Tag-Based Collaborative Filtering Recommendation in Personal Learning Environments

Mohamed Amine Chatti, Simona Dakova, Hendrik Thüs, and Ulrik Schroeder

Abstract—The personal learning environment (PLE) concept offers a learner-centric view of learning and suggests a shift from knowledge-push to knowledge-pull approach to learning. One concern with a PLE-driven knowledge-pull approach to learning, however, is information overload. Recommender systems can provide an effective mechanism to deal with the information overload problem in PLEs. In this paper, we study different tag-based collaborative filtering recommendation techniques on their applicability and effectiveness in PLE settings. We implement 16 different tag-based collaborative filtering recommendation algorithms, memory based as well as model based, and compare them in terms of accuracy and user satisfaction. The results of the conducted offline and user evaluations reveal that the quality of user experience does not correlate with high-recommendation accuracy.

Index Terms—PLE, recommender systems, collaborative filtering, offline evaluation, user evaluation

1 Introduction

One of the core issues in technology enhanced learning (TEL) is the personalization of the learning experience. There is a shared belief among TEL researchers that TEL models require a move away from one-size-fits-all models toward a learner-centric model that puts the learner at the center and gives her the control over the learning experience.

In recent years, the concept of personal learning environment (PLE) has been widely discussed among TEL researchers, as a natural and learner-centric model that supports the self-directed learning process by surrounding the learner with the environment that matches her needs best. PLE-driven TEL approaches have been proposed as an alternative to traditional learning management system (LMS)-driven TEL initiatives. While an LMS adopts a knowledge-push model and is concerned with exposing learners to content and expecting that learning will happen, then a PLE takes a knowledge-pull model. Learners can create their very own environments where they can pull knowledge that meets their particular needs from a wide range of high-value knowledge sources [1]. One concern with a PLE-driven knowledge-pull approach to TEL is information overload. It, thus, becomes crucial to examine some mechanisms that would help learners to cope with the information overload problem. This is where recommender systems (RSs) can play a crucial role to foster self-directed learning.

Several researchers have stressed the importance of recommender systems in PLEs and TEL environments in general. Verbert et al. [2], for instance, state that recommender systems offer a promising approach to facilitate both learning and teaching tasks, by identifying suitable learning resources from a potentially overwhelming variety of choices. Drachsler et al. [3] note that direct measures like ratings and tags given by users allow identify paths in a learning network, which are faster to complete or more attractive than others. Buder and Schwind [4] explore the potentials of recommender systems for learning from a psychological point of view. The authors note that commercial recommender systems need adaptations to facilitate learning and discuss potential adaptations both with regard to learners as recipients of recommendation and learners as producers of data by contributing annotations, tags or rating. The authors further stress that recommender systems provide (and require) user control, thus facilitating self-directed learning, as it is the case in PLEs.

There is a large number of recommender systems that have been deployed in TEL settings [5]. However, relatively little significant work around the evaluation of recommender systems has been undertaken. Until today, the evaluation of recommender systems gives emphasis to rather technical measures (e.g., accuracy, coverage, performance in terms of execution time) although the importance of including user-related evaluation methods (e.g., effectiveness, efficiency, satisfaction) has been highlighted [5], [6]. Moreover, an implementation of different recommendation algorithms within a single recommender system to compare them against each other is missing in the TEL recommenders literature.

It has been observed that in PLEs, learners rarely share the same or similar learning resources due to the fact that they follow their individual interests and preferences. Thus, recommendation in PLE should rely on the activities and the metadata (e.g., tags) generated by the learners [7]. In this paper, we consider using the tagging information and incorporating it into traditional collaborative filtering (CF)

• *The authors are with Informatik 9 (Learning Technologies), RWTH Aachen University, Ahornstr. 55, D - 52056 Aachen, Germany. E-mail: {chatti, dakova, thues, schroeder}@cs.rwth-aachen.de.*

Manuscript received 19 Oct. 2012; revised 15 Mar. 2013; accepted 17 June 2013; published online 24 June 2013.
For information on obtaining reprints of this article, please send e-mail to: lt@computer.org, and reference IEEECS Log Number TLT-2012-10-0140.
Digital Object Identifier no. 10.1109/TLT.2013.23.

recommendation methods to provide recommendation of learning items in a PLE. We focus on tag-based collaborative filtering recommendation methods and provide a thorough offline and user evaluation of different related techniques that can provide a basis for their further application and research in TEL environments.

2 Recommender Systems

Recommender systems can provide a potential solution to overcome the problem of information overload. Generally, recommender systems aggregate data about user's behavior or preferences to draw conclusions for recommendation of items she most likely might be interested in. Technically, recommender systems are classified into the following classes, based on how recommendations are made [8]:

- *Collaborative filtering*. The user will be recommended items that people with similar tastes and preferences liked in the past. The fundamental idea is that if users x and y rate two items similarly or have similar behaviors they will act on other items similarly. In CF, an item is considered as a black box, and user interactions with the item are used to recommend an item of interest to the user. To suggest items to a user, we either find similar users and recommend items liked by these users (user-based analysis), or we find items similar to an item of interest (item-based analysis).
- *Content-based recommendation (CB)*. The user will be recommended items based on the content of items the user preferred in the past.
- *Hybrid recommendation*. These methods combine collaborative and content-based methods. Hybrid RS overcome the limitations of CF and CB systems and are able to provide more accurate recommendations but at the cost of the complexity to build them.

3 Collaborative Filtering RS

We will focus further only on the collaborative filtering recommendation techniques as they are the main interest for this paper. The existing collaborative RS can be classified in two categories: *memory-based* and *model-based* CF techniques. Memory-based CF algorithms make rating predictions or recommendations based on the user's past ratings. For this purpose, the whole user-item rating database is used to predict the unknown rating of particular item. In contrast to memory-based CF methods, model-based CF algorithms use the collection of ratings to learn a *model*, which is then used to make rating predictions. Furthermore, the existing research distinguishes between *user-based* and *item-based* CF approaches, depending on whether the similarity is computed between users or items. The most popular approaches to compute similarity or weights between users or items are vector cosine-based similarity and correlation-based similarity. A comprehensive survey for CF techniques is presented in [9].

A big advantage of CF recommendation is that there is no information about the item itself needed [10]. CF treats an item as "black box" and the recommendation does not depend on the domain or the language of the application. This is also beneficial for rich media data recommendation. Another major advantage is the fact that CF relies on user interaction. Thus, CF is able to reflect how the preferences change over time. This is a plus in comparison to content-based recommendation approaches, where the content remains the same even over a period of time [10]. Moreover, CF techniques can guarantee the discovery of new items and, thus, increase the user satisfaction.

Despite the advantages of CF Recommendation, there are several shortcomings. These include, for example, the *cold start*, *sparsity*, *scalability*, and *Gray Sheeps* problems [9]. Moreover, as traditional CF recommender systems consider only the rating information, this results in the loss of flexibility [11].

To overcome the limitations of traditional CF recommendation, a significant amount of research has been done recently in trying to generate recommendations by harnessing the interactions in social media as they can reveal further user preferences. Formally, these approaches introduced a new type of RS, namely social recommender systems (SRS).

4 Social Recommender Systems

Social recommender systems address the issues of traditional recommendation by taking advantage of the social data about a user, i.e., his tagging behavior, relationships, membership in communities, likes, comments, votes, bookmarks, and so on. This data implicitly represents preference about certain items or additional contextual data for rich media. Augmenting the traditional collaborative or content-based recommendation with this information can lead to significant improvement.

4.1 Characteristics of SRS

The crucial step in SRS is to decide which social data to exploit and how to use it to achieve better quality of recommendation. Regardless of which social information is used, SRS share few common characteristics:

- *Third dimension*. Exploiting the social data requires adding a third dimension to the two-dimensional user-item matrix. For instance, for the systems using the social tagging information we need to add the connection of tags to users and items. This increases the complexity of the prediction/recommendation computation.
- *Temporal factors*. Unfortunately, the lifespan of social data is very limited as the users' interests tend to change quickly over time. Time is considered as an important factor when building social recommender systems.
- *New entities of recommendation*. While traditional systems are able to suggest only items, using a third dimension in recommendation makes it possible for SRS to recommend also other types of entities—for example, users, tags, or communities.

4.2 Tag-Based SRS

Tag-based SRS leverage the tagging behavior of the user to generate useful recommendations. Recognizing the importance of tags, a large number of SRS harness the tags of collaborative tagging systems. Alag [10] recapitulates why user-generated tags are useful: they use terms that are familiar to the user; they bring out the concepts related to the item; they can capture a semantic value associated with an item even though not mentioned directly in the content; and they can offer useful collaborative information about the user and the item. In TEL context, several researchers stressed the importance of tags in TEL environments. Vuorikari et al. [12], for instance, investigate the importance of tags in "tag ecologies" and show that tags can enrich and add value to multilingual controlled vocabularies. The authors further conclude that tags can become a useful source of metadata for learning repository owners, and help them better understand users' needs and demands. In another study in a context of European learning resources exchange, Vuorikari and Ochoa [13] state that tags could be used to facilitate the discovery of educational resources across country and language borders.

