Semantic Web 0 (0) 1
IOS Press

(Partial) User Preference Similarity as Classification-Based Model Similarity

Editor(s): Claudia d'Amato, Universita degli Studi di Bari, Italy
Solicited review(s): Vojtech Svatek, University of Economics, Prague, Czech Republic; Alexandra Moraru, Jozef Stefan Institute, Slovenia; anonymous reviewer

Amancio Bouza [a], Abraham Bernstein [a]
[a] *Department of Informatics, University of Zurich, Binzmühlestrasse 14, 8050 Zurich, Switzerland*
E-mail: {bouza,bernstein}@ifi.uzh.ch

Abstract. Recommender systems play an important role in helping people finding items they like. One type of recommender system is collaborative filtering that considers feedback of like-minded people. The fundamental assumption of collaborative filtering is that people who previously shared similar preferences behave similarly later on. This paper introduces several novel, classification-based similarity metrics that are used to compare user preferences. Furthermore, the concept of partial preference similarity based on a machine learning model is presented. For evaluation the cold-start behavior of the presented classification-based similarity metrics is evaluated in a large-scale experiment. It is shown that classification-based similarity metrics with machine learning significantly outperforms other similarity approaches in different cold-start situations under different degrees of data-sparseness.

Keywords: User Similarity, Partial User Similarity, Collaborative Filtering, Cold-Start Problem

1. Introduction

The vast amount of consumables and items (e.g., movies, books, restaurants) offered by Web stores or Web guides provides an amazing number of options to people but also challenges people with choosing proper options. Traditional information filtering techniques (e.g., keyword-based filtering approaches) are not suitable to reduce the vast amount of items to a reasonable size. Additionally, they do not consider people's personal preferences to filter proper items. Thus, people need to invest a lot of effort to filter proper items.

For that reason, recommender systems gain popularity and are employed in Web stores (e.g., Amazon [18]) to help people in finding proper items. The difference between traditional information filtering (IF) techniques and recommender systems is that recommender systems propose proper items to the user whereas IF lets the user search for these.

One type of recommender system is so-called *collaborative filtering*, which considers feedback of like-minded people to provide recommendations. The fundamental assumption of collaborative filtering is that people who previously shared similar preferences behave similarly later on. Consequently, each user benefits from the previous actions of like-minded users. Prior research proposed different similarity metrics to compute the similarities between two users' preferences and to discover like-minded people based on items that both users have rated (i.e., *common rated items*). Based on these similarity metrics, collaborative filtering provides accurate recommendations as long as people provide many ratings.

However, collaborative filtering does not provide accurate recommendations to people who have few prior ratings in general. This problem – known as the *cold-start problem* – commonly affects collaborative filtering-based recommender systems. The reasons are

twofold. First, few ratings generally do not adequately express the whole spectrum of a user's preferences.

Second, even when two users provide many ratings, common rated items may not be a representative sample of both users's preferences, especially when people share ratings only for few items. Furthermore, people who share similar preferences for specific types of consumable may not necessarily share similar preferences in general. In other words, they share partially similar preferences. Hence, similarity metrics that compute an overall user preference similarity do not account for partially similar preferences.

In contrast to collaborative filtering, applying machine learning to model or rather hypothesize the user's preferences to provide recommendations has been found effective in cold-start situations similar to collaborative filtering [29]. Machine learning is applied to generalizes from a user's ratings to the user's preferences. But such models are limited to the general preferences of a user and do not consider the individual quality of a consumable or item. For instance, a user may prefer action movies, but not every action movie is a good one.

This paper provides an empirical study, which compares the retrieval of like-minded users which is used in collaborative filtering. More precisely, the retrieval of like-minded people are compared, which is based either on ratings for common rated items or on hypothesized user preferences. To this goal, this paper introduces a formal framework for the comparison of user preferences based on classification similarity of the respective hypothesized user preference models. The user preference models are built with machine learning algorithms. The formal modal is instantiated in two concrete methods to compare hypothesized user preferences. In addition, a preprocessing step for methods is introduced that allows the comparison of partial preferences. More precisely, partial preferences are extracted from the representation of a machine learning model. The new methods — with and without preprocessing — are then compared in conjunction with different machine learning algorithms with the state-of-the-art of user-based collaborative filtering approaches.

