4.3}$$

For some properties the relative distance can be more important than the absolute distance. As an example the property "number of participants" can be considered. A user that was interested in methods for (on average) 10 people so far will probably be less interested in a method for 60 participants. However, if the user was interested so far in methods for about 500 people, then making a suggestion for a method for 550 participants seems to be a closer match. In both examples, the absolute difference is 50 persons, however, the relative difference is 500% in the first example, whilst it is only 10% in the second.

Therefore, we think it makes sense to consider the logarithm of the number of participants, as this takes the order of the magnitude into account. On a logarithmic scale, a comparable match for the 500 vs. 550 example would be 10 vs. 11 participants (which also implies a relative difference of 10%).

Then the sub-score for the interval types (such as number of participants) becomes:

$$\begin{aligned}\mathrm{subScore}_i = 1.0 \quad - \quad &\left(\frac{|\log(\mathrm{lowerBound}_i) - \log(\mathrm{averageLowerBound})|}{\log(\mathrm{range})} + \right. \\ &\left. \frac{|\log(\mathrm{upperBound}_i) - \log(\mathrm{averageUpperBound})|}{\log(\mathrm{range})}\right)\end{aligned} \tag{4.4}$$

where the range is calculated according to (4.3). In our algorithm a score close to 1.0 means a good match, however, the term in brackets is close to 0.0 if the intervals are close to each other. This is the reason for the 1.0 and the negative sign in the equation (4.4).

Further, if we have a property of an ordinal type, we use the following formula:

$$\mathrm{subScore_i} = 1.0 - \frac{|\mathrm{currentValue} - \mathrm{averageValue}|}{\mathrm{range}} \tag{4.5}$$

where range is difference between maximum and minimum possible values.

We also consider the possibility that the importance of different properties can vary. For example, it may be more important to know how long one method lasts, than to know during which sub-phase it can be suitable. Thus, we add weights for each property to our approach and define the final (content-based) matching score as the weighted sum of all sub-scores:

$$\mathrm{finalScore} = \sum_{i=1}^{n} (\mathrm{subScore_i} \cdot \mathrm{criteriaWeight_i}) \tag{4.6}$$

where $\mathrm{subScore}_i$ is calculated either in (4.4) for data of an interval type, or in (4.5) for ordinal types of data or, in case of a nominal type of data, set to 0.8 (matching) or 0.0 (no match).

The pseudo code of an algorithm is represented in Algorithm 2.

Algorithm 2 Calcualte matching score

```
 1: for every item that a user has not yet rated or marked as favorite do
 2:   for every property do
 3:     if a property is of a nominal type then
 4:       if  a value of a current property matches the calculated value then
 5:         sub-score for current property = 0.8;
 6:       else
 7:         sub-score for current property = 0.0;
 8:       end if
 9:     end if
10:     if a property is of an interval type then
11:       calculate sub-score according to (4.4);
12:     end if
13:     if a property is of an ordinal type then
14:       calculate sub-score according to (4.5)
15:     end if
16:   end for
17:   calculate the final matching score for a current item according to (4.6);
18: end for
```

In terms of realization of the algorithm, there are two possibilities: the scores can be calculated either on the server-side, or locally on a mobile device. Both ways have their benefits and disadvantages: as we have a web and mobile application of MobiDics, it is reasonable to have the scores stored in a central database. In this case we have to ensure that a mobile device has access to the data. Considering the fact that a mobile device might have no internet connection, we have to ensure that the recommendation service is provided anyway. For this reason, we store the calculated scores additionally in a local database on the mobile device. Changes made while being offline will be considered once internet connection is re-established. We take into consideration, that in this case a user will not be able to access the most up-to-date data while being offline.

An alternative would be calculating the scores on the server and the client side. However, this would require synchronisation between client and server, require additional computational resources on the mobile device and would lead to code duplication.

Updating the database table requires resources. Thus, we need to clearly set conditions, upon which recalculation of scores should take place. If a user rates a new method or stores one of them as favorite, recommendation scores have to be updated. This case also handles situations when a user changes a previously given rating of a didactic method. Additionally, we update scores upon login.

4.1.3. Challenges

The following cases have to be taken into account:

- In some cases, the lower and/or upper bounds of an interval are left empty to indicate that the method is suitable for every value the property can represent. This may jeopardize the calculation. To ensure correctness we set sensible default values for the upper and lower bounds (e.g. for the property "time" an upper bound of 90 minutes is used).
- Didactic methods that a user has rated or saved as a favorite should not appear in the recommendation list.

Like other content-based recommendation algorithms our approach has also its shortcomings. If we have a new user who has not yet rated any didactic method, or saved one of them as a favorite, the RS is unable to make recommendations ("new user problem"). Furthermore, we only consider the given user's preferences, thus the problem of over-specialization that we have discussed in Section 2.2 occurs. By adding a collaborative filtering algorithm to our approach and building a hybrid algorithm we hope to minimize these shortcomings and make recommendations more precise.

4.2. Collaborative Filtering Approach

We use Slope-One as collaborative filtering approach [25]. In this case only the average rating of a didactic method is considered. The average rating is calculated from all ratings given from users. Based on user preferences and ratings from other users a rating for a didactic method, which the current user has not yet rated, can be predicted.

The estimated rating according to the Slope-One algorithm is calculated in two steps: in the first step, a deviation matrix is built. The matrix contains average differences between ratings for every item pair (m_i, m_j). The average ratings comprise the ratings of all users that rated both m_i and m_j. Algorithm 3 shows all steps that are needed for the creation of the deviation matrix.

Algorithm 3 Create deviation matrix Δ

```
1: for every pair $(m_i, m_j)$ of didactic methods do
2:     for every user $u$ do
3:         if a user has rated both didactic methods $m_i$ and $m_j$ then
4:             add the user ratings $r(m_i)$ and $r(m_j)$ for the methods $m_i$ and $m_j$ to the overall ratings
               of $R(m_i)$ and $R(m_j)$, respectively.
5:         end if
6:     end for
7:     calculate the average ratings $r(m_i)= R(m_i)/u_{i,j})$ and $r(m_j)= R(m_j)/u_{i,j})$, where $u_{i,j}$ is
       the number of users that have rated both $m_i$ and $m_j$.
8:     calculate difference between these ratings: $\Delta_{i,j} = r(m_i) - r(m_j)$
9: end for
```

After the deviation matrix is created, the second step of the algorithm can be executed. This means that predictions are made for all didactic methods that a user has not yet rated. For a rated prediction $r(m_x)$ we consider every pair of methods: (m_i,m_x), where m_i represents all rated methods by the user and m_x is a method that a user has not yet rated (see algorithm 4). We add the average difference of the pair (m_i,m_x) to the actual rating of method m_i, for every such pair and build an average afterwards. The final result is the predicted rating.

Algorithm 4 Predict ratings for a user u

```
1: for every didactic method $m_x$ to predict the rating do
2:     for every didactic method $m_i$ that a user has rated do
3:         add a difference, $\Delta_{i,j}$ to the rating of $m_i$;
4:     end for
5:     build an average of the sum;
6: end for
```

In order to be able to make predictions for a rating $r(m_x)$ the following condition has to be fulfilled: there has to be at least one didactic method m_i which is rated by the user *and* the entry for $\Delta_{i,x}$ has to exist.

We now look closely at a deviation matrix that we create in the pre-calculation step of an algorithm. There are n didactic methods, which means that the deviation matrix has n^2 entries. As explained above, the deviation matrix is anti-symmetric ($\Delta_{i,j} = -\Delta_{j,i}$). Therefore, it would be possible to roughly halve the storage requirements to $\frac{1}{2}n(n-1)$. Nevertheless, we chose to store the whole matrix in the database for computational efficiency, as we expect that the number of didactic methods will remain rather small going forward.

The deviation matrix changes in the following situations: (i) when a new didactic method is added, (ii) when a didactic method is deleted, (iii) when a method is newly rated by a user, (iv) when a user changes the rating of an already rated method.

In all cases (except for (iii)) the computational effort is $O(n)$. However, if frequent changes of ratings for many users occur, this might cause some burden on the server. Therefore, we suggest that the matrix is updated only once per day (preferably during off-peak hours). For testing purposes the deviation matrix was calculated for each login which was fast enough in the development environment of MobiDics.

4.3. Combination of Both Approaches

We combine both approaches (content-based and collaborative filtering) as discussed above. The content-based approach determines a weighted score based on all sub-scores which eventually has to be normalized again (as the weights are in the range from 1 to 10). The collaborative filtering approach predicts a rating (from 1.0 to 5.0) as discussed in section 4.2, which also needs to be converted into a [0,1] range. The latter is achieved by $\mathrm{score} = (\mathrm{rating} - 1)/4.0$.

Finally, the two scores are combined by taking the weighted sum of both scores. The weights are for now set to the same values (i.e. the content-based and collaborative filtering approach have the same impact), but can be later adapted based on user feedback.

We benefit from a content-based approach because we do not depend on other users to make a recommendation. In addition, we benefit from a collaborative filtering approach because we do not depend on the didactic methods' specific properties. These results complement each other and it can be assumed that we have more accurate recommendations as a result.

Chapter 5.

Implementation

In this chapter the implementation of the content-based and collaborative filtering algorithms are discussed. Additionally, the steps of the combination of these approaches to a weighted hybrid RS will be shown.

5.1. Content-Based Approach

5.1.1. Android

Recommendation Screen and RecommendTask

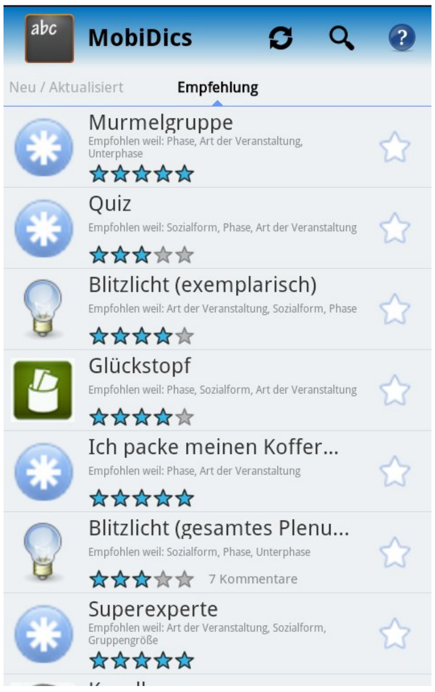
As it was noted in Subsection 4.1.2, for every selected method property a separate sub-score is calculated and the final recommendation score is the weighted sum of these sub-scores. Further, these sub-scores are calculated on the server-side and stored in a central database table. When a user lacks an internet connection, it must be verified that recommendation scores are also locally stored on an Android device. For this reason, a background (asynchronous) task was implemented to download the scores when the task is triggered. The benefit of a background task is that a user can use the application without noticing that something is going on in the background. Furthermore, downloading sub-scores can take time. Since the calculation on the server-side is generally faster when compared with the mobile device and because the download is already triggered at the home screen, the user will in general see the most up-to-date results by the time she switches to the recommendation page.

The object RecommendTask sends HTTP POST requests to the server. On the server-side the request is processed and an XML file containing method IDs and sub-scores for each method is created. The XML file is then sent back to the mobile device and parsed. Thereafter, the final score is locally calculated according to equation (4.6). The weights for every sub-score are stored on the mobile device in the SharedPreferences object, thus, these values do not need to be downloaded. Calculated scores are stored in a local database table *ddk_recommend*.

At this point, recommended didactic methods would be ready for presentation by simply displaying the list of didactic methods sorted by a recommendation score. However, a user may also be interested in why a method is being recommended to her. Thus, the RS should also provide information about which sub-scores/properties had the highest impact on the final score.

To accommodate this, a closer look at the list of sub-scores indicates that not every property needs to be listed, since some of them only minimally contributed to the final score. Therefore, a threshold of 0.6 for the sub-scores was defined and only up to the three highest of them are selected. The results are stored in the local database table *ddk_recommend_because*. (Note that at the early stages of implementation a sub-score for nominal types of data was set at 1.0, if it matched the comparable value. This would suppress the ordinal and interval data types, as these hardly reach a sub-score of 1.0. Therefore, the sub-scores of nominal data type for matches were changed to 0.8).

The page on which the recommended methods are displayed has been enhanced in its functionality. An additional *TextView* has been added, which lists the properties that represent the highest sub-scores (see Figure 5.1).


MobiDics

Neu / Aktualisiert
Empfehlung

Murmelgruppe
Empfohlen weil: Phase, Art der Veranstaltung, Unterphase

Quiz
Empfohlen weil: Sozialform, Phase, Art der Veranstaltung

Blitzlicht (exemplarisch)
Empfohlen weil: Art der Veranstaltung, Sozialform, Phase

Glückstopf
Empfohlen weil: Phase, Sozialform, Art der Veranstaltung

Ich packe meinen Koffer...
Empfohlen weil: Phase, Art der Veranstaltung

Blitzlicht (gesamtes Plenu...
Empfohlen weil: Sozialform, Phase, Unterphase
7 Kommentare

Superexperte
Empfohlen weil: Art der Veranstaltung, Sozialform, Gruppengröße

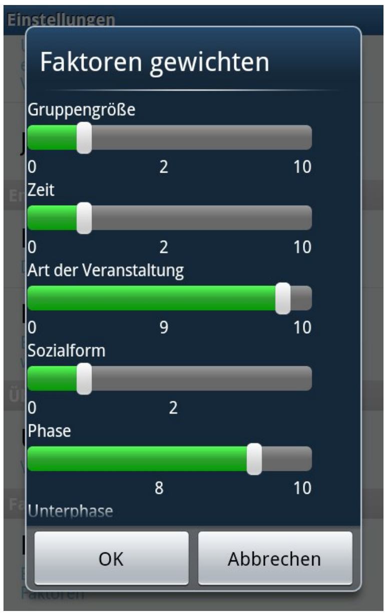
Figure 5.1.: A screen showing the recommended didactic methods along with up to three properties that had the most impact on the recommendation score.