Hsu and Chen [14] define three categories of user-generated tags:

- *Topic description*. This group of tags represents implicitly the topic of the target item. For this reason, the tags are highly correlated to the content of the bookmarked item.
- *Function related*. This type of tags can conceptually describe the function of a resource—for example, *blog*. Such annotations can be related to other tags but in terms of content relevance they play no role.
- *Personal use*. This category of tags reflects the user's attitude to a particular item. Because of the dependency on the user's reflection, they are less relevant to the content.

Similarly, Sen et al. [15] offer a classification of user-generated tags in three similar groups: *factual*, *subjective* or *personal*. They have concluded that the first group refers to "high-quality tags" that capture important concepts about a given item, while the other two are less preferred by the users.

As keyword annotations seem to be a comprehensive way to relate a user to items, incorporating them in recommender systems is a promising step for boosting the performance and quality of recommendation. From a recommendation perspective, the above presented tag categorization indicates that mining folksonomies [13] to discover user interests can be useful. Making use of factual tags can also help in organizing the items in topic clusters and suggesting relevant content based on the user preferences. On the other hand, using the second and third type of tags can increase the diversity of the suggestions. Moreover, tags can enrich the RS with additional information and, hence, improve the quality of recommendations with respect to relevance, coverage and diversity, as well as the user experience. Additionally, using tags to represent the user's preferences, we can even introduce recommendation for systems where rating information is not available. Furthermore, tag-based profiles can result in better user similarity calculation and in identifying more neighbors, thus achieving a more precise and complete recommendation [16], [17], [18]. Another benefit of tags is the fact that they can capture the changing user preferences over time and this can easily be updated in his profile by simply adding the new tags. Hence, the personalization of recommendation will adapt according to the changing preferences.

5 Tag-Based CF Recommendation

Although recommender systems are increasingly applied in TEL, there is relatively little research on recommender systems in TEL that rely on tagging information. In TEL systems, tags are often used to annotate learning resources [2], [5]. In a recent survey of TEL recommender systems, Manouselis et al. [5] discuss few examples of systems that apply tag and rating data for recommendation to overcome the cold-start problem of the recommender system. These approaches, however, mainly use classic CF recommendations based on the rating information. In another study, Verbert et al. [2] explore the extent to which implicit feedback of learners (e.g., tags) can be used to augment explicit feedback ratings to improve recommender performance in the TEL domain. As a potential solution for the data sparsity problem, the authors apply a standard CF recommendation on several data sets (e.g., MACE and Mendeley data sets) that include tags that are provided by users on learning resources [2]. In this experiment, the tagging information is just used to rank items to the user in order of decreasing relevance. In sum, state-of-the-art tag-based recommenders in TEL combine rating and tagging information. To our knowledge, there are no TEL recommenders that rely solely on tagging information.

In other application areas (e.g., e-commerce, artificial intelligence), the evaluation of several algorithms developed to use the tagging information for recommendation has shown that tag-based recommendation can improve the performance and quality of the recommendation in comparison to the traditional recommender systems [19], [20], [21], [11]. In the next sections, we discuss important state-of-the-art tag-based CF recommendation approaches that could be applied in a TEL context.

Unlike the classic collaborative recommendation, incorporating tags extends the <*user, item*> relation into a three-dimensional relation <*user, item, tag*>. Moreover, as users can give more than one tag to each item, the user-item matrix becomes a three-dimensional tensor [22]. Tso-Sutter et al. [20] suggest to solve the three-dimensional problem by projecting the <*user, item, tag*> relation into three two-dimensional matrices: <*user, tag*>, <*item, tag*>, and <*user, item*>. For this purpose, the standard user-item rating matrix was augmented once horizontally with the user-tag tag matrix as pseudo items and once vertically by the item-tag matrix as pseudo users. On each extended matrix, a user-based and, respectively, item-based collaborative filtering is performed. By fusing the results from both predictions a list of top-N items is generated.

Similarly, Wang et al. [22] decompose the tensor into user-tag and item-tag matrices and use them to find latent topics among users/items and to cluster these according to their neighborhood. The outcome is a predicted score based on the ratings of similar users, as well as the similarity to other items.

Zhou et al. [11] also split the tensor to solve the three-dimensional complexity. Their approach includes probabilistic factor analysis by utilizing both the rating and the tagging information. They propose a unified matrix factorization model on the three two-dimensional matrices: the user-item rating matrix, the user-tag, and the item-tag tagging matrices.

All examples above [20], [22], [11] utilize rating information. They all share the same technique to deal with the complex three-dimensional relation <user, item, tag>: splitting the folksonomy tensor, combining the low-dimensional matrices in a unified model or separately using them for neighborhood formation, and generation of top-N items lists.

However, most social tagging systems do not have explicit rating data and they cannot build the traditional user-item rating matrix. To overcome the lack of rating information, most researchers (e.g., [21], [23], [19], [24]) build an *implicit binary user-item matrix*, where each entry indicates if a user has bookmarked or annotated an item. In the next sections, we discuss the related work in the area of purely tag-based CF recommendation according to the main categories in CF, namely memory based and model based.

5.1 Memory-Based Techniques

Liang et al. [21] propose a memory-based tag-based CF approach that incorporates not only similar tagging behavior and number of items tagged in common but also if two users annotated a common item in a similar way. The authors present a user-based and an item-based alternative of their algorithm. In terms of precision and recall, the presented approach outperformed both the user-based and the item-based traditional CF, as well as the mentioned earlier tag-aware method of Tso-Sutter et al. [20]. Additionally, the item-based approach has considerable better precision and recall in comparison to its user-based counterpart.

Firan et al. [23] investigate CF recommendation for music by building different tag-based user profiles. The first type of profile relied on indirect signals, where the list of tags is extracted from the tracks that the user listens to. The second type of algorithm considered directly the tags a user has used. The proposed recommendation algorithm was evaluated in a study where the main objectives were to inspect: 1) the accuracy quality, meaning how well a recommended item fits the user's music preferences, and 2) the degree of novelty of the item. The authors used a normalized discounted cumulated gain as an evaluation metric. Compared to the traditional user-based CF as baseline, all tag-based CF algorithms using the different tag profiles performed worse. The authors stated that this is due to a strong tag ambiguity and identified it as a direction for improvement.

5.2 Model-Based Techniques

The algorithms in this section build different models and use them to provide recommendation.

The first approach uses a classification data mining method. Ji et al. [19] consider three two-dimensional input matrices; i.e., a user-tag, user-item, and a tag-item matrix. The algorithm consists of a candidate tag set (CTS) generation step followed by a top-N recommendation generation step based on a naive Bayes classifier. The algorithm was evaluated on a data set from Del.icio.us by splitting it in training and test sets, and measuring the average recall. It was observed that the performance improves with the increasing size of neighbors k, as well as with the increasing size of the CTS w. The best result, 8.839 percent, was achieved for $k = 50$ and $w = 70$. The authors also compared their algorithm to traditional user-based and item-based CF and concluded that for a small number of recommended items (<35) the tag-based approach outperforms the traditional ones. The authors noted that further improvements were needed to effectively deal with "noisy" tags.

The second model-based approach applies topic clustering of items as a solution to the synonymy problem of tags (i.e., two different tags have a similar meaning). The idea bases on the assumption that co-occurring tags will be interpreted as belonging to the same topic. Therefore, Nakamoto et al. [24] perform topic clustering of items by using expectation maximization (EM) clustering method. Then, each item is represented by a *topic domain vector*, where each entry gives the probability that the item k belongs to the topic j. The probability is calculated based on all tags annotating the item k. The algorithm proved that it can recommend items even for new users with few bookmarks. The proposed approach was evaluated by comparing it to three other algorithms: 1) based only the topic domain, 2) using the most popular tags of the currently viewed item, and 3) matching the topic domain vector of the viewed item to the bookmarks of all users with commonly shared items. The proposed algorithm performed best by achieving a precision of 0.594.

5.3 Comparison of Existing Approaches

In the preceding sections, we have presented the details of seven representative tag-based CF recommendation algorithms. Three of them [20], [22], [11] incorporate also rating information and the other four [21], [23], [19], [24] rely solely on tagging information. All algorithms decompose the original three-dimensional folksonomy into two-dimensional matrices. Typically for the CF recommendation, the most often used data mining technique is nearest neighborhood formation. As a similarity metric, the majority of approaches utilize the cosine distance.

From an evaluation perspective, all proposed algorithms were evaluated on popular data sets. Regarding the baseline methods used for comparison, it can be observed that most evaluations aimed to prove that using or incorporating tag information into traditional CF will improve the quality of recommendation and, thus, the focus has been on whether the proposed approach will outperform traditional CF recommendation approaches or not. As far as the evaluation methodology is concerned, there are predominant offline evaluations where the classification accuracy of the recommended top-N items is examined. None of the discussed research has used more than one evaluation methodology.