The reminder of this paper is as follows: In Section 2, previous research on user preference modeling, and collaborative filtering is presented. Then, the formal framework is introduced in Section 3 leading to an explanation on how user preferences are hypothesized with machine learning in Section 4. Section 5 presents two similarity metrics to compare user preference models and additionally, conceptualizes partial preferences together with a partial similarity metric. In Section 6, the performance of the presented approaches and their behavior in different cold-start situations is evaluated and compared to other, traditional collaborative filtering approaches. Specifically, the recommendation performance is analyzed for data sets of different rating sparseness degrees. Finally, conclusions are drawn from the evaluation in the final Section 7 together with the limitations and future work.

2. Related work

Collaborative filtering is a method to provide personalized recommendations to a user by considering the preferences from many other users [27,16,17]. The underlying assumption of collaborative filtering is that people who shared the same preferences tend to share the same preferences in the future. To provide personalized recommendations, preferences need to be formalized and conceptualized. Built on this, similarities among user preferences are computed to determine like-minded people. Afterwards, collaborative filtering is applied to provide personalized recommendations to a user by considering prior actions of like-minded users.

2.1. User preference modeling

To provide personalized recommendations and recognize like-minded people, a user's preferences needs to be modeled first. Commonly, user preferences are represented as an item rating vector with each element representing the user's rating for the corresponding item [8,26]. On the one side, no additional content information is required. On the other side, sparse item rating vectors model a user's preferences poorly and thus, results in poor recommendations.

For this reason, [13] proposes to represent a user's preferences as a topic preference vector instead, where items are assigned to one or more topics. Each value of the topic preference vector describes the user's preferences for the corresponding topic. In [21], the synergy between ontologies and recommender systems are exploited. Similarly to [13], the user's preferences are modeled as *topics of interest* with topics being specified by an ontology. Each topic is associated with a score for every user individually which represents the users individual preference for a certain topic. Based

on this approach, semantic topic similarity is considered additionally in [22]. Likewise, a topic taxonomy is assumed in [33,35,1]. Scores for topics are passed to the corresponding super-topic. A more complex approach is proposed in [7]. Semantic concepts are used to build an ontology-based decision tree to represent a user's preferences. The ontology-based decision tree is then used to classify an item as relevant or not, an ontology-based decision tree learner is presented to hypothesize a user's preferences. The problem of these approaches is that depending on the specification of the topics the resulting recommendation are too generic.

A correlation between user preference similarity and trust is reported in [34]. Based on this relationship, [12] shows how trust in Web-based social networks can be used to filter like-minded people and provide accurate recommendations for movies. This approach does, however, require a trust network to make accurate recommendations.

2.2. User preference similarity

Generally, any vector similarity metric can be applied to compute the similarity between item rating vectors. Past research focused mainly on cosine similarity, Pearson correlation, and Spearman's rank correlation [14,8]. In addition to user preference similarity, [1] propose to describe items with semantical concepts and incorporate semantic item similarity to the computation of user preference similarity. Following the same idea, a multilayer ontology-based hybrid recommendation model is proposed in [10]. Similar items are clustered whereby each cluster represents a topic of interest. Users are related to these clusters to form *communities of interests* [11]. Then, all items liked by a community of interests is recommended to its members. However, the computational complexity of nearest neighbor-based collaborative filtering is $O(n^2)$ because the preference similarity of each user pair has to be computed. However, the computational complexity of computing the preference similarity of users in nearest neighbor-based collaborative filtering is $O(n^2)$ because the preference similarity of each user pair has to be computed. As empirically shown in [31], users have rather partially similar preferences than similar overall preferences. User-item subgroups are proposed by [31] to perform collaborative filtering. A user can belong to multiple user-item subgroups. To recommend an item to a user, the users within the corresponding subgroups are used to predict the relevance of the particular item. These subgroups contain a subset of users having similar preferences for a subset of items. These subgroups do not represent partial preferences because the items' properties can be completely different. Users are modeled with partial preference relations in [24]. The general idea is to cluster similar users based on their partial preferences. In fact, the preferences of users are bootstrapped by classifying the user to a cluster and inheriting partial preferences of other users of the same cluster.