The question that must be asked: When and how often should RecommendTask be called? Should something change on a device (for example, a user rates a didactic method or stores it as a favorite), this change is sent to the server via background tasks [26]. Thus, it seems reasonable to trigger

RecommendTask at these points as well, because the recommendation scores will be recalculated on the server-side. Yet should these changes take place on the server-side, updated information must be received as fast as possible. Therefore, RecommendTask is additionally activated at the usage start of MobiDics and triggers the background task periodically after a set period of time (currently, every ten minutes).

Weighted Properties and SendWeightsTask

The user has an opportunity to set her preferences about the method properties which are more important to her. For this case, a new screen has been implemented, which can be accessed from the already existing preference screen. The weights of eight different properties can be changed via eight seekbars, whereas at the beginning the default values are set to 5.0 (on a range from 0 to 10; see Figure 5.2). Once a user changes the weights, the values are stored in SharedPreferences. The final recalculation of the score, which happens on the device, considers the new weights.


Faktoren gewichten

Gruppengröße
0 2 10

Zeit
0 2 10

Art der Veranstaltung
0 9 10

Sozialform
0 2

Phase
8 10

Unterphase

Figure 5.2.: A screen showing the weights for various method properties.

Once a user has saved her preferences, this information has to be uploaded to the server as well, so that the web application displays exactly the same results. To account for this, a new background task, SendWeightsTask has been implemented. The task sends a HTTP POST request to the server which includes the weights. These values are then stored in a database table alongside the user name. As soon as the recommendation scores are recalculated, the updated values are used.

The SendWeightsTask is triggered every time when a user changes one of the weights. Furthermore, as with RecommendTask, the SendWeightsTask is also triggered at the beginning of the MobiDics usage and every time after a certain amount of time passes (currently, every ten minutes).

5.1.2. Web Site

As stated in the previous chapter, the sub-scores, and subsequently the final matching score, are calculated for the content-based RS on the server-side. For content-based recommendation algorithm information about the didactic methods is required. Also, the didactic methods a user has already shown preference for (i.e. rated or marked as favorite) need to be known. Information about all of these aspects are stored in different tables in a central database. Thus, a new view is created for each user to have the necessary information in one place. The benefits of creating views is argued in [26]. We simply state here that we adapt our approach to an already existing solution. For every user we create a view by a following query:

```
CREATE OR REPLACE VIEW methodsUserView AS
        SELECT r.*,  if( f.method_id IS NULL, 0, 1) AS favorite
        FROM myRatingView AS r
        LEFT JOIN FAVORITES_TABLE as f
        ON (r.id=f.method_id AND f.username='username')
```

The resulting view contains every method that a user has rated or stored as a favorite.

To calculate the final score, the weights of the didactic properties have to be considered. These can currently be set only on a mobile device and uploaded to the server where they are stored in a database table *ddk_factorweights* (see Subsection 5.1.1). The final matching score for a CB approach is then the weighted sum of all sub-scores. All sub-scores and the final score are stored in a database table *dev_ddk_recommend* in the central database.

The didactic methods are presented on a web site in several categories (e.g., top rated didactic methods, user's favorite didactic methods) (Figure 3.1). The additional category ("Recommended") to display the recommended didactic methods was created. When a user navigates to this category, the recommended didactic methods sorted by the matching score are presented.

5.2. Collaborative Filtering Approach

5.2.1. Android

The adaptation of the Slope-One algorithm of the collaborative filtering approach was discussed in Section 4.2. Further, it was mentioned that the Slope-One scores are calculated on the server-side. This matching score is stored in the same server database table as the scores and sub-scores for CB approach (*dev_ddk_recommend*).

The Slope-One scores have to be downloaded from the server on a mobile device. To improve the efficiency, the response for the RecommendTask HTTP POST request was adjusted so that the downloaded XML file now contains Slope-One scores (see Subsection 5.1.1). How the Slope-One score is combined with the matching score resulted from the CB approach will be discussed in Section 5.3.

5.2.2. Web Site

The calculation of the Slope-One score occurs on the server-side. All relevant functions that are needed for this are included in the file *slopeOne.lib.php*. The calculation is divided into two steps:

The Deviation Matrix has to be created. The function *calculateSlopeOnePair(i,j)* calculates the entry $\Delta_{i,j}$. Basically, the function uses a nested SQL query which determines the value as follows (explanation from the inside to the outside):

(i) all users that have rated didactic method i *and* j are determined.

(ii) the view (i) is joined with table *dev_ddk_ratings* to determine the respective ratings and method IDs.

(iii) the average rating for each method ID is determined from view (ii) and the difference is calculated.

The result is returned by the function, or NULL is returned in case no result was available (e.g. if there is no user who has rated didactic methods i and j).

The function *createDeviationMatrix* creates the whole deviation matrix. This is basically achieved by iterating over all possible method IDs i and j and calling *calculateSlopeOnePair*. However, as the matrix is anti-symmetric, the computational complexity can be reduced to calculating only one of the pairs. The database table *ddk_deviation_matrix* stores the resulting matrix $\Delta_{i,j}$.

Because calculating the deviation matrix each time a rating has changed may use too many of the server system resources, particularly if many users are registered and many methods are available, it might be desirable to only recalculate the matrix at certain points in time. In order to provide this functionality, the PHP file *create_deviation_matrix.php* can be executed any time which will update the matrix.

Calculating the Slope-One Score: Once the deviation matrix is calculated, the Slope-One scores can be determined. The function *calculateSlopeOne(i)* calculates the Slope-One score for the current user for method ID i. The function uses an SQL query to get the current user ratings for method ID $j \neq i$ and the corresponding entries in the deviation matrix $\Delta_{i,j}$. From this data the estimated Slope-One rating can be calculated and is finally converted from rating scale [1, 5] to a score [0,1] and returned.

The function *calculateSlopeOneForUser* calculates all Slope-One scores (i.e. for all methods that were not rated by the user; if possible) for the current user. The scores are then stored in the server database table *dev_ddk_recommend*.

5.3. The Combination of Both Approaches

The CB and CF approaches have to be combined to get the weighted hybrid RS (see Section 2.4). The combination is a weighted sum of the scores derived from the CB and CF approaches. In the current version both approaches have the same weight. For future work the weights can be adapted according the users' feedback.

One detail has to be considered: the range of the sub-scores is [0,1] and the range of the weights is [0,10]. As there are in total eight sub-scores, their weighted sum will be in the range [0,80]. However, the range of the Slope-One score is [0,1]. Thus, the matching score for content-based approach has to be normalized. The combined matching score is then calculated.

5.4. Enhancements

In this section further enhancements that were added to MobiDics application functionality are presented.

Welcome Screen: Upon running the MobiDics application for the first time, a new screen has been added, which provides the overall information about MobiDics application. After reading the information a user can continue to another screen to set up the MobiDics application. The old welcome screen now serves as a set up screen.

Help Screen: The ActionBar of the MobiDics application consisted of *Sync/Update* and *Search* buttons [26]. The ActionBar was enhanced with a *Help* button. After clicking the button, the user can choose between several menu points and read background information about MobiDics, useful information about the didactic methods, or the instructions for the usage of MobiDics application.

Chapter 6.

User Study

A user study has been conducted to receive feedback from the participants on usability of already implemented features of MobiDics Android application. This chapter describes the design of the user study and both presents and discusses the subsequent results.

6.1. Preparation Phase of the User Study

For the purpose of receiving user feedback on the application usage, two questionnaires have been created. The start questionnaire requested participants' demographic data, their understanding/familiarity with didactic methods in addition to their experience in applying them. The end questionnaire collects feedback about the MobiDics application itself and its various features.

To log the usage of the MobiDics application the Questionnaire application is used [28].

6.1.1. Data Collection

The Questionnaire application allows for easily creating questionnaires that can be answered via mobile devices. This is helpful, considering the rising number of people using smart phones as they can use the application whenever they want.

In our case, the main reason for using the Questionnaire application is the opportunity to log user data, i.e., to log which activities a user conducts. The logged data is stored locally in a file and automatically send to the server every time a user employs the Questionnaire application. In case the automatic upload fails, users can trigger a manual upload very easily from the preference menu.

From the conductor's point of view, the following steps must be taken in order to have an operable Questionnaire application (for more detailed information please see [28]):

1. A conductor has to install the server and start it.

2. When the server is running correctly, a conductor can use the HTML page *QuesionnaireAdmin* and create questionnaires. This is also the place, where all names of applications and specific Activities that should be logged, can be listed.
3. After the questionnaires were created, a conductor can set further specific parameters using the HTML page *AppCreator*, which is responsible for creating a zip file that contains the questionnaires.
4. The downloaded files have to be imported into Eclipse. Once the project is compiled, the application package file (APK) is created. This APK can be sent to the participants of the user study.

The Questionnaire application has several modes: (i) it can be triggered automatically every time a logged application is used, (ii) based on a given time interval and (iii) manually. For our purposes the manual mode is sufficient.

The following Activities of the MobiDics application were logged:

- *Main:* Main Activity, called at application start.
- *Login:* called at the login screen.
- *MethodsPager:* this Activity represents all pages where the didactic methods are listed by some criteria, e.g., favorite, recently viewed or top rated. It is not possible to determine which page a user has viewed exactly.
- *Method:* called when a single didactic method is being observed.
- *SingleImage:* called when a user clicks on an image.
- *MyAccount:* Activity handling a screen where a user's personal information is displayed.
- *Search:* called when a user clicks the “search” button.
- *Preferences:* called when a user clicks on “preferences” via menu.
- *HelpViewer:* called when a user clicks the “help” button.

6.1.2. Pilot Study

A pilot study was conducted to ensure that the usage of the MobiDics application was being logged. APKs of both applications (Questionnaire and MobiDics app.) were sent to two users, who were asked to install them and use them for 72 hours. During the pilot study we were able to identify shortcomings of the Questionnaire application for our purposes, namely:

Two shortcomings of the Questionnaire app were identified during the Pilot Study: first, in order to send data to the server, one of the steps must be taken: (i) either a user answers the questionnaire every time the application is used, which guarantees that the log-data will be uploaded to the server along with the answers, or (ii) a user has to trigger manual upload from the preference menu. This could be a problem, because participants would be forced to use two applications during the study. Therefore, we asked the participants of our main study to trigger the Questionnaire application at the beginning of the study and at the end of it.

Secondly, it was noted that detailed information within the Activity is not available. For example, if the Activity *Method* is logged there is no information about the exact didactic method that was viewed. This could be very helpful for distinguishing the most viewed methods by a user.

6.2. Main User Study: Results and Discussion

A four-week study was conducted. The participants were asked to install the APKs of MobiDics and Questionnaire applications and run them upon receipt. The participants had to answer two questionnaires, which were provided online[1], one prior to the application usage and the second at the end of the conducted study. Furthermore, upon running the Questionnaire application, participants were asked to create an alias name. They were informed that if something went wrong, a general email containing the alias name would be sent to all participants. The results of the study are presented below.

6.2.1. Start Questionnaire

To gather general information about participants' demographic data, their knowledge in didactic methods and experience in using them, the participants had to answer the following questions:

1. What is your gender?
2. How old are you?
3. How many years of experience in university teaching do you have?
4. What is your current occupation?
5. In which faculty do you teach?
6. How would you estimate your knowledge of didactic methods?
7. How confident are you in applying didactic methods?
8. Do you think that using didactic methods is beneficial for students?

[1]`https://www.soscisurvey.de` (last visited: 21.03.13)

9. How do you learn about new didactic methods?
10. How do you choose didactic methods?

It should be noted that questions six and seven were asked both in start and end questionnaires to detect differences before and after the MobiDics usage.

Participants

A total of 21 participants answered the start questionnaire. Ten participants answered the end questionnaire, although, only eight completed it in its entirety. While statistical analysis of the start questionnaire considers the answers from all 21 participants (except questions six and seven), analysis of the end questionnaire only considers the answers of the eight participants who successfully completed the questionnaire.

Table 6.1.: Descriptive statistics for age and teaching experience.