5.4 Envisioned Progress beyond the State of the Art

In this paper, we aim at providing recommendation of learning items (i.e., resources, services, and peers) in a PLE.

We focus on tag-based CF recommendation that rely solely on the tagging information. Our approach differs from the state-of-the-art approaches in both the baseline methods used for comparison and the evaluation methodology. We will investigate, compare, and contrast a wide range of tag-based CF recommendation methods, memory as well as model based. And, we will apply two different evaluation methodologies, namely an offline evaluation by measuring the accuracy quality of the recommendation and a user evaluation to check whether the user satisfaction correlates with the measured high accuracy of the recommendation. Based on both evaluation methods, we aim at highlighting the best tag-based CF algorithms for personalized recommendation supporting learners in discovering new quality learning items in their PLEs.

6 Context of the Study

This study was conducted based on a data set generated within the PLEM[1] environment. PLEM is a personal learning environment that supports learners in creating a personalized space, where they can easily aggregate, manage, tag, and share learning items. A learning item in PLEM can be a link to a learning service, learning expert, a learning community, or a learning resource collection [25]. PLEM is an online service that has been used by, for example, students at RWTH Aachen University and Technical University Sofia as well as secondary school pupils at the European School Mol in Belgium [26]. The PLEM data set currently include 83 users, 773 items, and 1,059 tags. We enhanced PLEM with a recommendation engine to recommend learning items based on the tagging behavior of the learner.

The aim of the study is to experiment with different tag-based CF recommendation algorithms, memory based as well as model based, in a PLE context. In the case of memory-based tag-based CF we use the k-nearest neighbor method (kNN). For the model-developing algorithm, we focus on the following approaches: dimensionality reduction, probabilistic classification, clustering, and association rule mining. Further, we consider for most algorithms to perform user-based and item-based CF. For the user and item similarity computation, we utilize the cosine similarity metric.

6.1 Tools and Technologies

This section summarizes the tools and technologies used for implementing our approach. The PLEM system has been developed using Java programming language, Spring in the back end, and Google Web Toolkit for the front-end interface. The tag normalization and stemming task takes advantage of the Apache Lucene's *SnowballAnalyzer* [27]. For executing the data mining tasks, we leveraged the *Weka API*.[2] We use *Weka v*3.7, development version, as well as two separate packages: *LatentSemanticAnalysis* 1.0.1 and *OpticsandDBScan* 1.0.1. Since Weka does not support the cosine similarity measure, we extended the *EuclideanDistance* class of Weka and implemented a subclass *CosineDistance* to compute a cosine distance metric: $1 - cos(u, v)$, where $cos(u, v)$ is the cosine similarity between two users/items u and v. Building a recommendation module with Weka requires the following four basic steps [10]:

1. *Attributes generation* creates the list of attributes that characterize the data set. In PLEM, these are the user tags used for the annotation of a given item.
2. *Training set generation* builds the data set used for learning the data patterns.
3. *Learning the predictive model* specifies the classifier or clusterer to be used for building the respective model.
4. *Recommendation for the active user* by using the learned predictive model performs the recommendation generation.

Our implementation of all recommendation algorithms follows the above-described steps.

6.2 Data Generation and Preprocessing

The data generation and preprocessing task is performed in three main steps. First, we generate the list of attributes based on the user tags in the system. The second step is generating the user profiles, or, respectively, the item profiles. The third step builds upon the data from the previous two steps to create the training data set for Weka.

6.2.1 Tag Normalization and Stemming

As mentioned, the first step is to create a list of attributes. For this purpose, we query the database for all tags that are used in PLEM to annotate items. Once all tags are normalized and stemmed using Lucene's *SnowballAnalyzer*, we build the list of attributes. In all algorithms, the first attribute is an ID used to identify a user or item instance. The remaining attributes represent tags from the stemmed tag list. Most algorithms require numeric values. The Apriori algorithm [28], however, needs nominal values {0, 1}, where 1 indicates the presence and 0 the absence of a tag in a transaction.

6.2.2 Tag Profiles

The second step is to build user or item tag profiles. To build the profiles, we investigate the frequency of tag applications of a particular tag.

User profiles are created as follows: for each user u, a tag frequency vector is created. This vector indicates how often u applied the tag t. For this purpose, all tags of user u are queried from PLEM's database, stemmed, and added to a list. For each stemmed tag t_{stem}, we compute a frequency score as the number of its occurrences in the stemmed list. Finally, the pair $<tag, frequency>$ is added to user's u tag vector.

The *item profiles* contain information about the tags that are used to annotate an item i. For each item i, we create a tag frequency vector, similar to the approach used in user profiles. All tags applied on item i are extracted from the database, stemmed, and added to a list. For each stemmed tag t_{stem}, we compute how frequently it occurs in the stemmed list and add the pair $<tag, frequency>$ to the item's i tag vector.

1. subprogra.informatik.rwth-aachen.de:8180/PLEM/.
2. http://www.cs.waikato.ac.nz/ml/weka/.

6.2.3 Input Data Set Generation

The third step is to generate the input data set for Weka. In total, our chosen algorithms use five different data sets. We list shortly the generation procedure for all of them below:

- *User-tag frequency data set, A*. Create an empty data set. Take the list of numeric tag attributes and the collection of all user profiles. For each user profile, create an Instance object. Set the ID attribute to the ID of the user. For each tag attribute: if the selected user has applied the tag, set the attribute value to the frequency from his tag vector; otherwise 0. Finally, add the Instance to the data set.
- *Item-tag binary data set, Q*. First, an empty data set is created. The data set Q takes a list of numeric attributes and the collection of all item profiles. For each item profile, create an Instance object. Set its ID attribute to the item's ID. For each tag attribute: if the selected item has been annotated by this tag, set the attribute value to 1; otherwise 0. Finally, add the Instance to the data set.
- *Item-tag frequency data set, M*. Create an empty data set. For data set M, get a list of numeric tag attributes and the collection of all item profiles. For each item profile, create an Instance object. Set the ID attribute to be the ID of the item. For each tag attribute: if the selected item has been annotated with this tag, get its frequency from the tag vector and set the attribute value to it; otherwise 0. Finally, add the Instance to the Instances collection.
- *User-tag binary data set (nominal attributes), B*. Create an empty data set. For the data set B, take a list of nominal tag attributes and the collection of all user profiles. For each user profile, create an Instance object. Set the ID attribute to the ID of the user. For each tag attribute: if the selected user has applied the tag, set the attribute value to 1; otherwise 0. Finally, add the Instance to the Instances collection.
- *Item-tag binary data set (nominal attributes), L*. Create an empty data set. Take a list of nominal attributes and the collection of all item profiles. For each item profile, create an Instance object. Set the ID attribute to be the ID of the item. For each tag attribute: if the selected item has been annotated by this tag, set the attribute value to 1; otherwise 0. Finally, add the Instance to the Instances collection.

All frequency data sets are normalized before feeding them to the respective data mining algorithm of Weka. With the description of all required input data sets, we conclude the section of data generation and preprocessing. Next, we present all implemented algorithms that use the generated training data sets.

6.3 Recommendation Generation

In this study, we implemented 16 different tag-based CF recommendation algorithms: two memory based and 14 model based using different classification, clustering, association, or dimensionality reduction models. All data mining methods that require similarity computation in term of distance between two instances of the data set use the cosine distance function. Fig. 1 gives an overview of all implemented recommendation algorithms. In total, there are five categories of algorithms: k-nearest neighbor, latent semantic analysis for dimensionality reduction, naive Bayes classification, clustering, and Apriori association rule mining. All approaches, except the classification ones, are implemented both in their user-based and item-based alternatives. In the following sections, we outline each proposed algorithm. For reasons of space, we do not give the implementation details of the algorithms in terms of pseudocode.

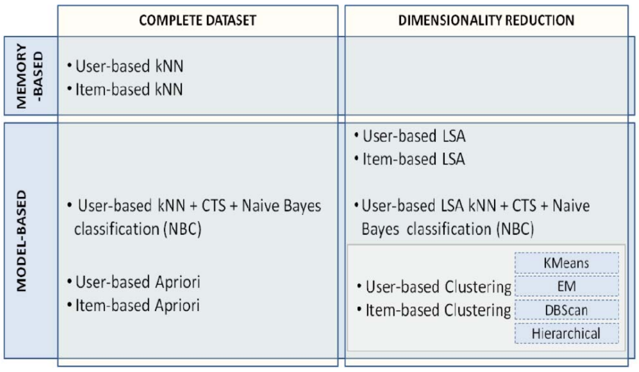


Fig. 1. Overview of the implemented algorithms.