To address the computational complexity cluster-based approach are proposed to reduce the number of comparisons. [32] cluster people according to their geographical location and compare preference similarities within a geographical cluster. Similarly, clustering user into groups of similar users is suggest in [25] with *ClustKNN*. Then, similarity is computed only between the centroids of these clusters. It is suggest to use k-Means clustering algorithms to control the tradeoff of computational complexity and recommendation accuracy which depends on the number of clusters. Empirical evidence is given in [25] that clustering techniques support the scalability of collaborative filtering whilst not sacrificing much recommendation accuracy. Given its reliance on item vector ratings, however, ClustKNN does not address the partial preference similarity problem.

In [3], social aspects of the users (e.g., age) and content-based information are used to compare users preferences. Specifically, single item properties are used to create item groups and preference similarity is then computed within such an item group. In other words, partial preference similarity is defined in [3] with respect to a single item property. The limitation of this approach is that it limits the partiality to one single, pre-defined item-property.

Generally, users differ in their rating behavior such that some users rate items on average higher then other users. Hence, it is shown in [4] that normalizing the users ratings improves the accuracy of computed user preference similarities and consequently results in more accurate recommendations.

2.3. General framework for collaborative filtering

A general framework for a nearest neighbor-based collaborative filtering is proposed in [14,26]. The general framework is shown in Eq. 1. The goal is to predict a rating $\widehat{r}_{a,j}$ for the active user a and item j based on his/her average rating $\overline{r}_a$ and of k nearest neighbors' (i.e., other k users with most similar preferences) weighted rating deviation $r_{b,j} - \overline{r}_b$ of that item.

$$\widehat{r}_{a,j} = \overline{r}_a + \frac{\sum_{b=1}^{k} w_{a,b} * (r_{b,j} - \overline{r}_b)}{\sum_{b=1}^{k} w_{a,b}} \quad (1)$$

Whilst many similarity metrics for the computation of $w_{a,b}$ have been proposed [14] show that the Pearson correlation outperforms the Spearman's rank correlation and cosine similarity.

2.4. Cold-start problem

All recommender systems face the cold-start problem when recommendations are required for users or items for which not enough information (i.e., ratings) is known. Given that users only use a recommender system that provides reasonable recommendations, providing poor recommendations will lead to user attrition. Hence, a recommender system may never achieve to attain a critical mass of ratings to provide reasonable recommendations. Thus, the challenge is to provide reasonable recommendation even with little information about the users' preferences.

Three different types of the cold-start problem can be distinguished [21]:

- *new-system cold-start* refers to the initial stage of a recommender system where no or only few initial ratings are provided. Every recommender system performs poorly in this situation because it lacks crucial information to build a good user profiles. Even content-based recommender systems need a few observations of the users' interest to provide reasonable recommendation.
- *new-user cold-start* refers to the situation where too few ratings for a user exist even in the light of sufficient information about others. Note that a similar problem may arise when ratings for some items are extremely sparse as the resulting set of common rated items may be small.
- *new-item cold-start* refers to the problem that arises with items that have no ratings yet.

Generally, content-based approaches are incorporated to counter the cold-start problem. Content filtering approaches are combined with collaborative filtering approaches to build hybrid recommender systems. A classification and survey of hybrid recommender system is found in [9]. One of the first hybrid recommender system is the *Fab system* [2], which recommends documents. Its users are asked to create a user profile by selecting topics of interest. Users are similar if they share many topics of interest. Documents are recommended when they match the user's profile and have been liked by users with similar user profiles. Whilst this approach addresses the cold-start problem its results are too generic and lack precision.

Another approach is to preprocess the data before replacing missing ratings with predicted ones. [20], e.g., predict unknown user ratings based on known ratings. For that purpose, a user's preferences are learned (or hypothesized) with a machine learning algorithm that predicts the ratings for not yet rated items. The mediation of user models across different domains is proposed in [5] for enhanced personalization of user models. It is empirically shown that like-minded people with respect to a domain tend to be like-minded in another domain. For instance, people sharing similar preferences for music tend to share similar preferences for movies. However, this is shown for overall preference similarity between users and does not necessarily apply to partial preference similarity between users.

3. Formal framework and notation

The basic elements for a collaborative filtering based recommender system is the set of users I, the set of items G, and the set of ratings R in which users explicitly or implicitly state about items. The rating set R is represented as the $m \times n$ rating matrix as it is shown in Figure 1 with $m = |I|$ number of users and $n = |G|$ number of items. A particular user is referred to as $i \in I$, a particular item as $g \in G$, and a particular rating of user i for item g as r_{ig}. Note that r_{ig} corresponds to the element at the ith row and gth column of the rating matrix R. User i's rating vector and item g's rating vector are referred tp as $R_{i\cdot}$ and $R_{\cdot g}$, respectively.