	N	**Minimum**	**Maximum**	**Mean**
age	21	25	48	33.57
experience	21	0	28	8.14

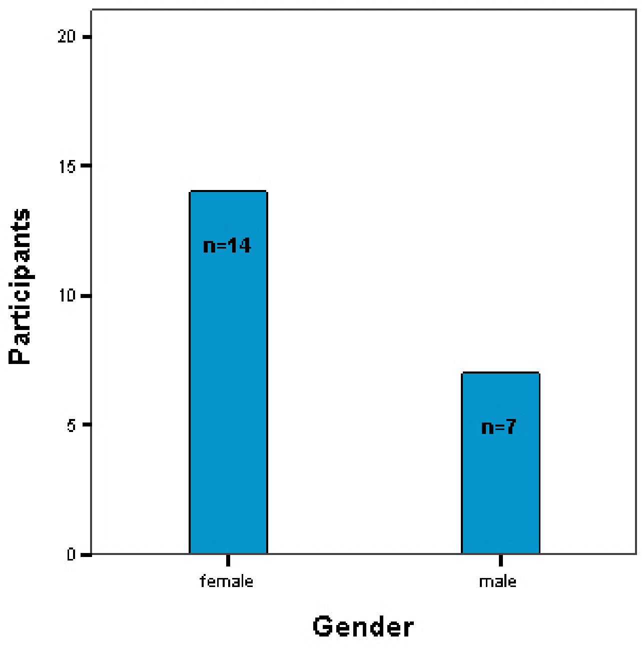


Figure 6.1.: Gender distribution of participants.

Out of the 21 participants, 14 were female and 7 were male (see Figure 6.1). Table 6.1 shows descriptive statistics regarding age and teaching experience of the participants. The youngest participant was 25 years old, the oldest was 48 years old. The average age was 33.6 years and the average teaching experience was 8 semesters. One participant did not have any teaching experience, while the longest teaching experience was 28 semesters (14 years).

Participants also had to state in which faculty they currently work (question 5, see Figure 6.2). 29% of the participants work at the Faculty of Psychology and Educational Sciences, 19% at the Medical Faculty and 5% of the participants individually represent the Faculties of Veterinary medicine, Languages and Literatures, Social Sciences, Mathematics and Statistics, Computer Sciences, and Faculty of Sports and Health Sciences. The remaining 24% (six participants) work at other faculties. Two of these six participants work at department for Teaching and Learning in Higher Education at the Technische Universität München. Two further participants work at TUM School of Education. One participant works at the Faculty of Aerospace Engineering. One participant is at the management board of a university.

47% of all participants work in faculties related to educational sciences. This could mean that individuals who work at educational faculties found the study more interesting.

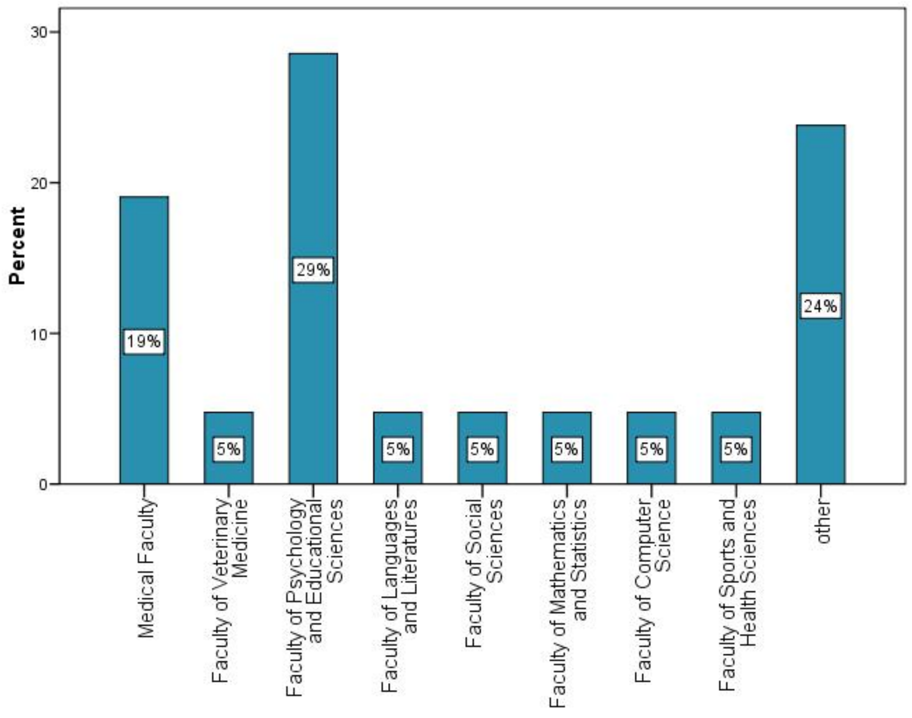


Figure 6.2.: Distribution of participants across faculties.

Questions Regarding the Knowledge and Execution of Didactic Methods

The participants had to answer questions 6 through 10 on a five-level Likert scale. The possible answers were: 1 = strongly disagree, 2 = disagree, 3 = neutral, 4 = agree, 5 = strongly agree. For statistical analysis SPSS Statistics (version 15.0) software was used. The following applies to all figures that contain box plots throughout this chapter: the blue box indicates the answers between lower and upper quartiles. The black horizontal line shows the median. The vertical lines going out of the box ("whiskers") indicate lower (twenty-fifth percentile) and upper quartiles (seventy-fifth percentile) of the data. The whiskers connect the minimum and maximum values to the box. The circles in the figures indicate "outliers". "Outliers are values that are more than 1.5 length from upper and lower quartiles. The stars mark "extreme values" that are more than three box lengths from lower and upper quartiles" [29].

In order to assess the benefits for applying didactic methods for students, the following statement had to be rated (question 8):

(8a) Diversity of didactic methods is beneficial.

(8b) Students participate actively in execution of new didactic methods.

(8c) Comprehension of new material is better when didactic methods are used.

(8d) Students are more likely to discuss new material with each other when a didactic method was used in the class.

Table 6.2.: Descriptive statistics for question 8 in the start questionnaire (self-assessment of benefits for students using didactic methods during class).

Statement	Minimum	Maximum	Mean	Median	Std. deviation
(8a)	2	5	4.43	5.00	0.811
(8b)	3	5	3.95	4.00	0.590
(8c)	3	5	4.10	4.00	0.700
(8d)	2	5	3.67	4.00	0.730

The result of these statements are shown in Figure 6.3. Descriptive statistics of these statements are presented in Table 6.2. With an average value of 4.43 ± 0.811, participants agreed with the statement (8a). One participant disagreed with this statement and is marked as an outlier. Overall, 57% agreed strongly and 33% agreed to the statement. Therefore, it could be assumed that the majority of the participants think that applying didactic methods during the class is beneficial for students.

Furthermore, participants generally agreed with statement (8b) (mean: 3.95 ± 0.590, median: 4.00). 19% of the participants agreed strongly, 67% agreed, and 14% (n=3) were neutral about this statement. These three neutral responses were marked as extreme outliers. Overall, the majority of the participants have observed an active participation of students in new didactic methods during a class.

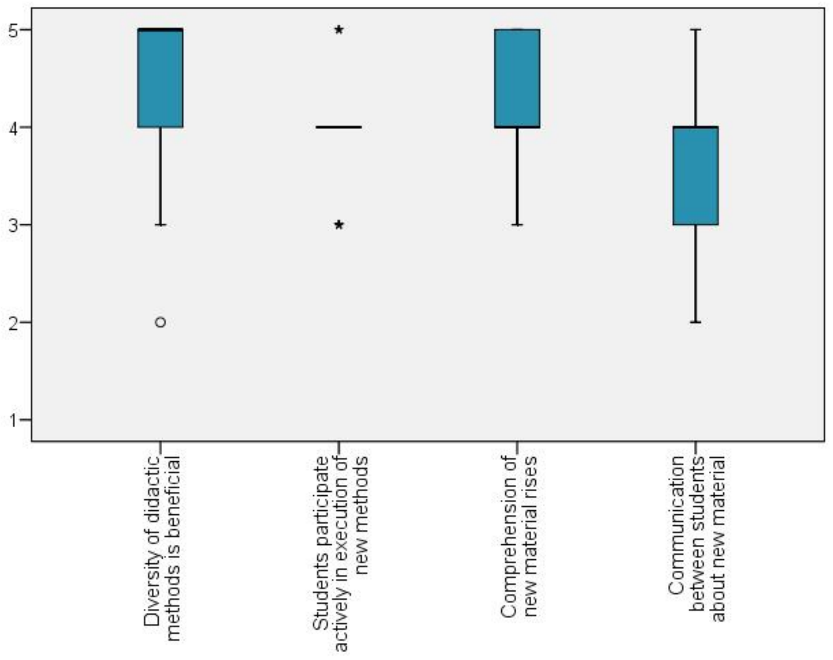


Figure 6.3.: Box plots representing self-assessment of benefits for students using didactic methods during class. First box plot from the left represents the distribution of the answers to statement (8a); second box plot: statement (8b); third box plot: statement (8c); fourth box plot: statement (8d). Box: interquartile range (25-75 percentage); Whisker: the range (excluding outliers and extreme values); Circle: outlier; Star: extreme value; Black line: median; 1 = "strongly disagree"; 2 = "disagree"; 3 = "neutral"; 4 = "agree"; 5 = "strongly agree".

With an average value of 4.10 ± 0.700 and median value of 4.0, participants agreed to a statement that students understand new material better when didactic methods are used to explain them (statement (8c)). 19% (n = 14) of the participants were neutral to this statement. However, considering that 52% of participants agreed and 29% of participants strongly agreed to this statement, it can be assumed that participants think that didactic methods help in comprehension of new material.

With an average value of 3.67 ± 0.730, participants (median: 4.0) slightly tended to agree to the statement (8d). One participant disagreed to the statement (8d) and there were no marked outliers for this statement. Interestingly, the range of the responses to (8d) was considerably wider spread than with the previous statements in this set. Overall, agreement with statements (8a) to (8c) was stronger than with statement (8d).

In question 9 the participants were asked to state how they learned about new didactic methods.

For this reason they had to rate the following statements:

(9a) I learn about new didactic methods from colleagues.

(9b) I learn about new didactic methods from books.

(9c) I learn about new didactic methods in the internet.

(9d) I learn about new didactic methods while attending my colleagues' classes.

(9e) I invent new didactic methods myself.

(9f) I learn about new didactic methods at vocational trainings.

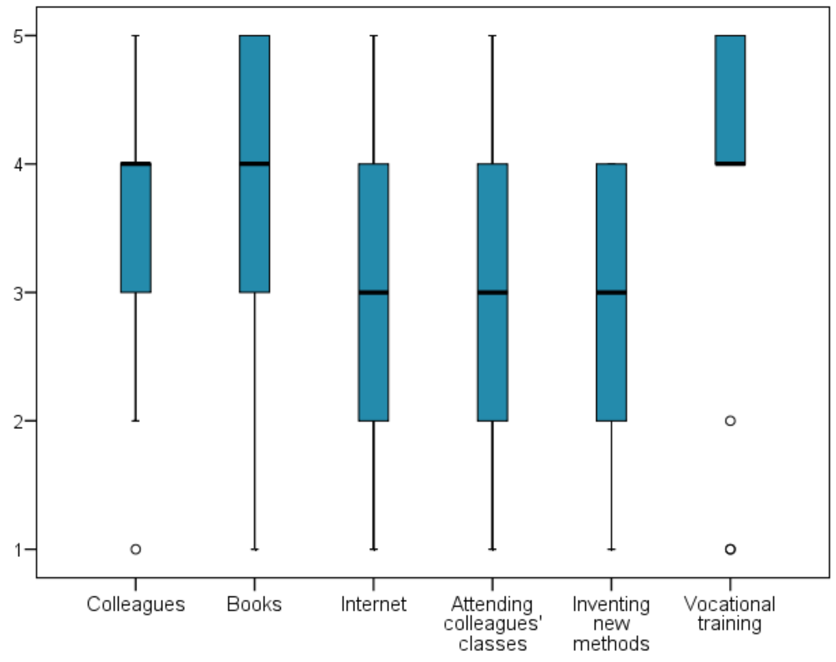


Figure 6.4.: Box plots represent the sources where the participants learn about new didactic methods. The first box plot on the left represents the distribution of the answers of the statement (9a). The last box plot (on the right side) represents the answers to the statement (9f).

The answers are shown in Figure 6.4. The answers for statements (9b), (9c) and (9d) were wider spread than those of the other statements. According to the median, participants agreed to statements (9a), (9b) and (9f). Participants were on average neutral to statements (9c) (mean: 3.19 ± 1.327), (9d) (mean: 2.90 ± 1.261) and (9e) (mean: 2.95 ± 1.024) (see Table 6.3).

Participants tend to agree more to the statement (9f) than to other statements (median: 4.0, mean: 4.0 ± 1.225). 48% of participants agreed to this statement and 38% of participants agreed strongly to it. Overall, it seems that colleagues, books and vocational trainings are more preferred sources for the participants than other sources.

Table 6.3.: Descriptive statistics for question 9 of the start questionnaire (for sources where the participants learn about new didactic methods).

Statement	**Minimum**	**Maximum**	**Mean**	**Median**	**Std. deviation**
(9a)	1	5	3.48	4.00	1.167
(9b)	1	5	3.67	4.00	1.197
(9c)	1	5	3.19	3.00	1.327
(9d)	1	5	2.90	3.00	1.261
(9e)	1	4	2.95	3.00	1.024
(9f)	1	5	4.0	4.00	1.225

In order to comprehend which criteria are more important for participants in choosing didactic methods, they were asked to rate the following statements (question 10):

(10a) I choose didactic methods depending on teaching goals for a given class.