6.3.1 Algorithm 1: User-Based kNN CF

The main assumption behind the k-nearest neighbor approach is that users with similar tagging behavior will share the same learning interests. The more their tags overlap, the more interests these users will have in common. Consequently, we can recommend the items of the similar users to the active user, given that he has not saved those items yet. kNN is an instance-based learning approach. This approach is also called lazy learning, because its training phase is simply storing all instances from the data set. Given a particular point of the data set, the task of the algorithm is to find the k closest points, the nearest neighbors, from the training data [28]. The distance between two instances is determined by the cosine distance. For the kNN discovery, performed by Weka's *Linear-NNSearch*, we input the user-tag frequency data set A and the number of k similar users we want to have. In case that the search reveals similar users, we get their items, remove those known by the target user u. Each candidate item receives one vote from each neighbor who has tagged it. Finally, the top-N item list for target user u is generated from the candidate item list by sorting the received votes.

6.3.2 Algorithm 2: Item-Based kNN CF

The item-based nearest neighborhood CF method aims to overcome the difficulty of finding similar users. The user's tagging behavior reflects the diverse and highly individual interests. Items, on the other hand, tend to be annotated according to the concepts they represent. It is more probable that one can find items sharing the same topic. For the neighborhood computation, the item-tag frequency M data set is given as input to Weka. For each target user u, it computes the k neighbor items that are most similar to each of the user's u items. Each new candidate item receives one vote. If an item is already in the candidate set, we increment its voting with 1. Given all candidate items, the top-N recommendation list is built.

6.3.3 *Algorithm 3: User-Based LSA kNN CF*

To overcome the problems of synonymy and polysemy of tags as well as the high dimensionality and sparsity of the data set in the memory-based CF approaches, dimensionality reduction techniques can be applied. We used latent semantic analysis (LSA), a text mining technique that transforms the set of terms (e.g., words, tags) in a linear combination of these terms that represents a *concept*. The main assumption is that co-occurring terms belong to the same latent topic and they are similar in their meaning. In LSA, a low-rank approximation is performed on the original data. It is based on the mathematical technique singular value decomposition (SVD). SVD is a matrix factorization approach. The resulting matrix is a lower dimensional space where the attributes represent the concepts learned, while the original similarity between the instances is preserved [28].

The initial step of our user-based LSA kNN CF algorithm is to find the latent topics in the data set, based on the co-occurrence of tags in the original user-tag frequency data set A. For this purpose, we utilize Weka's *LatentSemanticAnalysis*. The low-rank approximation of the full data requires a rank r parameter that specifies the proportion of total singular values to be used. Once we have established the LSA attributes of the lower dimensional semantic space A', the nearest neighborhood search is performed on this data set. The further steps of candidate items discovery and the generation of the top-N items follow the same logic as in Algorithm 1.

6.3.4 *Algorithm 4: Item-Based LSA kNN CF*

Analogous to the previous approach, we define its item-based nearest neighbor CF counterpart based on the item's dimensionality reduced space. We transform the original item-tag frequency data set M into the lower dimensional M' by the latent topics emerging from the annotations of learning items. Once LSA is performed, the k most similar items to the target user u items are selected. Given the ordered set of all candidate items, the top-N recommendation list is built, as described in Algorithm 2.

6.3.5 *Algorithm 5: User-Based kNN CF with CTS Generation and Naive Bayes Classification (NBC)*

This approach is based on the idea that a user can have a latent preference for the tags of his similar users [19]. The algorithm is divided in two steps. In the first step, we extract the tags from the k-nearest neighbors of the target user u. Then, we compute the predicted value of user u preference for a tag t as the sum of the tag frequency of the k-nearest neighbors of user u for tag t weighted by the cosine similarity between the target user u and his k-nearest neighbors. The w most highly ordered tags are then put in a candidate tag set. In the second step, by applying probabilistic classification, we can recommend to the target user u those items that have the highest probability to be annotated by the tags from CTS. For the classification task, we train Weka's naive Bayes classifier [29] using the item-tag binary data set Q, assuming that each item tag vector represents one class of items. In other words, the occurrence of given tags can be characteristic for one class of items.

6.3.6 *Algorithm 6: User-Based LSA kNN CF with CTS Generation and Naive Bayes Classificiation*

This algorithm is an adapted version of Algorithm 5, using LSA for dimensionality reduction. Due to the power law distribution of tags, it is possible that we cannot identify similar users when calculating the cosine similarity on the entire user-tag data set. Therefore, in the first step of this algorithm, we perform the k-nearest neighborhood discovery on the lower dimensional space A', the result of the application of LSA on the original user-tag frequency data set A. If the like-minded users are determined on the basis of collaboratively shared latent topics, it can be possible that the CTS contains more appropriate tags for the target user u.

6.3.7 *Algorithm 7: User-Based k-Means Clustering*

Using Weka's *SimpleKMeans* clusterer [29], we build groups of users based on the user-tag frequency matrix A. The parameters for the clustering process are: the number of clusters k, the maximum number of iterations, and the seed. The seed is used for random number generation, and this number is used for the initial assignment of instances to clusters. We can calculate the recommendation for each user, using the obtained assignments of users to clusters. As long as a user u is not the only member of a cluster m, we get the items of his neighbors in m and recommend those he has not seen yet.

6.3.8 *Algorithm 8: User-Based EM Clustering*

Unlike the k-Means clustering approach that assigns a user strictly to one cluster, in the expectation maximization approach [29], a user can belong to several clusters with a certain probability. We consider a probability threshold of 80 percent when assigning a user u to a cluster m. Therefore, for the target user u, we obtain first if he was clustered to any cluster with probability above 80 percent. If this is not the case, we cannot provide any recommendation. Otherwise, we get his neighbor users from the cluster he was assigned to and generate the items unfamiliar to user u as his candidate item set. In the last step, we recommend the top-N items for the target user u. Similar to k-means, the input parameters for the cluster learning phase are the user-tag frequency matrix A, the number of cluster k, the maximum number of iterations, and the seed value.

6.3.9 *Algorithm 9: User-Based Density-Based Clustering*

With the help of Weka's *DBSCAN* [28], [30] we build clusters of users that are highly dense in the user-tag space A. For this purpose we specify ε as the neighborhood radius of an instance and *minPoints* as the minimum number of points to be found in this radius to assign the instance to the present cluster. As usual, the metric for measuring the distance between two points is the cosine distance function. Given the constructed clusters, for a target user u we check if he is assigned to a cluster. If this is the case and there are other users in the cluster, we get the items of these other members as candidates and remove the ones already seen by user u. The top-N items are then generated and recommended.

6.3.10 Algorithm 10: User-Based Hierarchical Clustering

The user-based hierarchical clustering algorithm creates a hierarchical decomposition (a tree of clusters, dendrogram) [29] of the given set of users. The closeness of two users is defined by the cosine distance. Once the hierarchy is learned, we specify the number of clusters k we want to have. Then, we find all the users within the cluster the target user u has been assigned to. Finally, the top-N items are computed and recommended to the user u.

6.3.11 Algorithm 11: Item-Based k-Means Clustering

Based on the item-tag frequency data set M, we build k item clusters, using Weka's *SimpleKMeans* clusterer. For each item i of the target user u, we obtain the neighbor items from the cluster m the item i was assigned to as recommendation candidates. From the list of candidate items, we generate the top-N items to be suggested to the user u.

6.3.12 Algorithm 12: Item-Based EM Clustering

The EM clustering on the item-tag frequency data set M assigns one item i to a cluster m with a certain probability. We iterate over all items of the target user u. If an item i is clustered successfully with a probability more than 70 percent to a cluster m containing other items, we add to the candidate item set of the user u those items that are new for him. The top-N items are then recommended to the user u.

6.3.13 Algorithm 13: Item-Based Density-Based Clustering

For a specified ε as the neighborhood radius of an instance and the minimum number of points to be found in this radius, Weka's DBSCAN builds clusters of items that are highly dense in the item-tag space M. For each item i of the target user u, we do the following: we estimate the cluster assignment for item i if any. Unless there are no other items in this dense cluster m, we get the neighbor items of i inside m and put them into the candidate item set, given they are unknown to user u. The top-N items in the candidate item set are then recommended to the user u.

6.3.14 Algorithm 14: Item-Based Hierarchical Clustering

Given the item-tag frequency data set M, we build hierarchical groups of items. For each item i of the target user u, we get the assigned cluster m. If user u has not seen a neighbor item from cluster m, this item is added to the candidate item set. From the candidate item set, we then generate the top-N items and recommend them to the user u.

6.3.15 Algorithm 15: User-Based Apriori Rule Mining

In this algorithm, we use the association rule mining approach [29]. We assume that each user represents a transaction while the tags he uses are the items of this transaction. We apply the Apriori algorithm [28] on the user-tag binary data set B after defining a minimum support, a minimum confidence, and the number of rules that should be learned. When all rules are found, for each user u we perform the following steps: for each rule whose head contains tags of the user u, we get the tags from the body of the rule that are new for u. We query the database for all items that are annotated by these tags and add them to the candidate item list. The target user u is then recommended the top-N items in the candidate item list.

6.3.16 Algorithm 16: Item-Based Apriori Rule Mining

This algorithm is very similar to Algorithm 15. The only difference is that the association rules are learned from the item-tag binary data set L.