The value space of the ratings r_{ig} is denoted as K which consists of rating concepts C or $\varnothing$ in case of no rating. In other words, $K = C \cup \varnothing$. A particular rating concept is referred to as $c \in C = \{1, \ldots, k-1\}$ with $k = |K|$ number of rating concepts. Generally, a value space is classified to one of the following four groups:

- *Nominal rating*: The task of item recommendation can be treated as a classification problem that associates an item with one ore more rating classes. Popular classes of rating concepts C are $\{relevant, irrelevant\}$ or $\{likes, does\ not\ likes\}$.

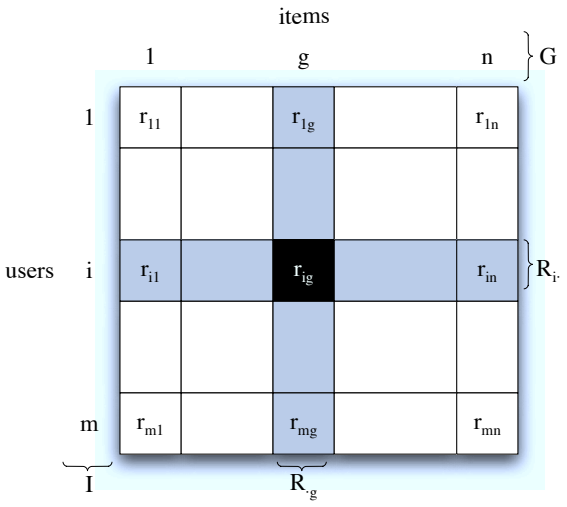

Fig. 1. Representation of the users's ratings for all items as rating matrix R

- *Ordinal rating*: The rating concepts are interrelated and can be ordered. The typical example for ordinal rating concepts is the star-rating on a 1-5 integer scale: $\{\bigstar, \ldots, \bigstar\bigstar\bigstar\bigstar\bigstar\}$. With ordinal ratings only the assertion can be done that a 4-star rated item is better then a 2-star rated item, but not that it is two times better.
- *Interval rating*: Items can be rated with a numeric value from $\mathbb{R}$. Ratios on this scale are not meaningful. In general, such ratings can be normalized to a $[-1, 1]$ scale.
- *Ratio rating*: Items can be rated with a numeric value from $\mathbb{R}$. In general, such ratings can be normalized to a $[0, 1]$ scale.

The subset of items that user i has rated with $r \neq \varnothing$ is called the user's i rated item set $G_i \subseteq G$. The subset of users which have rated item g is called the item's g rated user set $I_g \subseteq I$. The user for which we compute the recommendations is denoted as the active user $a \in I$.

Based on the introduced notation, a recommender system is defined as a total function f which returns for the active user a the item rating vector $\widehat{R}_{a\cdot}$. The item rating vector $\widehat{R}_{a\cdot}$ provides the user a's rating r_{ag} for item $g \in G_a$ or a predicted rating $\widehat{r}_{ag}$ for item $g \in G \setminus G_a$. For the computation, the function f considers the active user a as well as all other users in U, all items in G, user a's item rating vector $R_{a\cdot}$ as well as the rating matrix R and the set of rating concepts C. Hence, a recommender system is conceptualized as follows:

$$\widehat{R}_{a\cdot} = f(a, U, G, R_{a\cdot}, R, C) \tag{2}$$

with $\widehat{R}_{a\cdot}$ as the predicted rating vector for the active user a.

4. User preference modeling

Generally, the true user preferences can be represented as the true item rating vector $R_{i\cdot}^{true}$ which contains for every item $g \in G$ the user i's true rating $r_{ig} \in C$:

$$R_{i\cdot}^{true} = \langle r_{i1}^{true}, \ldots, r_{in}^{true} \rangle \quad , \forall r_i^{true} \in C \tag{3}$$

In case that all true item ratings of a user are known, providing recommendation comes down to the trivial recommendations of the top rated items from the vector $R_{i\cdot}^{true}$) . Generally, a recommender system has only partial knowledge about the user i's true preferences $R_{i\cdot}^{true}$. The reasons for a user not providing all ratings can be:

- *Costs*: Temporal or monetary costs limit the amount of items that a user is able to consume and ultimately to rate
- *Usability*: The rating effort is too high because the item has to be found first (search costs) and then being rated with too much effort (usability)
- *Privacy*: The user may have an interest in not to publish some ratings.