(10b) I choose didactic methods depending on time I have for a class.

(10c) I choose didactic methods depending on students who attend my class.

(10d) I choose didactic methods depending on topics of a class.

(10e) I choose didactic methods depending on my experience with this method.

(10f) I choose didactic methods depending on my expertise with this method.

(10g) I choose didactic methods depending on a composition of a group of students (group type).

(10h) I choose didactic methods depending on how confident I feel in class.

Table 6.4.: Descriptive statistics for question 10 of the start questionnaire (evaluating the impact of every criteria while choosing a didactic method for a class).

Statement	**Minimum**	**Maximum**	**Mean**	**Median**	**Std. deviation**
(10a)	2	5	4.05	4.00	0.921
(10b)	3	5	4.24	4.00	0.625
(10c)	1	5	3.76	4.00	1.136
(10d)	3	5	4.05	4.00	0.669
(10e)	1	5	3.86	4.00	1.153
(10f)	1	5	3.43	4.00	1.248
(10g)	1	5	3.76	4.00	1.091
(10h)	1	5	3.29	4.00	1.347

Results of these statements are shown in Figure 6.5. With the median value of 4.0 for every statement, participants agreed that they consider every criteria while choosing the didactic methods for their classes. However, in comparison to other criteria, teaching goals (10a), time (10b) and topics of the class (10d) tend to have on average more impact on choosing the didactic methods

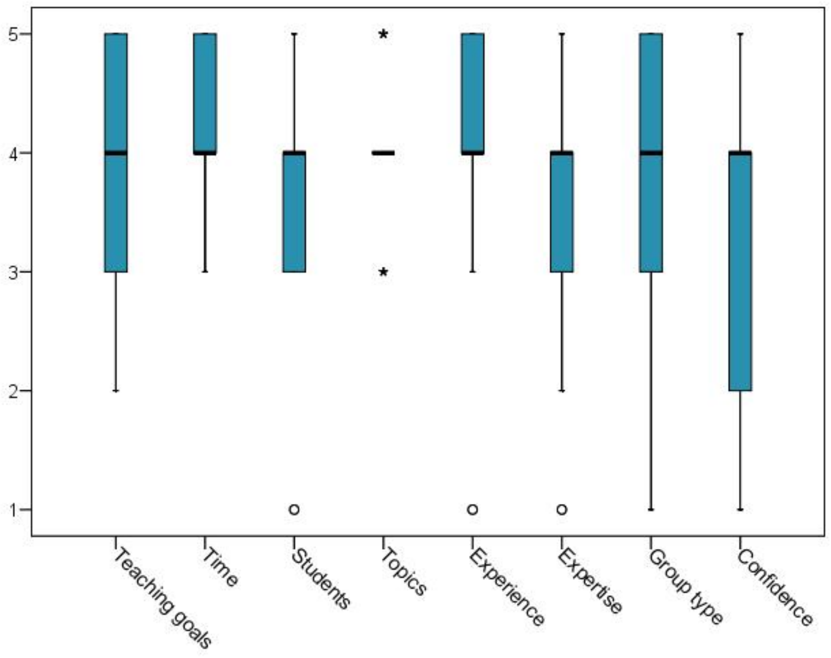


Figure 6.5.: Box plots indicate the impact of every criteria while choosing a didactic method for a class. “Teaching goals” = statement (10a); “Time” = statement (10b); “Students” = statement (10c); “Topics” = statement (10d); “Experience” = statement (10e); “Expertise” = statement (10f); “Group type” = statement (10g); “Confidence” = statement (10h); Circle: outliers; Star: extreme values.

with mean values of 4.05 ±0.921, 4.24 ±0.625 and 4.05 ±0.669, respectively (see Table 6.4). Additionally, participants were on average rather neutral to the statement (10h) (mean: 3.29 ±1.347). The range of answers regarding statements (10f) and (10h) were only slightly wider spread (std. deviation 1.248 and 1.347, respectively) than for the other statements.

6.2.2. End Questionnaire

The participants were asked to fill out the end questionnaire at the end of the user study. As it was stated before, ten participants answered the end questionnaire, although, only eight completed it in its entirety. The following results are based on the answers of eight participants who successfully completed the end questionnaire. The questions are presented below:

1. Have you been satisfied with the functionality of MobiDics?
2. How helpful were the different MobiDics functions?
3. Were you satisfied with the descriptions of didactic methods?
4. Were you satisfied with the background information of MobiDics?

5. How often did you search/view for different didactic-method properties?
6. How would you estimate your knowledge of didactic methods after the usage of MobiDics?
7. How confident are you in applying didactic methods after the usage of MobiDics?
8. How often did you search for a suitable didactic method for your class?
9. Did you use "advanced search" to find didactic methods? Please specify the criteria which you used for "advanced search".
10. How well is an app/website suited for transferring knowledge about didactic methods?
11. What cannot or should not be transferred via an app/website?
12. What do you think would help you to become more confident in applying didactic methods?
13. What do you think one has to know to be an expert in didactic methods?
14. Overall, how satisfied are you with MobiDics?

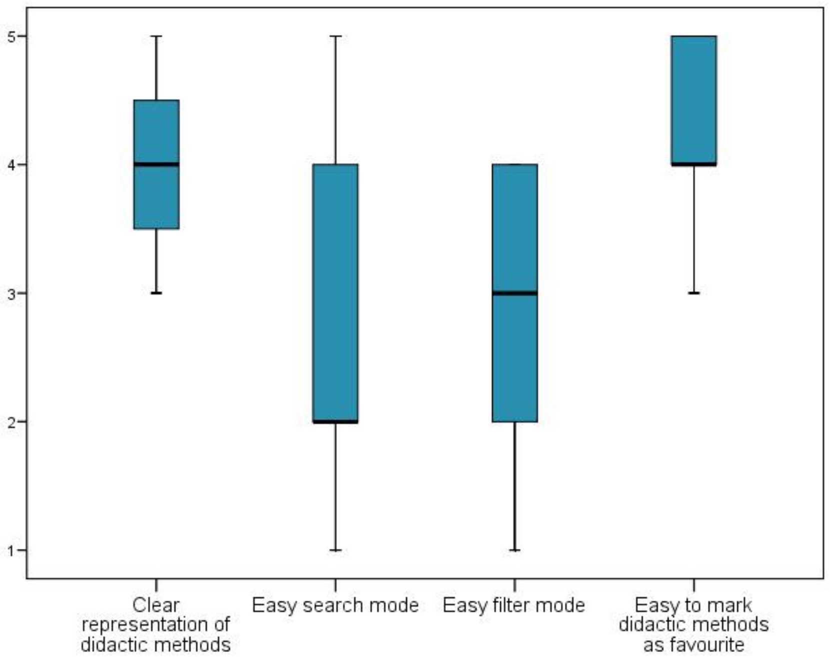


Figure 6.6.: Box plots show the distribution of answers for statements (1a) to (1d). First box plot (on the outer left side): statement (1a); Second box plot: statement (1b); Third box plot: statement (1c), fourth box plot: statement (1d); Box: interquartile range; Whiskers: upper and lower quartiles; The black line: median; 1 = strongly disagree; 2 = disagree; 3 = neutral; 4 = agree; 5 = strongly agree.

Table 6.5.: Descriptive statistics for question 1 of the end questionnaire (evaluating how easy certain MobiDics features were to use).

Statement	**Minimum**	**Maximum**	**Mean**	**Median**	**Std. deviation**
(1a)	3	5	4.00	4.00	0.756
(1b)	1	5	2.75	2.00	1.389
(1c)	1	4	2.88	3.00	1.126
(1d)	3	5	4.25	3.00	0.707

In question 1, participants were asked to rate the following statements about the MobiDics functionality:

(1a) The didactic methods were represented clearly.

(1b) The search mode was easy to use.

(1c) The filter mode was easy to use.

(1d) It was easy to mark a didactic method as favorite.

Results of these statements are shown in Figure 6.6. The descriptive statistics regarding these statements are presented in Table 6.5. Participants on average agreed that didactic methods were represented in a comprehensible way with the mean value of 4.0 ± 0.756. The overall range of answers to the statement (1a) varied from "neutral" to "strongly agree". Participants also agreed to the statement (1d) with the mean value of 4.25 ± 0.707. However, one participant stated that it was possible to accidentally mark a wrong method as favorite while scrolling.

With an average value of 2.75 ± 1.389, participants were rather neutral to the statement that the search mode was easy to use (1b). The median value of the statement was "disagree". The distribution of the answers is moderately skewed left for this statement. The statement (1c) has a mean value of 2.88 ± 1.126. Thus, participants tend to be rather neutral to this statement.

Overall, it can be assumed that participants found the whole representation of a didactic method comprehensible and had only minimal difficulties with marking didactic methods as favorite, whereas search and filter modes were more difficult to use.

In the next question (question 2), participants were asked to rate the following MobiDics functions depending on how helpful they were:

(2a) Representation of a didactic method was helpful.

(2b) The search mode was helpful.

(2c) The filter mode was helpful.

(2d) Marking a didactic method as favorite was helpful.

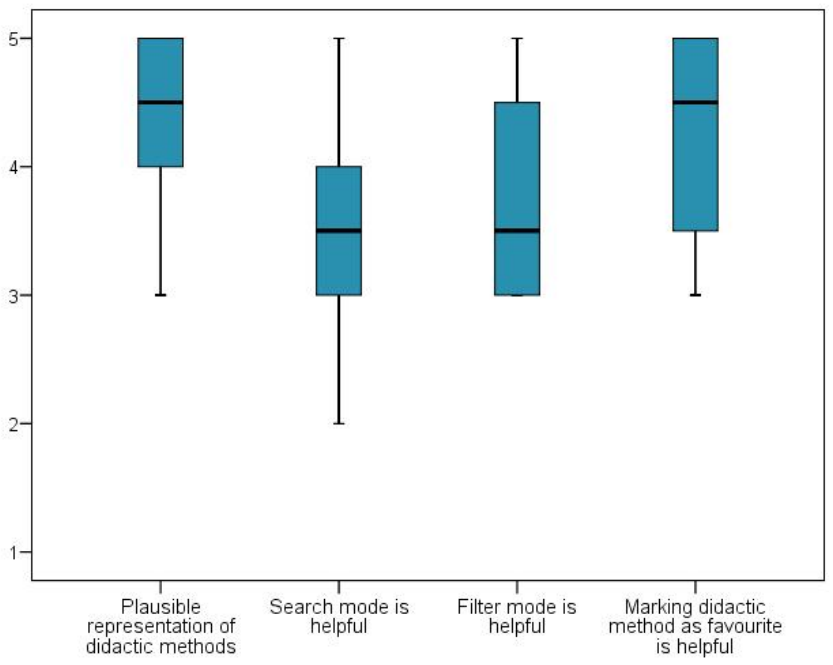


Figure 6.7.: Box plots represent the distribution of answers regarding the statements (2a)(first box plot from the left) to (2d) (first box plot from the right). Box: interquartile range; Whiskers: lower and upper quartiles; The black line in the box: the median; 1= strongly disagree; 2= disagree; 3= neutral; 4= agree; 5= strongly agree.

On average (mean: 4.38 ± 0.744) participants agree to the statement (2a) (Tabelle 6.6). The distribution of the answers is moderately skewed left (skewness = -0.824). The median value of the statement is 4.5, thus, the value is between "agree" and "strongly agree" (see Figure 6.7).

With an average value of 3.50 ± 0.926, there is a slight trend that participants agree with the statement (2b). The range of the answers for this statement is wide spread, varying from "disagree" to "strongly agree". The distribution of the values is approximately symmetric for this statement.

Regarding the statement (2c), participants tend to agree that the filter mode is helpful with a mean value of 3.75 ± 0.886. The distribution is moderately skewed right. The median value of the statement (3d) is 4.50, indicating that participants slightly tend to agree strongly to the statement.

Overall, marking didactic methods as a favorite was on average more helpful than search and filter modes with mean values of 3.50 and 3.75, respectively. One participant additionally stated that pictures were very helpful. Also, participants tend to find the representation of methods helpful. A comparison between these answers and those of the previous question show that while participants appeared to have some difficulties using search and filter modes (see the results for question 1),

Table 6.6.: Descriptive statistics for question 2 of the end questionnaire (investigating how helpful the certain MobiDics features were).