7 Evaluation Results

In the following sections, we present the evaluation details of the different tag-based CF recommendation algorithms outlined above. The data set we have used in our experiments is the PLEM data with 83 users, 773 items, and 1,005 tags in their stemmed version. The constructed user-tag matrix is a $83 \times 1{,}005$ matrix with a sparsity level of 0.9823, and the item-tag matrix is a $773 \times 1{,}005$ matrix with a sparsity level of 0.9968. The sparsity level is defined as the relation $1 - \frac{nonzero\ entries}{total\ entries}$ [31]. We apply two different evaluation methodologies, namely an offline evaluation and a user evaluation.

7.1 Offline Evaluation

We conducted an offline evaluation to gauge the performance of all the proposed algorithms. This process required additional experimental tuning of the parameters of all used data mining techniques.

7.1.1 Evaluation Methodology and Metrics

To evaluate the accuracy of the 16 proposed algorithms, we have measured three evaluation metrics for each approach: precision, recall, and F1 score. Each recommendation was set to generate a list of top $N = 10$ items most likely to be interesting for a given target user. We use the complete data set as testing data for evaluating the different tag-based CF approaches. We compute the true positive (TP) items in the top-N items. The TP rate is the number of recommended top-N items that appear also in the set of already saved items I_u of the target user u. The precision for user u is calculated as $precision(u) = \frac{|TopN_u \cap I_u|}{TopN_u}$, while the recall is given by: $recall(u) = \frac{|TopN_u \cap I_u|}{I_u}$. The precision of a recommendation algorithm X is averaged over the sum of the precisions for all users k:

$$Precision(X) = \frac{\sum_{n=1}^{k} precision(u)}{k} \times 100. \quad (1)$$

Similarly, the average recall of approach X is computed as

$$Recall(X) = \frac{\sum_{n=1}^{k} recall(u)}{k} \times 100. \quad (2)$$

Due to the conflicting nature of the previous two evaluation metrics [31], we further consider their harmonic mean—the F1 score: $F_1 = \frac{2 * Precision(X) * Recall(X)}{Precision(X) + Recall(X)}$.

We performed extensive experimental tuning for all proposed algorithms by utilizing these three metrics together with the rate of recommendation misses; i.e., the

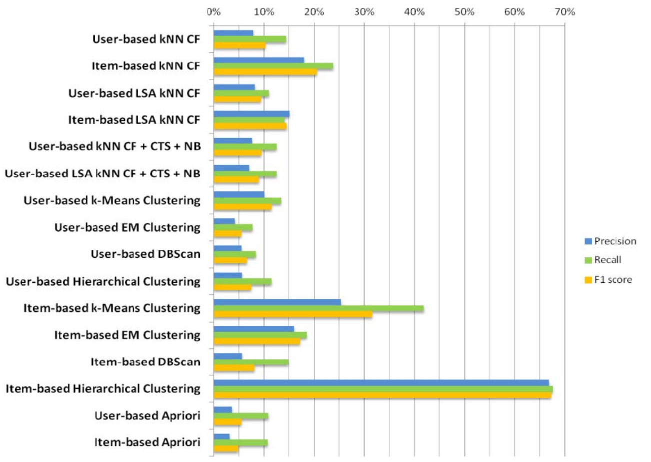
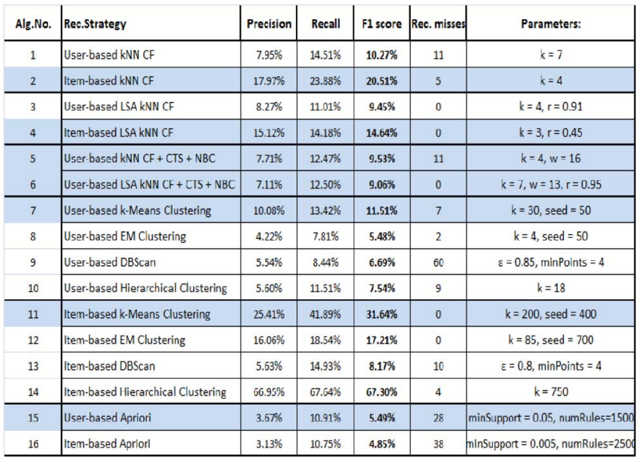


Fig. 2. Evaluation of the proposed recommendation algorithms in terms of precision, recall, and F1 score.



Alg.No.	Rec.Strategy	Precision	Recall	F1 score	Rec. misses	Parameters:
1	User-based kNN CF	7.95%	14.51%	**10.27%**	11	k = 7
2	Item-based kNN CF	17.97%	23.88%	**20.51%**	5	k = 4
3	User-based LSA kNN CF	8.27%	11.01%	**9.45%**	0	k = 4, r = 0.91
4	Item-based LSA kNN CF	15.12%	14.18%	**14.64%**	0	k = 3, r = 0.45
5	User-based kNN CF + CTS + NBC	7.71%	12.47%	**9.53%**	11	k = 4, w = 16
6	User-based LSA kNN CF + CTS + NBC	7.11%	12.50%	**9.06%**	0	k = 7, w = 13, r = 0.95
7	User-based k-Means Clustering	10.08%	13.42%	**11.51%**	7	k = 30, seed = 50
8	User-based EM Clustering	4.22%	7.81%	**5.48%**	2	k = 4, seed = 50
9	User-based DBScan	5.54%	8.44%	**6.69%**	60	ε = 0.85, minPoints = 4
10	User-based Hierarchical Clustering	5.60%	11.51%	**7.54%**	9	k = 18
11	Item-based k-Means Clustering	25.41%	41.89%	**31.64%**	0	k = 200, seed = 400
12	Item-based EM Clustering	16.06%	18.54%	**17.21%**	0	k = 85, seed = 700
13	Item-based DBScan	5.63%	14.93%	**8.17%**	10	ε = 0.8, minPoints = 4
14	Item-based Hierarchical Clustering	66.95%	67.64%	**67.30%**	4	k = 750
15	User-based Apriori	3.67%	10.91%	**5.49%**	28	minSupport = 0.05, numRules=1500
16	Item-based Apriori	3.13%	10.75%	**4.85%**	38	minSupport = 0.005, numRules=2500

Fig. 3. Summary of best performing algorithms and the respective input parameters.

number of users to whom the algorithm was not able to provide a recommendation. The goal was to estimate the optimal input parameter(s) for each algorithm. We then compared the performance of the proposed recommendation algorithms. Finally, from each CF recommendation category we selected those algorithms providing the best recommendation and deployed them for the user evaluation.

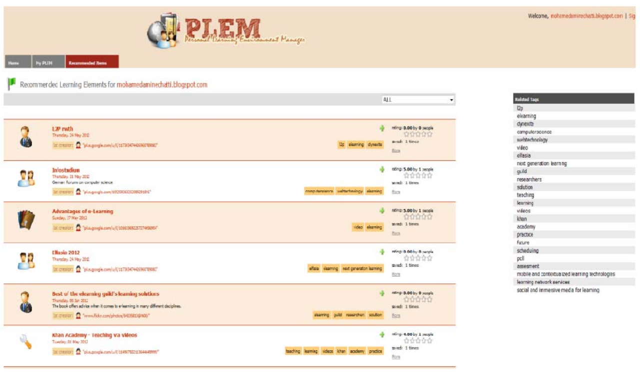
7.1.2 Comparison of Algorithms

Considering the obtained best input parameters for each approach, Fig. 2 depicts the performance of all 16 proposed algorithms. In general, the item-based variants of the different categories of algorithms outperformed the user based, with exception for the Apriori approach. The *Item-Based Hierarchical Clustering* achieved the best performance. We believe, however, that this is only due to overfitting the model. If we compare the item-based clustering techniques for the same number of clusters k ($k = 200$), the *Item-Based k-Means Clustering* approach takes the lead with an F1 score of 31.64 percent. It is followed by the *Item-Based kNN CF* with an F1 score of 20.51 percent. From the other techniques, the *Item-Based LSA kNN CF* and the *User-Based k-Means Clustering* achieve also good precision and recall. The worst recommendation performance was observed for the association rule mining approaches, as well as for some of the user-based clustering techniques (i.e., EM, DBSCAN, and hierarchical clustering). Association rule mining approaches did not perform well due the lack of frequent tag item sets as a result of the different tagging behavior of the users. The relative bad performance of some of the user-based clustering techniques can be explained by the fact that users have different tagging interests and, thus, do not use the same tags to annotate the same items. In fact, in our experiment, the user distribution among clusters results in 1-2 big clusters, a few clusters with just 2-3 users, and most of the users, especially in the User-based DBScan case, were assigned to single clusters. The algorithm could not make recommendation to those users as there are no other users in their clusters.

Fig. 3 gives a summary of best performing algorithms and their respective input parameters. Algorithms 2, 4, 5, 6, 7, 11, and 15, marked in blue, represent the approaches that have been selected for the user evaluation. The aim was to select one algorithm per category. The selection criteria were two: first, high value for the F1 score, and, second, minimum number of recommendation misses. For the category of classification methods, we have selected both algorithms, as approach 6 suggests an improvement for Algorithm 5, originating from Ji et al. [19].