Since the true item rating vector $R_{i\cdot}^{true}$ is only partially known, the representation of the user's preferences is adjusted to a more general way that is adaptable to machine learning. It is assumed that every user $i \in I$ is able to assign the proper rating concept $c \in C$ to every item $g \in G$ based on his preferences. Hence, the user i's preferences are defined as the mental rating function or rather utility function $u^i(g)$:

$$u^i(g) : g \mapsto c = r_{ig}^{true} \quad , \forall g \in G_i \tag{4}$$

Note that $u^i(g)$ is a total function. Since user i always associates the item g with the true rating $r_{ig}^{true} \in C$, the probability that the utility function $u^i(g)$ provides the true rating concept is always 1:

$$P\big(u^i(g) = c | r_{ig}^{true} = c\big) = 1 \quad , \forall g \in G_i \tag{5}$$

With the representation of the user's preferences as utility function $u^i(g)$ (see Eq. 4), a computer program is able to learn an approximation of $u^i(g)$ based on the user i's ratings $r_{i\cdot}$ and the item set G_i. A computer program is said to learn from past experience E if its performance in some tasks T is improved with more experience E [23]. the learning problem of learning the user's preferences needs to be well-defined. Learning the user's preferences is a well-defined learning problem. According to [23], the following three features need to be defined: the class of tasks, the measure of performance to be improved and the source of experience:

- Task T: For the user i, predict for the item $g \in G$ the proper rating concept $c \in C$ such that $c = r_{ig}^{true}$
- Performance measure P: A metric which represents the accuracy of the prediction (e.g, percentage of correct predictions of rating concepts $c = r_{ig}^{true}$ of items $g \in G_i$ for the user i).
- Experience E: The user i's item rating vector $R_{i\cdot}$ and the set of items G_i

The learner faces the problem to hypothesize the rating concepts $c \in C$ for the item $g \in G_i$. Hence, the learner has to find the hypothesis $h^i(g)$ from the hypothesis space H of all possible hypotheses which best predicts the rating concept c the user i associates with item g. For this purpose, the performance measure P is used to determine the best hypothesis h^i. This hypothesis $h^i(g)$ is the user i's preference model or rather user i hypothesized utility function. A perfect hypothesized utility function $h^i(g)$ for $g \in G_i$ is defined as:

$$h^i(g) = u^i(g) \quad , \forall g \in G_i \tag{6}$$

In general, $h^i(g)$ is an approximation of $u^i(g)$ such that the probability that $h^i(g) = u^i(g)$ is below 1. This is expressed as:

$$P\big(h^i(g) = u^i(g)\big) \leq 1 \tag{7}$$

Consequently, $h^i(g)$ approximates $u^i(g)$ by the error term $\varepsilon^i(g)$:

$$u^i(g) = h^i(g) + \varepsilon^i(g) \tag{8}$$

The performance of $h^i(g)$ depends on the one side on the hypothesis space H, which is based on human designer's choice, and on the other side on the amount of Experience E and the number of the user i's rating. In case of an adequate hypothesis space H, it can be argued that the error term $\varepsilon^i(g) \to 0$ because the performance of $h^i(g)$ improves with more experience E. Thus, it is feasible to conclude that $h^i(g)$ approximates $u^i(g)$:

$$\boxed{h^i(g) \approx u^i(g)} \tag{9}$$

Comparison of user preference models

Traditionally in collaborative filtering, the user i's preferences are represented as the user i's item rating vector $R_{i\cdot}$. The item rating vector $R_{i\cdot}$ approximates the user i's preferences $u^i(g)$ proportionally to the number of ratings $|G_i|$ because each rating is independent from each other. In other words, the approximation rate with every additional rating $r_{ig} \in C$ remains constantly $\frac{1}{m}$.