Statement	**Minimum**	**Maximum**	**Mean**	**Median**	**Std. deviation**
(2a)	3	5	4.38	4.50	0.744
(2b)	2	5	3.50	3.50	0.926
(2c)	3	5	3.75	3.50	0.886
(2d)	3	5	4.25	4.50	0.886

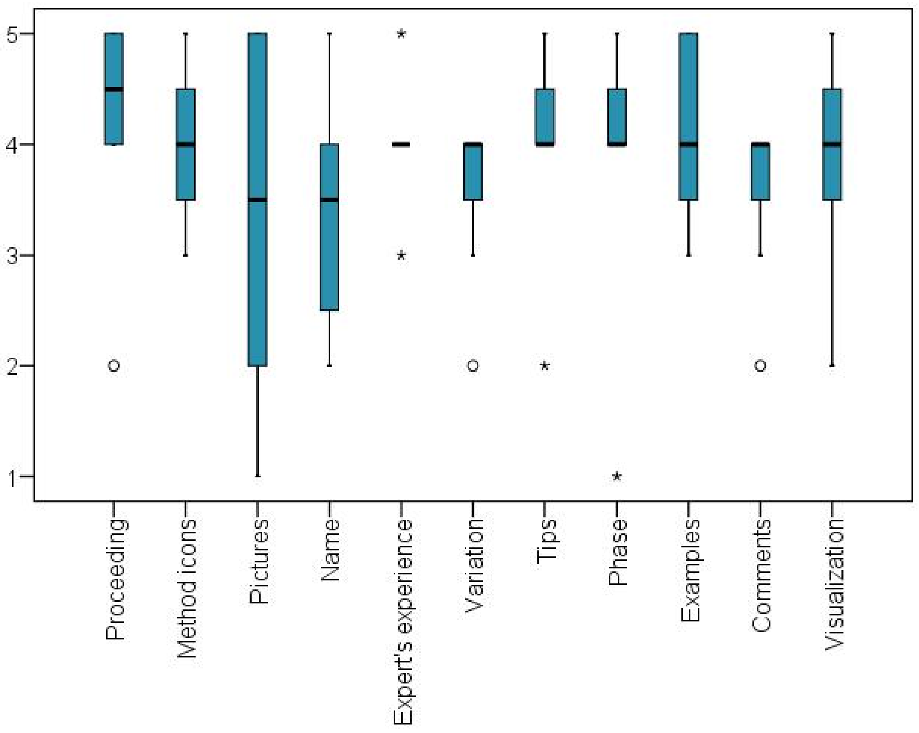


Figure 6.8.: Box plots indicate how content the participants were with the description of the didactic method (question 3)). Circle: outliers; Star: extreme values.

they tend to agree that these features are helpful.

In question 3, participants were asked about the properties of didactic methods and how well they were described/how helpful these properties are. Participants could state if they had not read the given property. Thus, possible answers do not only include 1= "strongly disagree" to 5= "strongly agree" but also -1= "did not read". The results are represented in Figure 6.8.

All eight participants stated that they have read/viewed the following properties of a didactic method: (i) description of the proceeding, (ii) individual method icons, (iii) pictures, (iv) name of the didactic method, (v) examples, and (vi) visualization. On average, participants found description of the proceeding of the didactic method helpful (mean: 4.25 ± 1.035). The individual icons and examples were on average helpful with the mean values 4.0 ± 0.756 and 4.13 ± 0.835, respectively. With the average value of 3.88 ± 0.991, participants tend to agree that they found visualization helpful. The participants were rather neutral to the statements that pictures (mean: 3.38 ± 1.598) and the name of the didactic method (mean: 3.38 ± 1.061) were useful.

Of the eight participants seven read/viewed the following properties: (I) variation of a didactic method, (II) tips, (III) phase, and (IV) comments. Based on the answers of these seven participants, tips and phase were helpful with mean values of 4.0 ± 1.00 and 3.86 ± 1.345, respectively. Variation and comments both had mean values 3.57 ± 0.787, indicating that there is a slight trend to agree that these properties are helpful for the participants.

Finally, two participants stated that they have not read expert's expertise. Based on the answers of the six participants, they agree to the statement that this property is helpful with a mean value of 4.0 ± 0.756.

Question 4 covered the statements about the background information that was provided for MobiDics application users. The following statements had to be rated:

(4a) Information about systematisation was helpful.

(4b) AVIVA scheme was helpful.

(4c) Instructions were helpful.

(4d) Help was helpful.

Table 6.7.: Descriptive statistics for question 4 of the end questionnaire (evaluating the background information of MobiDics).

Statement	**Minimum**	**Maximum**	**Mean**	**Median**	**Std. deviation**
(4a)	4	5	4.50	4.50	0.744
(4b)	4	5	4.50	4.50	0.926
(4c)	4	5	4.33	4.00	0.886
(4d)	5	5	5.00	5.0	–

Participants had the opportunity to state if they have read the background information. The results are represented in Figure 6.9. The descriptive statistics are presented in Table 6.7. Only two participants had read the information about systematisation. One of them agreed to the statement (4a) and one of them agreed strongly to it. Further, also only two participants had read the information about AVIVA scheme and agreed to the statement (4b) (mean: 4.5 ± 0.926). Overall, three participants read the instructions and found them helpful with the mean value of 4.33 ± 0.886. Lastly, only one participant used the help function and found it helpful (mean: 5.00).

While all four statements had a high average value, one has to keep in mind that the sample group of participants for these statements was rather small. Therefore, it is difficult to draw any further conclusions. It is unclear due to the lack of data whether a correlation can be made between the necessity of background information and the number of participants who read it, and if the results would have been different had all participants actually read the provided information.

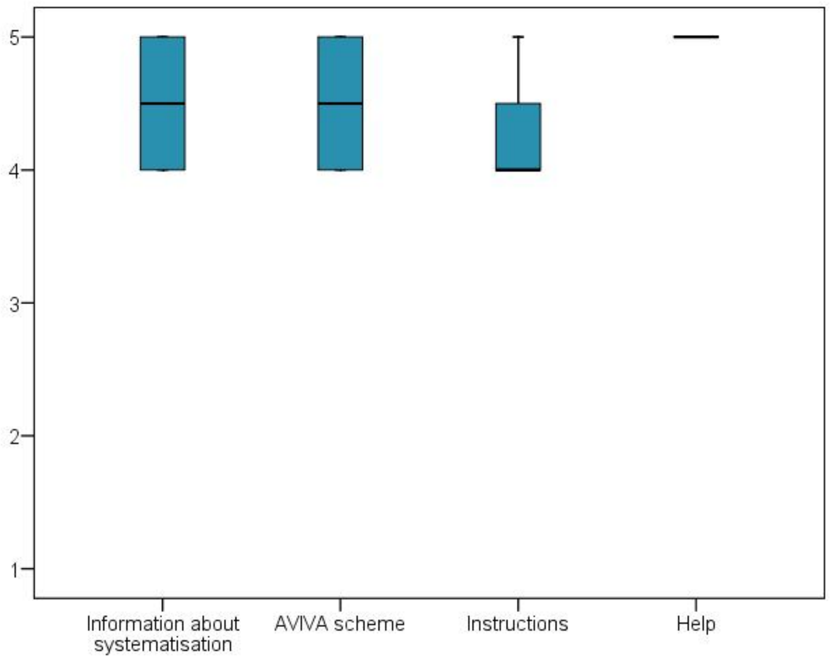


Figure 6.9.: Box plots indicate how helpful the background information was. (Statements (4a) to (4d)). 1 = strongly disagree; 2 = disagree; 3 = neutral; 4 = agree; 5 = strongly agree.

In question 5, the participants were asked how often they have searched/viewed the different properties of a didactic method. The possible answers were: 1 = "never", 2 = "rarely", 3 = "occasionally", 4 = "fairly often", 5 = "very often". The results are shown in Figure 6.10.

On average, participants stated that they have searched/viewed the proceeding (mean: 3.88 $\pm 1,246$) and top rated methods (mean: 4.00 ± 0.756) fairly often. With an average value of 3.50 ± 0.756 and 3.50 ± 0.926, there is a slight trend to agree to have searched/viewed the new/updated methods and tips, respectively. On average, examples (mean: 3.25 ± 0.707), variation (mean: 3.13 ± 1.246), visualization (mean: 3.00 ± 0.756), and phase (mean: 3.25 ± 1.282) were read/viewed rather occasionally. The comments (mean: 2.50 $\pm 1,246$) and expert's experience (mean: 2.63 ± 1.188) were the properties that were on average rarely searched/viewed.

In question 8, participants had to state how often they have searched specific properties of didactic methods. Overall, thirty properties of a didactic method were listed. The answers to question 8

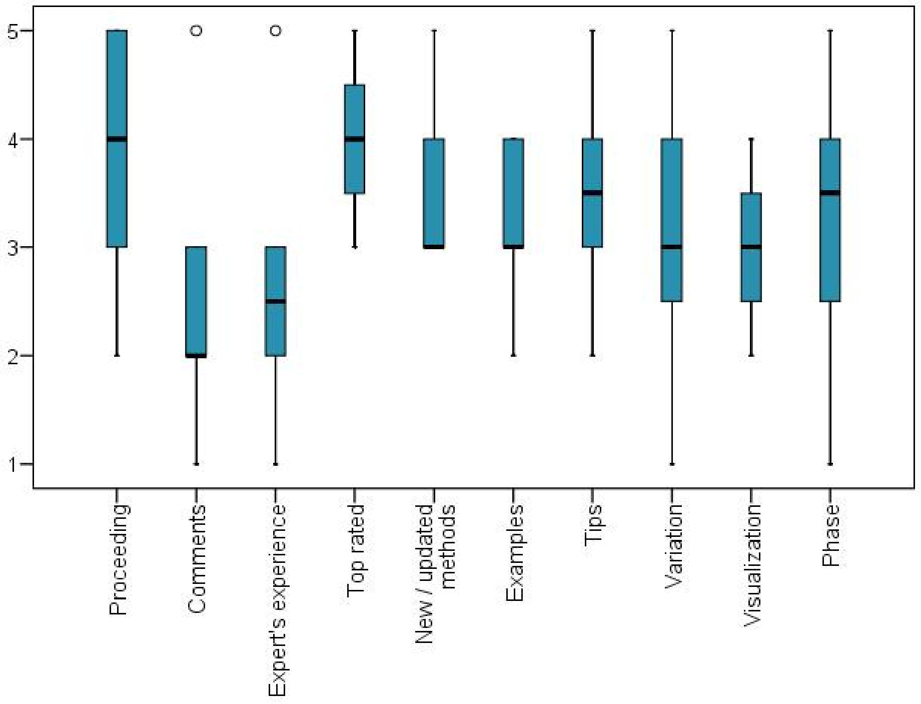


Figure 6.10.: Box plots indicate how often the participants searched/viewed different properties (question 5). 1 = never; 2 = rarely; 3 = occasionally; 4 = fairly often, 5 = very often; Circle: outliers.

are represented in Figure 6.11. According to participants, the most frequently searched properties were social form (mean: 3.75 ± 1.282) and course type (mean: 3.25 ± 1.035). The participants searched occasionally the properties composition of participants (mean: 2.75 ± 1.282) , user rating (mean: 3.00 ± 1.512) and group size (mean: 3.13 ± 1.458).

In addition to question 8, the participants were also asked if they used the "combined search" function of MobiDics. In an open-end question, they had to state which criteria they have combined. One participant had combined "language and phase", "phase and group size", "phase and course type", and "phase and time". Because only one participant covered this question, we can not predict which combinations of criteria could be interesting for others.

Question 14 covered participants' overall impression that MobiDics application made on them. The participants had to rate the following statements:

(14a) MobiDics is a helpful tool for improving teaching quality.

(14b) Technical application of MobiDics was effortless.

(14c) I think MobiDics will be used by teaching staff.

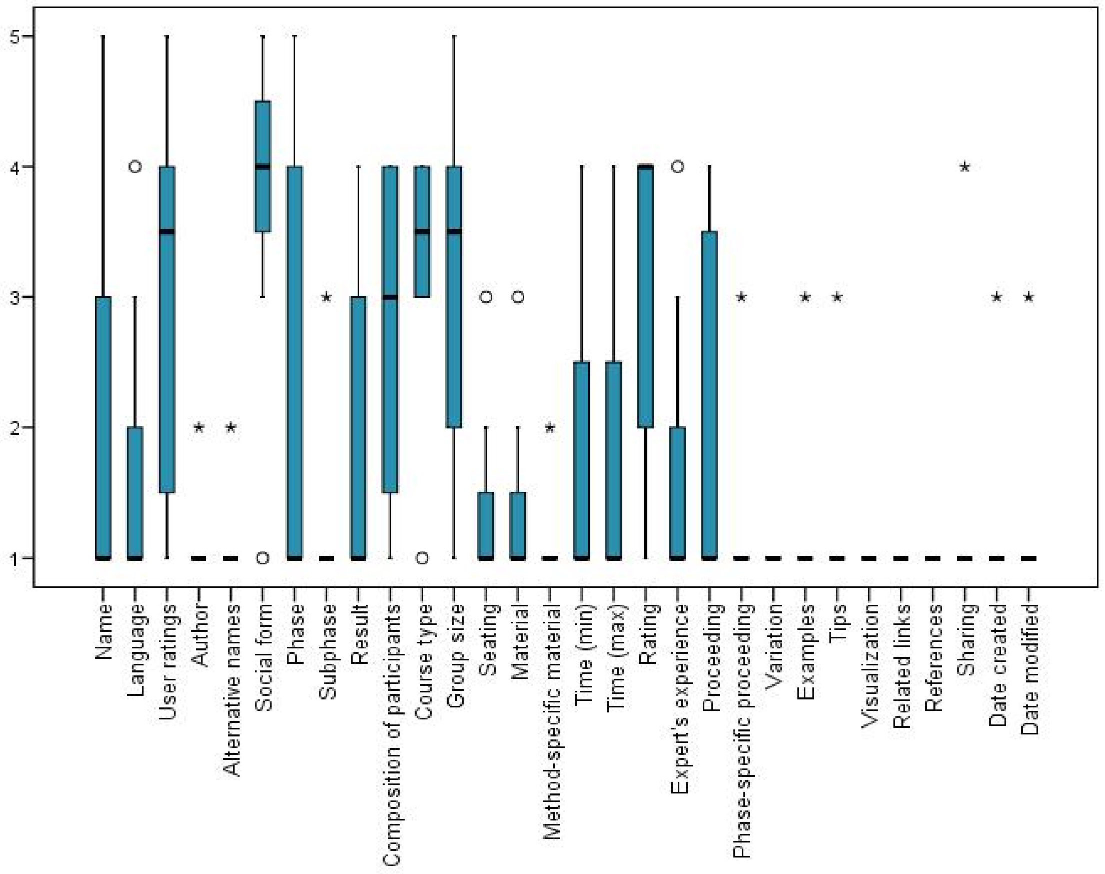


Figure 6.11.: Box plots represent how often the participants have searched various properties of the didactic methods (question 8). 1 = never; 2 = rarely; 3 = occasionally; 4 = fairly often, 5 = very often; Circle: outliers; Stars: extreme values.