7.2 User Evaluation

The offline evaluation aimed to compare different tag-based CF recommendation algorithms and to select those that can offer the best recommendation accuracy based on three evaluation metrics: precision, recall, and F1 score. However, as mentioned earlier, these metrics do not measure the user satisfaction with the recommended items. The quality of user experience often does not correlate with high recommendation accuracy measured by these metrics [32]. Thus, we conducted a user evaluation on the seven selected recommendation approaches, as discussed in the previous section. We evaluated the recommendation module in PLEM with students and teaching assistants in the "Advanced Learning Technologies" course offered at RWTH Aachen University in Spring 2012. The participants were asked to collect, manage, and tag learning items related to the topics of eLearning and web technologies. A screenshot of the PLEM recommendation user interface is shown in Fig. 4.


PLEM
Personal Learning Environment Manager
Welcome, mohamedaminechatti.blogspot.com | Sig
Home
My PLEM
Recommended Items
ALL
Related Tags

Recommended Learning Elements for mohamedaminechatti.blogspot.com

L2P rwth
Thursday, 24 May 2012
l2p elearning dynexite
rating: 0.00 by 0 people
saved: 1 times

Infostudium
Thursday, 31 May 2012
German Forum on computer science
computerscience webtechnology elearning
rating: 5.00 by 1 people
saved: 1 times

Advantages of e-Learning
Sunday, 27 May 2012
video elearning
rating: 5.00 by 1 people
saved: 1 times

Elfasia 2012
Thursday, 24 May 2012
elfasia elearning next generation learning
rating: 0.00 by 0 people
saved: 1 times

Best of the elearning guild's learning solutions
Thursday, 05 Jan 2012
The book offers advice when it comes to e-learning in many different disciplines.
elearning guild researchers solution
rating: 0.00 by 0 people
saved: 1 times

Khan Academy - Teaching via videos
Tuesday, 01 May 2012
teaching learning videos khan academy practice
rating: 4.00 by 1 people
saved: 1 times

l2p
elearning
dynexite
computerscience
webtechnology
video
elfasia
next generation learning
guild
researchers
solution
teaching
learning
videos
khan
academy
practice
future
scheduling
pcll
assessment
mobile and contextualized learning technologies
learning network services
social and immersive media for learning

Fig. 4. Recommendation section in PLEM.

TABLE 1
Selected Algorithms for User Evaluation

Rec. No.	Algorithm
Rec 1	Item-based kNN CF (IB kNN CF)
Rec 2	Item-based LSA kNN CF (IB LSA CF)
Rec 3	User-based kNN with CTS generation and NBC (CTS + NBC)
Rec 4	User-based LSA kNN with CTS generation and NBC (LSA + CTS + NBC)
Rec 5	User-based k-Means Clustering (UB k-Means)
Rec 6	Item-based k-Means Clustering (IB k-Means)
Rec 7	User-based Apriori (UB Apriori)

7.2.1 Online Questionnaire

For capturing the user experience with the recommendation module in PLEM, we created an online questionnaire. The questions were separated in three main groups. The first section covered demographic and general information. The second group of questions aimed to gather insight about previous knowledge and experience of the participants with recommender systems. The third section was devoted to evaluate each of the even selected recommendation algorithms (see Table 1). It examines how the respondents perceive the recommendation quality and usefulness of the different algorithms. We have leveraged the framework ResQue, developed by Pu et al. [32] for user-centric evaluation of recommender systems. The model consists of 13 constructs that contain in total 60 questions divided into four main categories: user perceived qualities, user beliefs, user attitudes, and behavioral intentions, as depicted in Fig. 5. The users are asked to answer on a 5-point Likert scale, where 1 equals to "strongly disagree" and 5 to "strongly agree." We have selected six sample questions, reworked and adapted them to gauge recommendation in a learning context. This resulted in a list of seven questions as given below:

1. *Q1—Ability to recommend*. The system is able to provide recommendation for me. (Y / N)
2. *Q2—Accuracy*. In my opinion, the system is able to recommend to me 1-3 / 4-6 / 7-10 interesting or relevant learning items.
3. *Q3—Novelty*. The learning items recommended to me are novel and still interesting.
4. *Q4—Diversity*. The learning items recommended to me are diverse (not all of them are similar to each other).
5. *Q5—Context compatibility*. The recommended learning items take my tag preferences into consideration.
6. *Q6—Perceived usefulness*. I feel supported in finding learning items that I like with the help of the recommendation.
7. *Q7—Attitude*. Overall, I am satisfied with the recommendation.

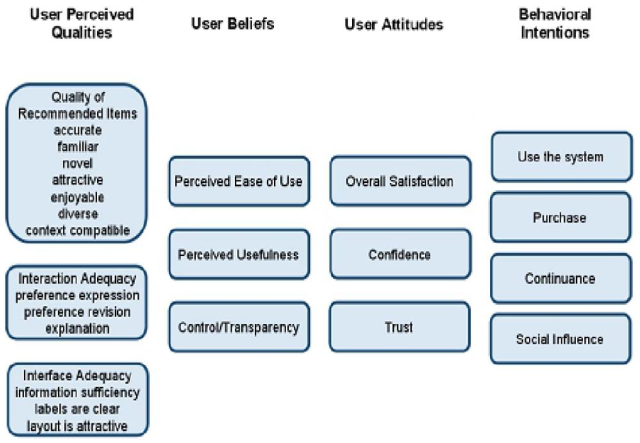

User Perceived Qualities
User Beliefs
User Attitudes
Behavioral Intentions
Quality of Recommended Items accurate familiar novel attractive enjoyable diverse context compatible
Interaction Adequacy preference expression preference revision explanation
Interface Adequacy information sufficiency labels are clear layout is attractive
Perceived Ease of Use
Perceived Usefulness
Control/Transparency
Overall Satisfaction
Confidence
Trust
Use the system
Purchase
Continuance
Social Influence

Fig. 5. Constructs of the evaluation framework ResQue [32].

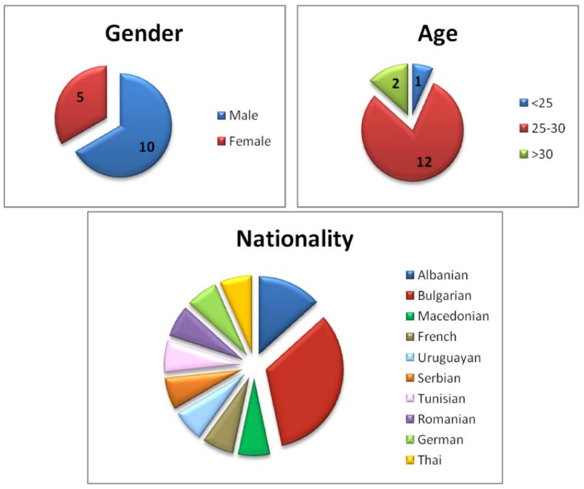


Fig. 6. User profiles.

7.2.2 Discussion of User Evaluation Results

Questionnaires were sent to 15 evaluators. Altogether, five females and 10 males from different nationalities tested the recommender system. The evaluators were students and teaching assistants from computer science at the age range 24-34 years. The spectrum of profiles based on gender, age, nationality is summarized in Fig. 6.

Most of the evaluators (67 percent) have had prior experience with recommender systems. However, in regards to trust only (47 percent) of the respondents feel confident with such systems. Fig. 7 summarizes these results.

Further, we were interested in the testers' experience specifically with tag-based recommendation: 67 percent agreed to be familiar with it, while 27 percent disagreed. Lastly, the majority of users consider it useful to have recommendations in learning environments based on the behavior of like-minded learners (see Fig. 8).

The main aim of the user evaluation was to gauge the user satisfaction with the recommendation results. In the following, we focus on the perceived quality and usefulness of the best performing seven recommendation algorithms

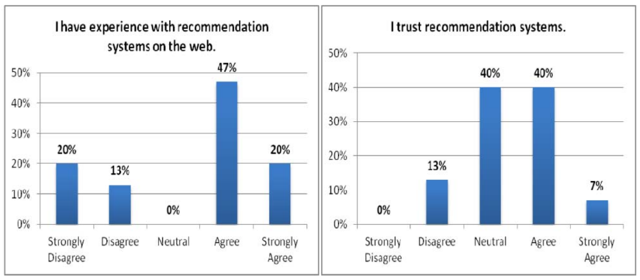


Fig. 7. Users' experience and trust in recommender systems.

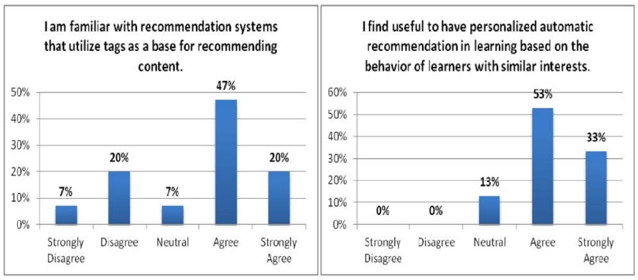


Fig. 8. Familiarity with tag-based recommendation and attitude toward recommendation in learning.