In contrast, the approximation rate of user i's preferences with machine learning is characterized by a sigmoid function. In other words, the approximation rate increases in the beginning, but towards the end, it converges to 0. The reason for this behavior is that with every additional rating $r_{ig} \in C$, additional information about the item g's properties are provided such that item properties get interrelated and more information is gained. Towards the end, additional ratings mostly reinforce the user preference model and do not contribute to the user preference models accuracy. Referring to Eq. 8, the prediction accuracy of the user preference model $h^i(g)$ is limited due to the limitations of the defined hypothesis space H' representing the user i's preferences. The hypothesis space needs to be to limited to reduce the search space, thus reducing computational complexity.

However, in combination with collaborative filtering, both types of user preference models (i.e., item rating vector and machine learning models) get interrelated to each other such that missing ratings can be predicted to complement a user i's preference model respectively user i's item rating vector $R_{i\cdot}$ such that the predicted item rating vector $\widehat{R}_{i\cdot}$ approximates the user i's preferences super-proportional.]

5. Hypothesized user preference similarity

This section introduces this paper's novel metrics for comparing user preferences. The general idea is to approximate the user's preferences with a learned hypothesis about his/her preferences instead of his/her

item rating vector containing all ratings the user provides.

The preference similarity between two users a and b is denoted as $sim(a,b)$. If user a and b rate the same item identically, then a and b share identical preferences. If user a and b rate the same item differently the similarity metric determines the preference similarity between both users.

5.1. Classification-based user preference similarity

Referring to Eq. 4, the preference similarity between two users a and b is equivalent to the similarity of both utility functions:

$$sim(a,b) \equiv sim\big(u^a(g), u^b(g)\big) \quad ,\forall g \in G \quad (10)$$

As neither $u^a(g)$ nor $u^b(g)$ are known, both are approximated with $h^a(g)$ and $h^b(g)$ based on Eq. 9. Consequently, the similarity of both utility functions $u^a(g)$ and $u^b(g)$ is approximated by the similarity of the hypothesized user preferences $h^a(g)$ and $h^b(g)$:

$$sim\big(u^a(g), u^b(g)\big) \approx sim\big(h^a(g), h^b(g)\big) \quad ,\forall g \in G \quad (11)$$

Based on Eq. 11, a probabilistic classification similarity in Section 5.1.1 and a Pearson correlation-based classification similarity in Section 5.1.2 are presented to define the similarity $sim\big(h^a(g), h^b(g)\big)$.

5.1.1. Probabilistic classification similarity

If the target concept $c \in C$ is a nominal class then the task of rating item g by user i is equivalent to classifying item g by user i. The *probabilistic classification similarity metric* is defined as the probability that two users a and b both classify item i identically respectively rate item i identically:

$$sim(a,b) = P\big(u^a(g) = u^b(g)\big) \quad (12)$$

The probabilistic classification similarity is defined on the interval $[0,1]$ with 0, no similar preferences, and 1, identical preferences. With Eq. 9, the user rating function is approximated with the hypothesized user preferences:

$$P\big(u^a(g) = u^b(g)\big) \approx P\big(h^a(g) = h^b(g)\big) \quad (13)$$

The probabilistic classification similarity of hypothesized user preferences has to be defined on an item set. In the following, three candidates for the appropriate item set are discussed. The first candidate is the *common rated item* set $G_a \cap G_b$ of both users. The common rated item set may not necessarily be a representative sample of both users' preferences. The reason is that it corresponds to the intersection of both users' preferences which may represent only partially each users' preferences. The second candidate is the set of all items. Typically, it is very large and thus inappropriate due to the computational effort involved. The third candidate is the set of *unified rated items* $G_a \cup G_b$ which either of both users has rated. This set represents both users entire preferences, thus countering the issue of partial preference representation of the set of common rated items. Furthermore the set of unified rated items needs less computational effort than the set of all items. For these reasons, the probabilistic classification similarity of hypothesized user preferences is defined for the *unified rated items* $G_a \cup G_b$:

$$\boxed{sim(a,b) = P\big(h^a(g) = h^b(g)\big) \quad ,\forall g \in G_a \cup G_b} \quad (14)$$