(14d) I would recommend MobiDics to other people.

The possible answers were: 1 = strongly disagree, 2 = disagree, 3 = neutral, 4 = agree, 5 = strongly agree. Figure 6.12 represents the answers for question 14. Additionally, Table 6.8 shows the descriptive statistics for this question.

On average, participants agreed to statements (14a), (14b) and (14c) with mean values 4.38 ± 0.744, 4.38 ± 0.518 and 4.38 ± 0.744 , respectively. Only one participant was neutral to the statement (14a) and (14c). There is a slight trend that participants agree strongly to the statement (14d), with a mean value of 4.68 ± 0.518 and a median value of 5.0. Overall, participants agreed that MobiDics is a helpful tool which they would continue to use and recommend to other people.

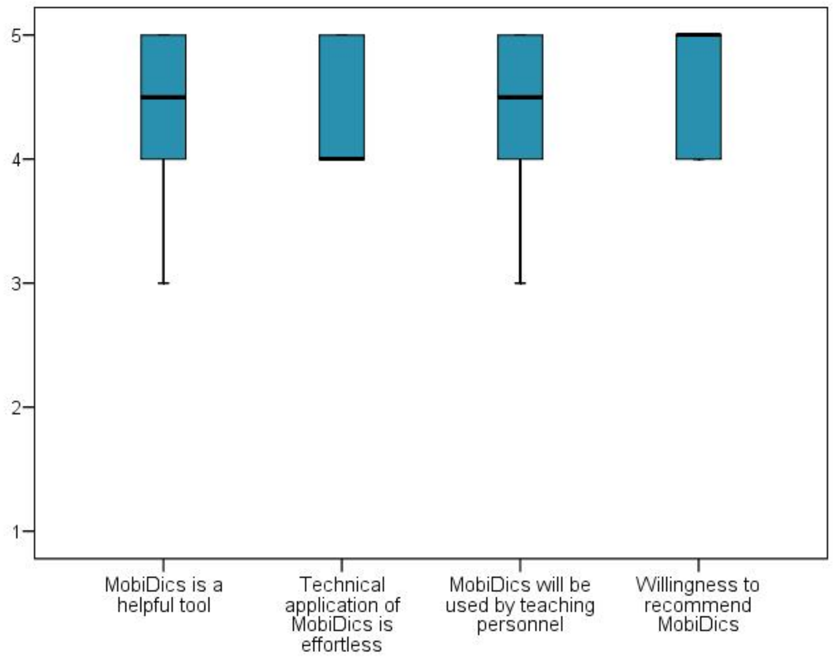


Figure 6.12.: Box plots represent the overall impression of MobiDics (statements (14a) to (14d)). 1 = strongly disagree; 2 = strongly agree; 3 = neutral; 4 = agree; 5 = strongly agree.

Table 6.8.: Descriptive statistics for question 14 of the end questionnaire (overall impression of MobiDics).

Statement	**Minimum**	**Maximum**	**Mean**	**Median**	**Std. deviation**
(14a)	3	5	4.38	4.50	0.744
(14b)	4	5	4.38	4.00	0.518
(14c)	3	5	4.38	4.50	0.744
(14d)	4	5	4.68	5.00	0.518

Comparing Feedback from Both Questionnaires

Questions 6 and 7 were asked in both questionnaires. In order to estimate participants' knowledge about didactic methods, they had to rate the following statements (question 6):

(a) I know a number of didactic methods that I can use during class.

(b) I can modify didactic methods depending on learning/teaching situation.

(c) I can adjust didactic methods during class if the situation changes.

(d) I know a suitable didactic method for every class I give.

Figure 6.13 shows answers for statement (a) given in both questionnaires in box plots. The median value was “I agree” during the start questionnaire. The median changed to 3.5 in the end questionnaire. This value is between “neutral” and “I agree”. Although the interquartile range is shorter in the end questionnaire and varies between “neutral” and “I agree”, the range of answers did not change. Mean values for start and end questionnaires were 3.50 ± 1.069 and 3.50 ± 0.926, respectively, therefore rated between “neutral” and “I agree”. There was no significant difference between start and end questionnaires for the statement (a) ($p = 1.0$), perhaps caused by a low number of participants.

An explanation for the drop in median value in the end questionnaire could be attributed to the steep learning curve during the user study. While users may have considered themselves experienced with didactic methods prior to commencing the study, they soon realized just how many more methods were available and therefore came to the conclusion that their experience level was in fact quite limited.

Figure 6.14 covers the statement (b). The median value increased from 3.5 to 4.0, indicating that in the end questionnaire more participants agreed to statement (b). There was also a change in mean values: from 3.63 ± 0.744 (start questionnaire) to 4.00 ± 0.926 (end questionnaire). One participant disagreed to the statement in the end questionnaire, marking this value as an extreme outlier. However, there was no significant difference between the answers of the two questionnaires ($p = 0.180$).

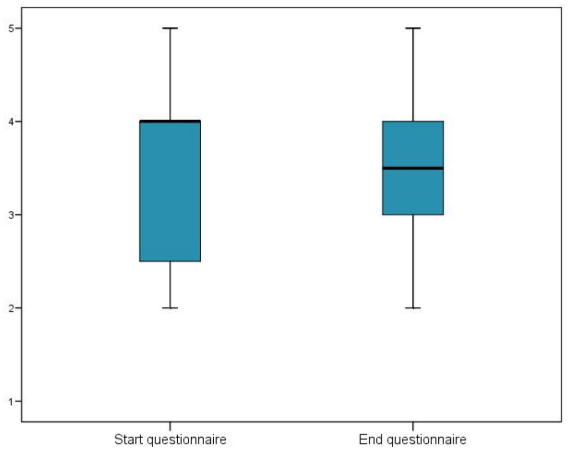


Figure 6.13.: Self-assessment of didactic method knowledge in start and end questionnaires (statement (a)).

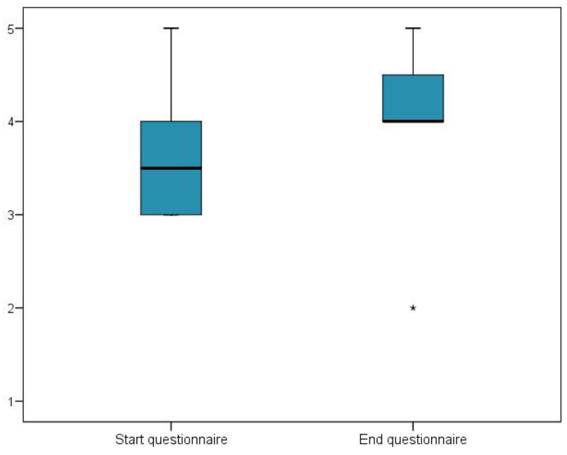


Figure 6.14.: Self-assessment of the ability to modify didactic methods depending on learning/teaching situation in start and end questionnaires (statement (b)). Star: extreme value.

Results of statements (c) and (d) are shown in Figure 6.15 and Figure 6.16, respectively. With an average value of 3.38 ± 0.744, participants were neutral to the statement (c) in the start questionnaire. In the end questionnaire, we can see a slight change in favor of the statement, with an average value of 3.88 ± 0.835. This shows that participants were more likely to agree with the statement (c) after using MobiDics application. The median value of the end questionnaire ("I agree") also slightly increased in comparison to the start questionnaire, where the median varied between "neutral" and "I agree" (median value: 3.5). One participant disagreed to statement (c) before and after using the MobiDics application. One participant was neutral to the statement before the user study, but completely agreed to it after the application usage. These two cases are marked as extreme outliers.

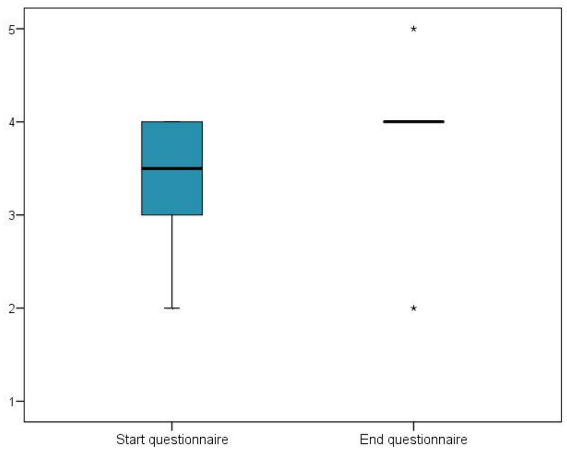


Figure 6.15.: Self-assessment of the ability to adjust the didactic methods to a changing situation during class before and after MobiDics usage (statement (c)). Stars: extreme values.

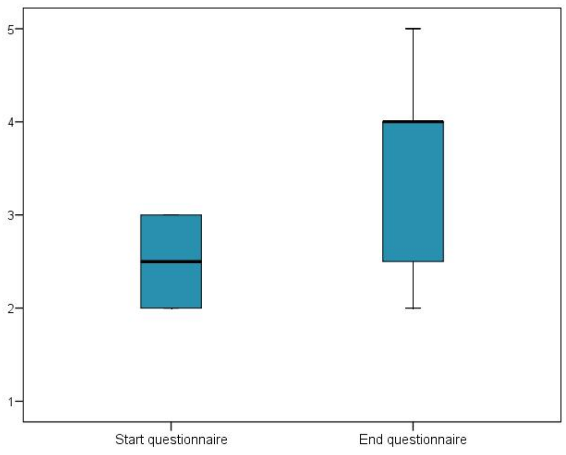


Figure 6.16.: Self-assessment of knowledge of a suitable didactic method for every class (statement (d)). Stars: extreme values.

No significant change between start and end questionnaires was found ($p = 0.102$), although we can see that participants in general tend to agree to statement (c). It can be argued that this trend is due to observed didactic methods and their properties, such as examples, proceeding of every didactic method or tips from experts. Having read this information, participants are more likely to have understood how a method can be adjusted to different situations.

Answers to statement (d) varied in the start questionnaire from "I disagree" to "neutral", with a mean value of 2.50 ± 0.535 and a median value of 2.5. According to the end questionnaire, the median and mean values increased to 3.5 ± 1.069 and 4.0, respectively. Thus, the participants tend to agree to the statement (d) after MobiDics usage. The range of answers in the end questionnaire varied from "I disagree" to "strongly agree". The average value of the statement (d) increased significantly compared to the start questionnaire ($p = 0.038$). Our hypothesis was that on average participants would be more likely to agree to statement (d) after the MobiDics usage. The change could be understood considering the variety of didactic methods that are provided by MobiDics.

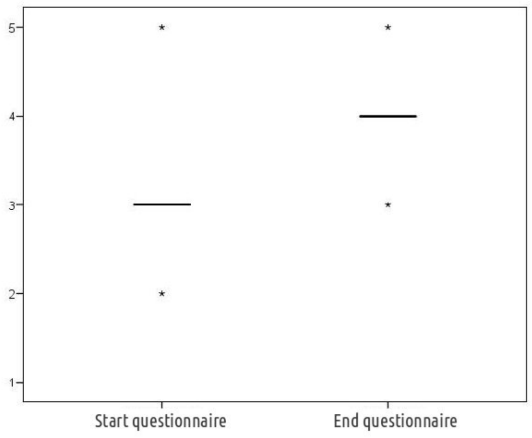


Figure 6.17.: Comparison of participants' self confidence while introducing a new didactic method before and after MobiDics usage (statement (e)). Star: extreme values.

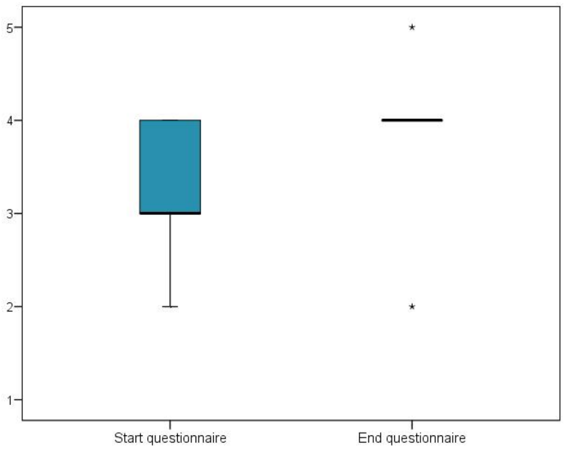


Figure 6.18.: Comparison of participants' self confidence while applying a didactic method before and after MobiDics usage (statement (f)). Star: extreme values.