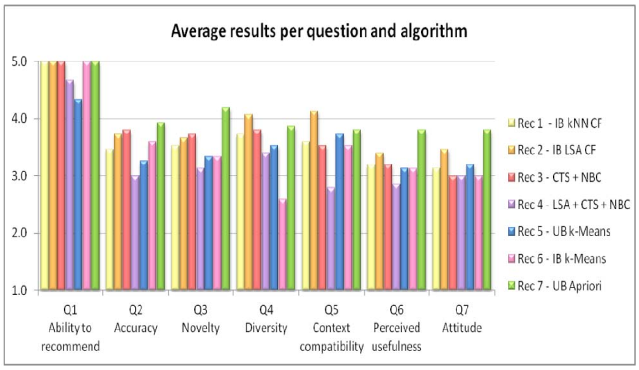


Fig. 9. Summary of results per question and algorithm.

from our offline evaluation (see Table 1). We analyse the results per question basis for all approaches. A summary of the average scores per question and per algorithm are given in Fig. 9, based on the 5-point Likert scale where 1 indicates "strongly disagree" and 5 "strongly agree."

The first question (Q1—Ability to recommend) investigates if the proposed recommendation algorithms are able to make any suggestions for the target users. All techniques have a success rate above 80 percent. The algorithms that fail in providing suggestions to all users are the *User-Based LSA kNN with CTS Generation and NBC* and the *User-Based k-Means Clustering*. All other approaches have successfully recommended items to all evaluators. Further, we have observed that the number of tags in the user profile does not relate to the fact whether a user will receive a recommendation or not. The system could recommend items to participants that have used less than 10 tags.

However, in regards to the percentage of accurately suggested items (Q2—Accuracy) the number of applied tags do play a role. Most users with less than 25 applied tags receive less relevant items. The larger the tag set of one user, the more relevant items are recommended. From Fig. 9, we can observe that the outstanding approaches are the *User-Based Apriori* and *User-Based kNN with CTS Generation and NBC*. The proposed alternative for the last algorithm using LSA (Rec 4), as well as the *User-Based k-Means Clustering* suggest the least interesting items to the user. Another observation we made is that the *Item-Based k-Means Clustering* algorithm can always offer at least one to three relevant items.

Following, we evaluated the ability of the proposed recommendation algorithms to provide novel and attractive items to the users (Q3—Novelty). The *User-Based LSA kNN with CTS Generation and NBC* and both the user-based and the item-based clustering methods (Rec 5 and 6) had the worst performance, while the *User-Based Apriori* and the *User-Based kNN with CTS Generation and NBC* received the highest scores.

In terms of diversity (Q4—Diversity), the *Item-Based LSA kNN CF* and the association rule mining approach achieved the best scores. The *Item-Based k-Means Clustering* is the algorithm that offered minimal diversity in recommendation. This is not surprising as clustering aims to group similar items, thus diversity within a cluster cannot be achieved.

The users were asked to evaluate to what extent the provided recommendation incorporate their tag preference (Q5—Context Compatibility). The *Item-Based LSA kNN CF*, the *User-Based Apriori*, and the *User-Based Clustering* achieved the best acceptance. The *User-Based LSA kNN with CTS Generation and NBC* algorithm is the only algorithm below the average with a score of 2.8.

Another important aspect for us is how the participants perceive the support that the recommendation algorithms offer in the context of learning Q6—Perceived Usefulness. With a significant lead of 3.8 score, the *User-Based Apriori* seems to be the best recommendation approach that helps in the discovery of quality items for learning purposes. Second best, scores the *Item-Based LSA kNN CF*. All other algorithms shared similar perceived usefulness.

Lastly, we analyzed the overall satisfaction of the user with the recommendation provided by all seven algorithms Q7—Attitude. Not surprisingly, the leader in terms of accuracy, novelty, and perceived usefulness; i.e., the *User-Based Apriori* algorithm, received the best score, followed by the *Item-Based LSA kNN CF* that performed best in regards to diversity and consideration of users' tag preferences. The *User-Based k-Means Clustering* and the *Item-Based kNN CF* algorithms received ranks 3, and, respectively, 4. Both classification approaches and the *Item-Based k-Means Clustering* received the same score of 3.0.

In conclusion, according to the evaluators, the best recommendation is achieved by association rule mining of users' tags, as well as by item-based CF on a lower dimensional semantic space. Independent of the specific approach, the good overall satisfaction with all techniques proves that tag-based collaborative filtering has high potential for recommendation tasks in PLEs.

7.3 Offline versus User Evaluation

The results from our user evaluation confirmed the claim of Pu et al. [32] that the quality of user experience with recommender systems does not meet the high accuracy performance measured by metrics, such as precision and recall. In our evaluation, the *User-Based Apriori* algorithm that achieved a low F1 score of only 4.85 percent in the offline evaluation was rated by users as the best recommendation algorithm. And, the evaluators were less satisfied with the recommendation of the *Item-Based k-Means Clustering* algorithm, which ranked first in the offline evaluation. According to the measured precision and recall, the *Item-Based LSA kNN CF* and the *User-Based k-Means Clustering* algorithms ranked third, respectively, fourth in the offline evaluation. However, the evaluators found their recommendation as more useful than the *Item-Based kNN CF*

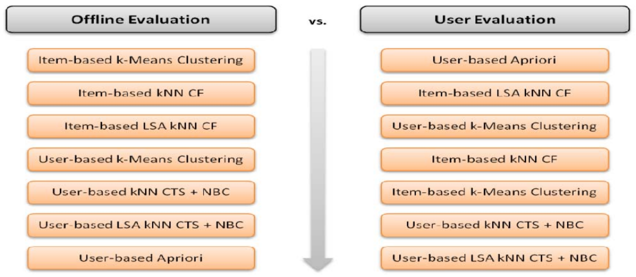


Fig. 10. Comparison of the ranking of algorithms from the offline and user evaluation.

algorithm, which had the second place in the offline evaluation. Fig. 10 depicts the ranking of the seven best performing algorithms in descending order according to the findings from our offline and user evaluation.

The evaluation results show that it is inappropriate to rely solely on offline evaluation to gauge the effectiveness of tag-based recommender systems. We believe that user evaluation methods are more suitable when assessing the recommendation quality of tag-based collaborative filtering algorithms in PLEs. The perceived usefulness of recommended items depends on the learners' subjective opinion and background knowledge. Statistically measured accuracy does not correlate with the individual understanding of relevant and interesting items. Most probably, this is due to the fact that tags are instruments of natural language and hold semantics that cannot be captured completely by recommendation algorithms.

8 Discussion and Future Work

The results of the evaluation in this experiment revealed that the *Item-Based k-Means Clustering* was the best performing algorithm in the offline evaluation whereas the user-based Apriori algorithm was ranked first in the user evaluation. The major finding in this study is that the offline evaluation of recommender systems does not always correlate with their user evaluation. However, due to the small sample size (83 users, 773 items, and 1,005 tags), it is not possible at this stage of research to generalize that the *Item-Based k-Means Clustering* and *User-Based Apriori* are always the best performing algorithms in tag-based CF recommendation tasks. To draw similar conclusions, it is necessary to apply the recommendation techniques used in this study on other (educational) data sets, such as Delicious, MACE, ReMashed, Mendeley, and PSLC DataShop [33].

From a recommendation point of view, there are several directions for improvement. First, we can enhance the way the user profiles are built. For example, we can ignore tags that occur rarely. Second, we can improve also the item profiles. This can be done by extracting tags for a given item from third-party services like del.icio.us. Furthermore, the problem of polysemy and synonymy of tags can be reduced by leveraging a dictionary like WordNet, Wikipedia, or Freebase to compute whether two tags are semantically similar.

It is important to note that the data set used in our study was generated within the PLEM environment. The methods of the study can, however, be applied in other (TEL) environments, where users manage and tag items. From a technical perspective, this should be possible because the recommendation methods that we implemented in this experiment just require as input a user-tag matrix and an item-tag matrix that can easily be built from any available tag-based data set.

9 Summary

In a PLE-driven approach to learning, there is a crucial need for recommendation methods to help learners find quality knowledge nodes (i.e., information and people) that can populate their PLEs. In this paper, we investigated the application of tag-based CF recommendation methods to recommend learning items in PLEs. We implemented and experimented with 16 different tag-based CF algorithms, memory based as well as model based. We conducted an extensive offline and user evaluation to contrast and compare the different algorithms in terms of accuracy and user satisfaction. The results of the evaluation carried out confirm that the quality of user experience does not correlate with high-recommendation accuracy measured by statistical methods.

References

[1] M.A. Chatti, M. Jarke, and M. Specht, "The 3P Learning Model," *J. Educational Technology and Soc.*, vol. 13, no. 4, pp. 74-85, 2010.