Statements for question 10:

(e) I feel confident while introducing a new didactic method in a class.

(f) I feel confident when using a didactic method.

(g) After using a didactic method I know how to handle the resulting feedback.

(h) I know how to change the classroom in order to use a didactic method.

The median of participants in the start questionnaire regarding statement (e) was neutral to the statement (see Figure 6.17). In the end questionnaire the median changed to "I agree". One participant disagreed to this statement in the start questionnaire, but stated that she agreed to the statement after MobiDics usage. On average, participants were neutral to the statement (e) in the start questionnaire (mean: 3.13 ± 0.835). The mean value increased to 4.0 ± 0.535, thus, on average, participants agreed to the statement after the MobiDics usage.

In the start questionnaire, the range of answers varied from "disagree" to "strongly agree". The range of answers in the end questionnaire no longer contains the case "disagree", thus, the range varies from "neutral" to "strongly agree". The results show that confidence of participants

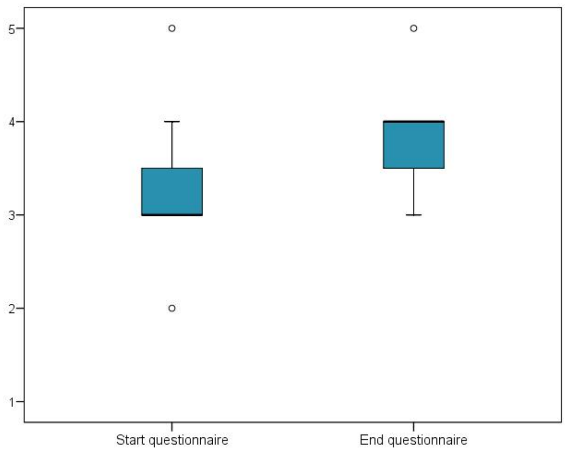


Figure 6.19.: Self-assessment of the ability to handle results of the didactic method before and after MobiDics usage (statement (g)). Circle: outliers.

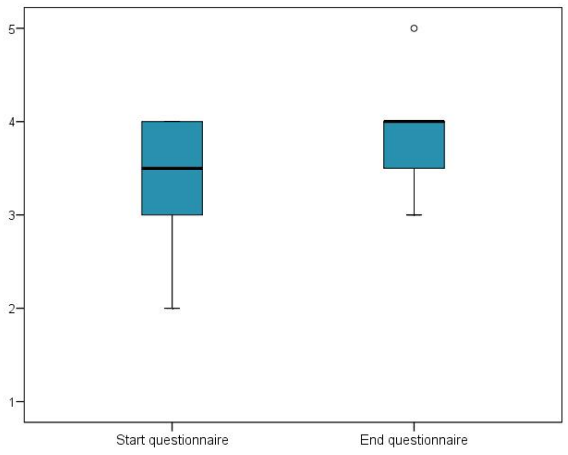


Figure 6.20.: Self-assessment of the ability to adjust the classrooms to execute the didactic method before and after MobiDics usage (statement (h)). Circle: outliers.

in introducing new didactic methods rises significantly after the MobiDics usage ($p = 0.020$). Perhaps, the clear representation of the didactic methods and of the proceedings increased the participants' confidence.

Figure 6.18 shows results regarding the statement (f). On average, participants were neutral to the statement in the start questionnaire (mean: 3.25 ± 0.707). One participant disagreed to this statement. The interquartile range varied from "disagree" to "agree". Both median and mean values increased according to the end questionnaire. With a median value of 4.0, participants state that they feel comfortable when using the didactic method. The mean value is 3.88 ± 0.835. One participant agreed strongly to this statement, although there was no significant difference between the questionnaires ($p = 0.187$).

On average, in the start questionnaire participants were neutral to statement (g) with a mean value of 3.25 ± 0.886. The median value also was "neutral" (see Figure 6.19). One participant disagreed to the statement. Another participant agreed to the statement (strongly). These cases are marked as outliers. In the end questionnaire, the range of answers was smaller. The minimum value is now "neutral", while the median and mean values are "agree" (mean: 3.88 ± 0.64, median: 4.0). One participant agreed strongly to statement (g), marking it therefore as an outlier. With $p = 0.025$, there is a significant difference between start and end questionnaires regarding statement (g).

Statement (h) is covered in Figure 6.20. In the start questionnaire, participants were on average neutral to the statement (mean: 3.38 $\pm 0.0.744$). The median value of 3.5 suggests a slight trend to agreeing to the statement. Overall, the range of answers in the start questionnaire varied from

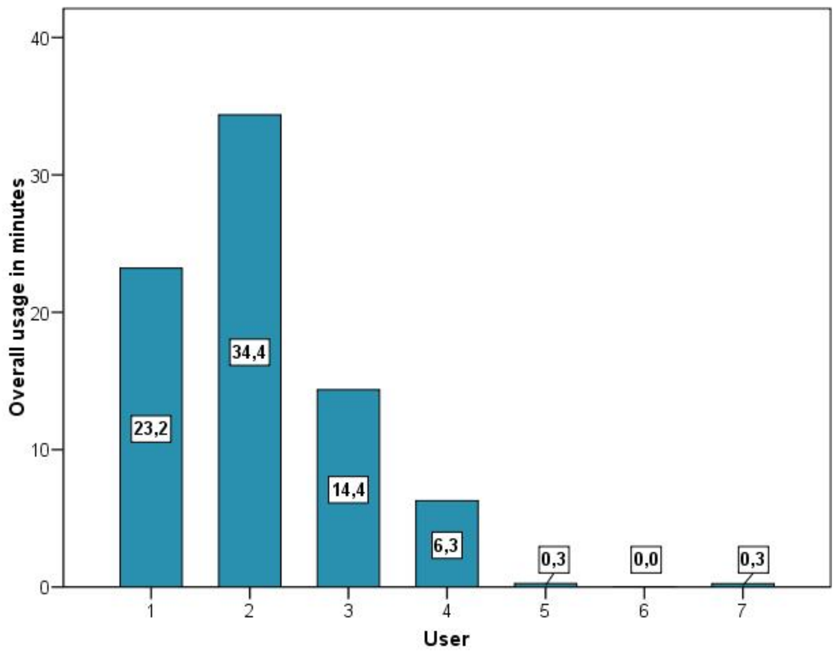


Figure 6.21.: Usage distribution by participant.

"disagree" to "agree". The range of answers in the end questionnaire varied from "neutral" to "strongly agree", therefore concentrating the answers in the upper half of the figure. The median value increased from 3.0 to 4.0 and mean value increased from 3.38 ± 0.744 to 3.88 ± 0.641. Therefore, participants on average tend to agree to statement (h). It was interesting that one participant changed the answer from "agree" to "disagree" after the MobiDics usage. There was no significant difference between the start and end questionnaires ($p = 0.157$).

Additional Feedback

Questions 10- 14 were open-end questions. The participants were asked to express their opinion about the possibilities and shortcomings of systems that provide didactic methods. Their answers were gathered and will be presented below.

In question 10, participants were asked to discuss the knowledge transfer for didactic methods. Participants stated that general information about the didactic method was easy to understand via website/app. New didactic methods could be found this way and they could get inspiration for the next class. The exchange of experience and knowledge with other users was also considered useful.

Question 11 focused on aspects of the knowledge of didactic methods that cannot be transferred via a website/application. Participants stated that it is solely dependent on the individual user with

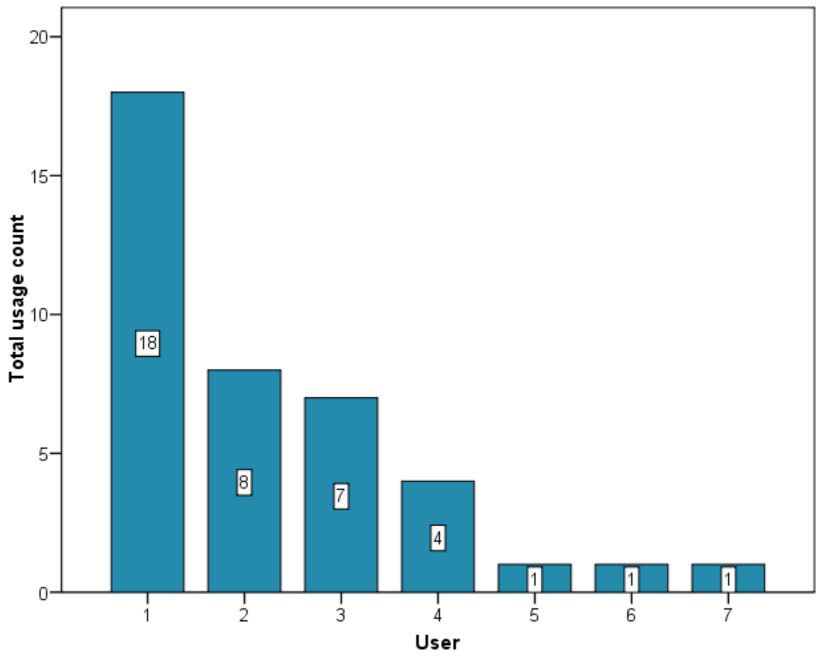


Figure 6.22.: Shows the total number of usages of MobiDics application per participant.

regards to how well they can use a didactic method. Also, it is difficult to learn how to correctly handle unforeseen situations. Participants believe that to combat these types of difficulties, a personal training session or role-playing exercise would be more suitable.

Participants were asked to discuss what could help them to become more confident when using didactic methods (question 12). The participants stated that seminars and practical exercises would be very helpful. One participant stated that it was important to communicate with colleagues from similar fields to discuss subject-specific features/enhancements. One participant hopes for continuation of MobiDics project.

In the final open-end question, participants were asked about the characteristics of a didactic method expert. Participants stated that it is important to have vast knowledge about didactic methods (e.g., which material is needed, for how many students is the didactic method suitable etc.). They additionally stated that it is important to know how to handle results as well as how to moderate and introduce didactic methods. One participant stated that the most important thing was personal appearance (i.e., aspects like language, voice, being approachable, body language).

6.2.3. Investigating the Logged Data

Seven participants have installed the Questionnaire application and have run it at least once. Based on the logged data, three out of seven participants used MobiDics application only once

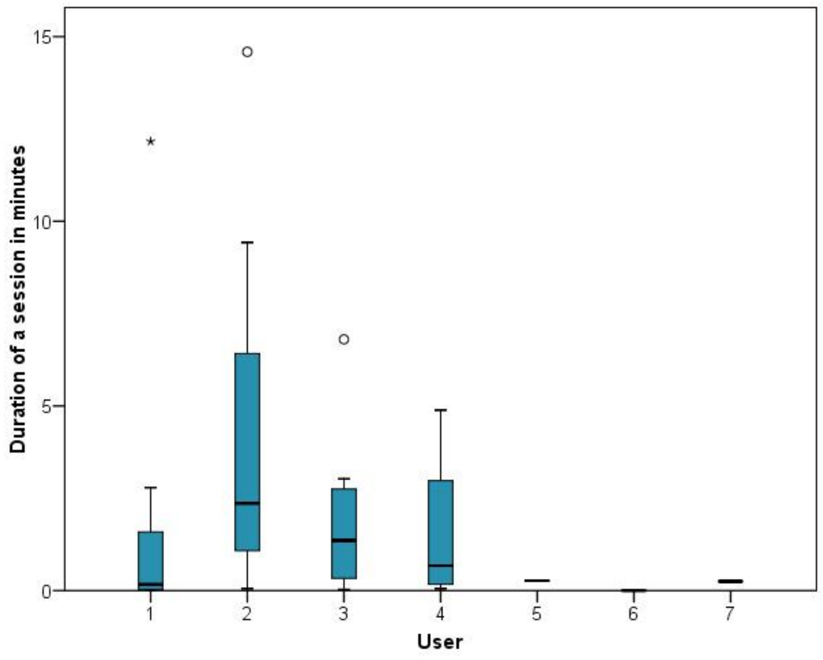


Figure 6.23.: Shows the duration of a single session in minutes for every participant. Box: interquartile range; Whiskers: lower and upper quartiles; The black line: median; Circle: outliers; Star: extreme values.

at the beginning of the user study. It can not be said if they did not use MobiDics afterwards or whether they did not trigger the Questionnaire application during or at the end of the user study. Therefore, the data of their usage could not be sent to the server. Below are the results of the gathered data. Figure 6.21 shows the overall duration in minutes of application usage throughout the user study. The longest overall time a participant has used the application was 34.4 minutes, followed by 23.2 minutes. One participant only installed the application but either did not use it, or she did not use the Questionnaire application to upload the logged data. Two other participants also seem to only have used the application once, the duration of the usage being approximately 30 seconds.

A session is defined as the set of viewed Activities without an interruption that lasted longer than one minute. Interruptions less than one minute could be caused by settings that vary from participant to participant. For example, the amount of time until the "sleeping mode" is activated.

Figure 6.23 shows the duration of sessions per user. The median of the duration of a single session was rather short. This could be explained by the assumption that the participants in most cases just wanted to check some information about a specific didactic method. Nevertheless, participants "1", "2" and "3" have used the application in a session that lasted 12, 14.5 and 7 minutes, respectively.

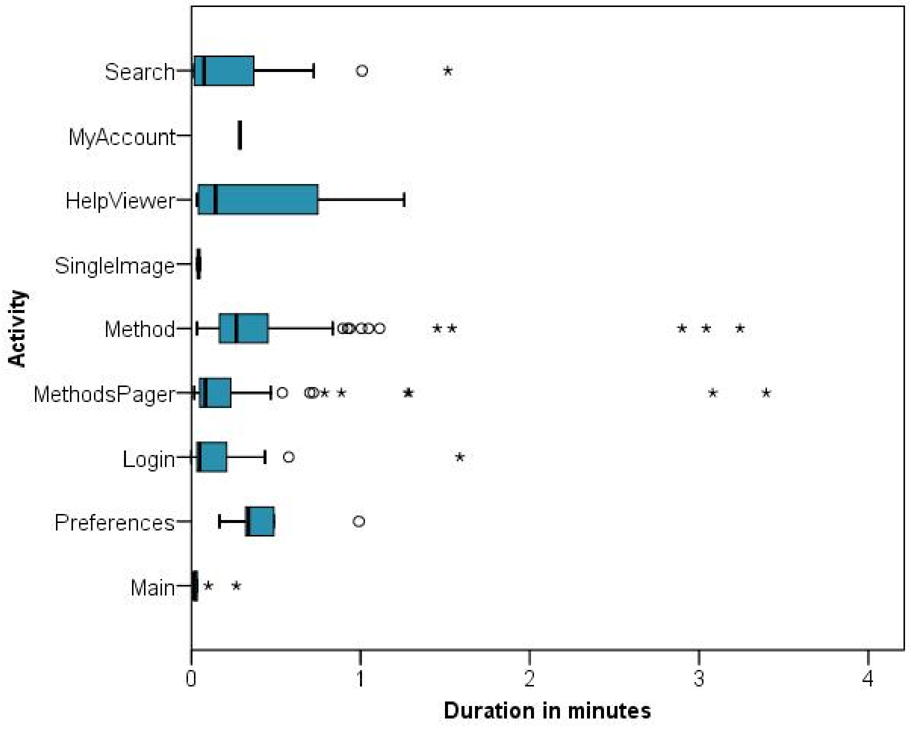


Figure 6.24.: Shows the duration in minutes of observing the Activities before switching to another Activity. Circle: outlier; Star: extreme values.

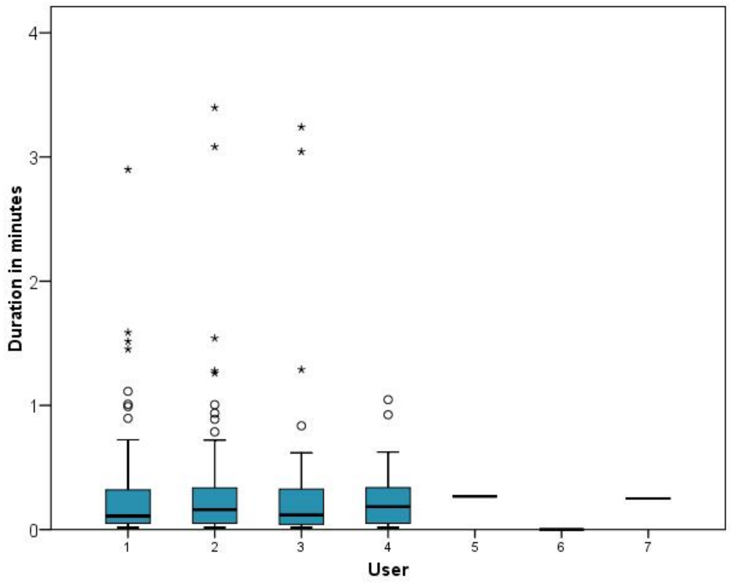


Figure 6.25.: Duration of observing the single Activity per user. Circles: outlier; Stars: extreme values.

Figure 6.22 shows the frequency of the MobiDics application usage throughout the user study. One participant has used the application eighteen times during the four weeks that the study lasted.

Further, we wanted to know which Activities were viewed by the participants and for how long. Participants observed one Activity no longer than 4 minutes. The median value of the duration was rather short for every Activity and varied from 0 to 1 minutes (see Figure 6.24).

Figure 6.25 shows the duration of observing a single Activity per user. The users "1", "2" and "4" have observed an Activity on average for 0.3 (median: 0.11 minutes), 0.3 (median: 0.16 minutes) and 0.4 minutes (median: 0.18 minutes), respectively. User "3" has viewed a single activity on average for 0.3 (median: 0.18).

Chapter 7.

Conclusions

In this chapter we provide the conclusions of this thesis and address possible improvements for future work.

The already existing features of MobiDics, their functionality and usability were tested in a conducted user study (Chapter 6). The results show that participants were more confident in applying didactic methods after the usage of MobiDics. Moreover, the participants have found MobiDics to be a helpful tool and would be willing to use it to improve their knowledge in didactic methods. Considering the finding that participants have observed beneficial outcomes for students when didactic methods are applied during the class, we can deduce that didactic methods applied by well prepared and confident teaching personnel will be beneficial for teaching quality. Since many didactic methods exist, RSs can assist users to find didactic methods that are suitable for them.

Recommendations are made based on various approaches. In the second part of the thesis the benefits and shortcomings of content-based (Section 2.2), collaborative filtering (Section 2.3), and hybrid approaches (Section 2.4) of RSs were discussed. MobiDics web and mobile applications were enhanced in their functionality to recommend didactic methods to a user based on a hybrid approach. The content-based approach recommends didactic methods that are similar to the methods that a user has already liked, and the collaborative filtering approach takes other users' ratings in consideration. The hybrid approach was chosen to combine the benefits of both content-based and collaborative filtering algorithms. We think that these additional features will contribute to the positive user experience with MobiDics.

Future Work

There are some features that could be added to improve the recommendation algorithm. We implemented a hybrid RS that combines content-based and collaborative filtering algorithms. The RS can further be extended by adding knowledge-based, demographic-based and community-based approaches. We will shortly describe benefits of the other approaches and also discuss how the existing ones could be improved further.

Content-based approach: In our calculations we have considered a limited number of didactic methods' properties. Further improvement could be considering all properties of a method. Furthermore, for the nominal data type only the most frequently used value is considered. An addition could be if the algorithm takes all values into account (for example, weighted by their frequency).

Collaborative filtering approach: In its current implementation the algorithm considers all users as being equal. That means, that all ratings are considered for calculating the recommendation scores. An improvement could be to only consider "similar users", i.e., users that have similar characteristics as the user the recommendation is made for.

Furthermore, the current implementation does not take into account whether an element in the deviation matrix (2.1) is based on the ratings of one, a few or many users. A weighted approach to take these differences into account might further improve the results of the Slope-One algorithm.

Finally, implicit ratings, such as search terms typed in by a user can be viewed closely. However, this would require the search queries to be saved.

Demographic-based approach: In our algorithm we do not consider personal information about the user. It would be interesting to take users' similarities in consideration and by doing so, provide more precise recommendations. For example, by considering a user's current teaching position or faculty, didactic methods that were interesting for users of the same background could be recommended. This could be beneficial for new users, who have not rated any of the given didactic methods. One way could be to use the demographic data in the collaborative filtering approach (as discussed above).

Community-based approach: To give a user an opportunity to become friends with other users can offer an additional recommendation approach: to recommend methods which the user's friends found interesting.

List of Figures

List of Tables

List of Acronyms

APK	application package file
app	application
CB	content-based
CBR	Case-Based Reasoning
CF	collaborative filtering
RS	Recommendation system
TUM	Technische Universität München
VMI	Fachgebiet Verteilte Multimodale Informationsverarbeitung

Bibliography

[1] F. Ricci, L. Rokach, and B. Shapira, "Introduction to recommender systems handbook," *Recommender Systems Handbook*, pp. 1–35, 2011.

[2] F. Ricci, "Mobile recommender systems," *Information Technology & Tourism*, vol. 12, no. 3, pp. 205–231(27), 2010.

[3] A. Möller, A. Thielsch, B. Dallmeier, L. Roalter, S. Diewald, A. Hendrich, B. E. Meyer, and M. Kranz, "MobiDics – Improving University Education With A Mobile Didactics Toolbox," in *Ninth International Conference on Pervasive Computing (Pervasive 2011), Video Proceedings*, (San Francisco, CA, USA), June 2011.

[4] A. Möller, B. Beege, S. Diewald, L. Roalter, and M. Kranz, "MobiDics - Cooperative Mobile E-Learning for Teachers," in *Proceedings of the 11th World Conference on Mobile and Contextual Learning (mLearn)* (M. Specht, J. Multisilta, and M. Sharples, eds.), (Helsinki, Finland), pp. 109–116, Oct. 2012.

[5] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Transactions on Information Systems (TOIS)*, vol. 22, no. 1, pp. 5–53, 2004.

[6] R. Van Meteren and M. Van Someren, "Using content-based filtering for recommendation," in *Proceedings of the Machine Learning in the New Information Age: MLnet/ECML2000 Workshop*, 2000.

[7] M. de Gemmis, L. Iaquinta, P. Lops, C. Musto, F. Narducci, and G. Semeraro, "Preference learning in recommender systems," in *Preference Learning (PL-09) ECML/PKDD-09 Workshop*, 2009.

[8] M. J. Pazzani and D. Billsus, "Content-based recommendation systems," *The adaptive web*, pp. 325–341, 2007.

[9] P. Melville and V. Sindhwani, "Recommender systems," *Encyclopedia of machine learning*, pp. 829–838, 2010.

[10] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen, "Collaborative filtering recommender systems," *The adaptive web*, pp. 291–324, 2007.

[11] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," pp. 43–52, Morgan Kaufmann, 1998.

[12] R. Burke, "Hybrid web recommender systems," *The adaptive web*, pp. 377–408, 2007.

[13] P. Melville, R. J. Mooney, and R. Nagarajan, "Content-boosted collaborative filtering for improved recommendations," in *Proceedings of the National Conference on Artificial Intelligence*, pp. 187–192, Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2002.

[14] M. Balabanovic and Y. Shoham, "Fab: content-based, collaborative recommendation," *Communications of the ACM*, vol. 40, no. 3, p. 67, 1997.

[15] F. Ricci and H. Werthner, "Case base querying for travel planning recommendation," *Information Technology and Tourism*, vol. 4, no. 3/4, pp. 215–226, 2001.

[16] F. Ricci, "Travel recommender systems," *IEEE Intelligent Systems*, vol. 17, no. 6, pp. 55–57, 2002.

[17] R. Sinha and K. Swearingen, "Comparing recommendations made by online systems and friends," in *Proceedings of the Delos-NSF workshop on personalization and recommender systems in digital libraries*, vol. 106, Dublin, Ireland, 2001.

[18] D. Gavalas, C. Konstantopoulos, K. Mastakas, and G. Pantziou, "Mobile recommender systems in tourism," *eCOMPASS: eCO-friendly urban Multi-modal route PlAnning Services for mobile uSers*, 2012.

[19] C. Stiller, F. Roß, and C. Ament, "Integration of spatial user-item relations into recommender systems," *International Journal for Infonomics (IJI)*, vol. 3, no. 1, pp. 190–196, 2010.

[20] B. N. Schilit and M. M. Theimer, "Disseminating active map information to mobile hosts," *Network, IEEE*, vol. 8, no. 5, pp. 22–32, 1994.

[21] G. Adomavicius and A. Tuzhilin, "Context-aware recommender systems," *Recommender Systems Handbook: A Complete Guide for Research Scientists and Practitioners*, 2010.

[22] G. Tumas and F. Ricci, "Personalized mobile city transport advisory system," *Information and Communication Technologies in Tourism 2009*, pp. 173–183, 2009.

[23] D. Vico, W. Woerndl, and R. Bader, "A study on proactive delivery of restaurant recommendations for android smartphones," in *ACM RecSys Workshop on Personalization in Mobile Applications, Chicago, USA*, 2011.

[24] S.-K. Ko, S.-M. Choi, H.-S. Eom, J.-W. Cha, H. Cho, L. Kim, and Y.-S. Han, "A smart movie recommendation system," *Human Interface and the Management of Information. Interacting with Information*, pp. 558–566, 2011.

[25] D. Lemire and A. Maclachlan, "Slope one predictors for online rating-based collaborative filtering," *Society for Industrial Mathematics*, vol. 5, pp. 471–480, 2005.

[26] Y. G. Quintana, "A mobile didactics toolbox supporting peer learning," Master's thesis, Lehrstuhl für Medientechnik, Technische Universität München, October 2011.

[27] C. Sammut and G. I. Webb, *Encyclopedia of machine learning*. Springer, 2011.

[28] B. Schmid, "An investigation of activity logging methods in user studies," Master's thesis, Lehrstuhl für Medientechnik, Technische Universität München, Oct. 2012.

[29] C. Brownlow, *SPSS Explained*. Routledge, 2004.