[2] K. Verbert, H. Drachsler, N. Manouselis, M. Wolpers, R. Vuorikari, and E. Duval, "Data Set-Driven Research for Improving Recommender Systems for Learning," *Proc. First Int'l Conf. Learning Analytics and Knowledge (LAK '11)*, pp. 44-53, 2011.

[3] H. Drachsler, H. Hummel, B. van den Berg, J. Eshuis, W. Waterink, R. Nadolski, A. Berlanga, N. Boers, and R. Koper, "Effects of the Isis Recommender System for Navigation Support in Self-Organized Learning Networks," *J. Educational Technology and Soc.*, vol. 12, no. 3, pp. 115-126, 2009.

[4] J. Buder and C. Schwind, "Learning with Personalized Recommender Systems: A Psychological View," *Computers in Human Behavior*, vol. 28, no. 1, pp. 207-216, 2012.

[5] N. Manouselis, H. Drachsler, R. Vuorikari, H. Hummel, and R. Koper, "Recommender Systems in Technology Enhanced Learning," *Recommender Systems Handbook*, L.R.P.B. Kantor, F. Ricci, and B. Shapira, eds., pp. 387-415, Springer, 2011.

[6] N. Manouselis, H. Drachsler, K. Verbert, and E. Duval, *Recommender Systems for Learning*. Springer, 2013.

[7] F. Mödritscher, "Towards a Recommender Strategy for Personal Learning Environments," *Proc. First Workshop Recommender Systems for Technology Enhanced Learning*, vol. 1, no. 2, pp. 2775-2782, 2010.

[8] G. Adomavicius and A. Tuzhilin, "Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions," *IEEE Trans. Knowledge and Data Eng.*, vol. 17, no. 6, pp. 734-749, June 2005.

[9] X. Su and T.M. Khoshgoftaar, "A Survey of Collaborative Filtering Techniques," *Advances in Artificial Intelligence*, vol. 2009, article 4, http://www.hindawi.com/j.s/aai/2009/421425/, Jan. 2009.

[10] S. Alag, *Collective Intelligence in Action*. Manning Publications Co., 2009.

[11] T.C. Zhou, H. Ma, I. King, and M.R. Lyu, "TagRec: Leveraging Tagging Wisdom for Recommendation," *Proc. Int'l Conf. Computational Science and Eng. (CSE '09)*, vol. 4, pp. 194-199, 2009.

[12] R. Vuorikari, M. Sillaots, S. Panzavolta, and R. Koper, "Are Tags from Mars and Descriptors from Venus? A Study on the Ecology of Educational Resource Metadata," *Proc. Eighth Int'l Conf. Advances in Web Based Learning (ICWL '09)*, pp. 400-409, 2009.

[13] R. Vuorikari and X. Ochoa, "Exploratory Analysis of the Main Characteristics of Tags and Tagging of Educational Resources in a Multi-Lingual Context," *J. Digital Information*, vol. 10, no. 2, http://journals.tdl.org/jodi/index.php/jodi/article/view/447, 2009.

[14] M.-H. Hsu and H.-H. Chen, "Tag Normalization and Prediction for Effective Social Media Retrieval," *Proc. IEEE/WIC/ACM Int'l Conf. Web Intelligence and Intelligent Agent Technology,* vol. 1, pp. 770-774, 2008.

[15] S. Sen, J. Vig, and J. Riedl, "Tagommenders: Connecting Users to Items through Tags," *Proc. 18th Int'l Conf. World Wide Web,* pp. 671-680, 2009.

[16] D. Parra-Santander and P. Brusilovsky, "Improving Collaborative Filtering in Social Tagging Systems for the Recommendation of Scientific Articles," *Proc. IEEE/WIC/ACM Int'l Conf. Web Intelligence and Intelligent Agent Technology,* pp. 136-142, 2010.

[17] D. Zeng and H. Li, "How Useful Are Tags? An Empirical Analysis of Collaborative Tagging for Web Page Recommendation," *Proc. IEEE ISI 2008 PAISI, PACCF, and SOCO Int'l Workshops Intelligence and Security Informatics,* pp. 320-330, 2009.

[18] A.K. Milicevic, A. Nanopoulos, and M. Ivanovic, "Social Tagging in Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions," *Artificial Intelligence Rev.,* vol. 33, no. 3, pp. 187-209, 2010.

[19] A.-T. Ji, C. Yeon, H.-N. Kim, and G.-S. Jo, "Collaborative Tagging in Recommender Systems," *Proc. 20th Australian Joint Conf. Advances in Artificial Intelligence (AI '07),* pp. 377-386, 2007.

[20] K.H.L. Tso-Sutter, L.B. Marinho, and L. Schmidt-Thieme, "Tag-Aware Recommender Systems by Fusion of Collaborative Filtering Algorithms," *Proc. ACM Symp. Applied Computing (SAC '08),* pp. 1995-1999, 2008.

[21] H. Liang, Y. Xu, Y. Li, and R. Nayak, "Tag Based Collaborative Filtering for Recommender Systems," *Proc. Int'l Conf. Rough Sets and Knowledge Technology,* vol. 5589, pp. 666-673, 2009.

[22] Z. Wang, Y. Wang, and H. Wu, "Tags Meet Ratings: Improving Collaborative Filtering with Tag-Based Neighborhood Method," *Proc. Workshop Social Recommender Systems (SRS '10),* 2010.

[23] C.S. Firan, W. Nejdl, and R. Paiu, "The Benefit of Using Tag-Based Profiles," *Proc. Latin Am. Web Conf. (LA-WEB '07),* pp. 32-41, 2007.

[24] R.Y. Nakamoto, S. Nakajima, J. Miyazaki, S. Uemura, H. Kato, and Y. Inagaki, "Reasonable Tag-Based Collaborative Filtering for Social Tagging Systems," *Proc. Second ACM Workshop Information Credibility Web (WICOW '08),* pp. 11-18, 2008.

[25] M.A. Chatti, Anggraeni, M. Jarke, M. Specht, and K. Maillet, "PLEM: A Web 2.0 Driven Long Tail Aggregator and Filter for E-Learning," *Int'l J. Web Information Systems,* vol. 6, no. 1, pp. 5-23, 2010.

[26] D. Verpoorten, C. Glahn, M.A. Chatti, W. Westera, and M. Specht, "Self-Reported Learning Effects of a Tagging Activity Carried out in a Personal Learning Environment (PLE) by Secondary-School Pupils," *Int'l J. Cross-Disciplinary Subjects in Education,* vol. 2, no. 1, pp. 276-284, 2011.

[27] E. Hatcher and O. Gospodnetic, *Lucene in Action.* Manning Publications Co., 2004.

[28] X. Amatriain, A. Jaimes, N. Oliver, and J.M. Pujol, "Data Mining Methods for Recommender Systems," *Recommender Systems Handbook,* F. Ricci, L. Rokach, B. Shapira, and P.B. Kantor, eds., pp. 39-72, Springer, 2011.

[29] I.H. Witten, E. Frank, and M.A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques.* Morgan Kaufmann, 2011.

[30] D. Breitkreutz and K. Casey, "Clusterers: A Comparison of Partitioning and Density-Based Algorithms and a Discussion of Optimisations," technical report, http://eprints.jcu.edu.au/11999/, 2008.

[31] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, "Item-Based Collaborative Filtering Recommendation Algorithms," *Proc. 10th Int'l World Wide Web Conf. (WWW '10),* pp. 285-295, 2001.

[32] P. Pu, L. Chen, and R. Hu, "A User-Centric Evaluation Framework for Recommender Systems," *Proc. Fifth ACM Conf. Recommender Systems,* pp. 157-164, 2011.

[33] K. Verbert, N. Manouselis, H. Drachsler, and E. Duval, "Data Set-Driven Research to Support Learning and Knowledge Analytics," *Educational Technology and Soc.,* vol. 15, no. 3, pp. 133-148, 2012.

Mohamed Amine Chatti received the diploma degree in computer science from the University of Kaiserslautern, Germany, in 2004 and the PhD degree in computer science from the RWTH Aachen University, Germany, in 2010. He is an assistant professor of computer science in the Learning Technologies Group (Informatik 9) at the RWTH Aachen University, Germany. His research focuses on web information systems, technology-enhanced learning, and knowledge management.

Simona Dakova is working toward the master's degree in computer science at the RWTH Aachen University.

Hendrik Thüs received the diploma degree in computer science from the RWTH Aachen University in 2010. He is working toward the PhD degree with the Learning Technologies Research Group (Informatik 9) of the RWTH Aachen University, Germany, where he is a research assistant. He focuses on mobile learning in different context situations and on the generation of user profiles according to their usage of media.

Ulrik Schroeder is a professor of computer science at the RWTH Aachen University. He heads the Learning Technologies Research Group. He is also the head of the Center for Innovative Learning Technology (CiL) and the director of the school laboratory for computer science (InfoSphere) at RWTH Aachen University. His research interests include assessment and intelligent feedback, mobile learning, gender mainstreaming in education, and computer science teachers education.