5.1.2. Correlation-based classification similarity

If the target concept $c \in C$ is either an interval rating or a ratio rating as defined in Section 3 then we can exploit the notion of correlation that is defined on both scales. Hence, the preference similarity between two users a and b is defined as the correlation between both utility functions $u^a(g)$ and $u^b(g)$:

$$sim(a,b) = corr\big((u^a(g), u^b(g)\big) \quad ,\forall g \in G \quad (15)$$

As proposed in [14], the Pearson correlation is used to measure the user preference similarity. The Pearson correlation is computed as the covariance of both utility functions $u^a(g)$ and $u^b(g)$ divided by the product of their standard deviations:

$$\rho_{u^a(g),u^b(g)} = \frac{cov\big(u^a(g), u^b(g)\big)}{\sigma_{u^a(g)}\sigma_{u^b(g)}} \quad (16)$$

Hence, the similarity of two users a and b is defined as the Pearson correlation between both user preferences:

$$sim(a,b) = \frac{cov\big(u^a(g), u^b(g)\big)}{\sigma_a \sigma_b} \quad (17)$$

In contrast to the probabilistic classification similarity in Section 5.1.1, the correlation-based classification similarity is defined on the interval $[-1, +1]$, with -1, contrary preferences, 0, no similar preferences, and 1, identical preferences.

With Eqs. 11 and 17, the user preference similarity is approximated by the similarity between the hypothesized user preferences such that the correlation-based similarity is defined as:

$$\boxed{sim(a,b) = \frac{cov\big(h^a(g), h^b(g)\big)}{\sigma_{h^a(g)}\sigma_{h^b(g)} \quad , \forall g \in G_a \cup G_b}} \tag{18}$$

5.2. Partial user preference similarity

Generally, users' preferences are divers and manyfold such that they prefer items with different properties over others. As a consequence, people share similar preferences for a limited number of items, or in other words they share partially similar preferences as it is shown in [3]. Current preference similarity metrics do not account for partially similar preferences among people. Hence, collaborative filtering approaches wrongly consider users as like-minded in some contexts. Thus, partial user preference similarity metrics are needed to retrieve the correct (or true) set of like-minded users within a specific context.

To account for partial preference similarity, a user's preferences are interpreted as a set of individual preferences. Referring to Eq. 4, the user's utility function $u(g)$ is therefore interpreted as a function with case distinctions, whereby each case refers to a single partial preference $u_v(g)$:

$$u(g) = \begin{cases} u_1(g) & \text{if } g \text{ satisfies case } 1 \\ \quad\vdots & \\ u_v(g) & \text{if } g \text{ satisfies case } v \end{cases} \tag{19}$$

Note that all cases describe disjoint item sets. Each case is considered as a user's partial preference that constitute the user's entire preferences. Considering two users a and b, some items are evaluated to be a single case by $u^a(g)$ by user a whereas the same items are evaluated to be distinct cases by by user b in $u^b(g)$. For that reason, partial user preference similarity is defined as a directional preference similarity and is defined as:

$$sim(a, b|g) = sim\big(u_s^a(g), u^b(g)\big) \quad , \forall g \text{ satisfying case} D_s^a \tag{20}$$

where D_s^a is satisfied for the items g where $u_s^a(g)$ is applicable. I.e., D_s^a specifies the condition under which $u_s^a(g)$ gets chosen over other $u_:^a(g)$.

The partial preference similarity is defined as the probability that the user b rates item g identically to user a and is expressed as:

$$sim\big(u_s^a(g), u^b(g)\big) = P\big(u_s^a(g) = u^b(g)\big) \quad , \forall g \text{ satisfying case} D_s^a \tag{21}$$

Putting Eq. 9, a user's preferences or rather utility function $u(g)$ is approximated with the hypothesized user preferences or rather hypothesize utility function $h(g)$. Consequently, $h(g)$ is also interpreted as a function with case distinctions whereby each case refers to a single preference $h_v(g)$. A single preference is realized as a preference rule and consists of a premise D_v and its conclusion c_k with the premise being a conjunction of constraints of the set of features [23]. Ultimately, a case corresponds to the proper premise respectively conjunction of constraints. Thus, $h(g)$ is defined as:

$$h(g) = \begin{cases} h_1(g) & \text{if } g \text{ satisfies premise } D_1 \\ \quad\vdots & \\ h_v(g) & \text{if } g \text{ satisfies premise } D_v \end{cases} \tag{22}$$

An example of a hypothesized partial user preference is presented in Figure 2. The YOULIKE ontology provides the semantic concepts and properties to describe people's partial preferences. In the following the preprocessing step is described to specify a user's partial preferences.

Referring to Eq. (22), the product's properties determine which case is satisfied, thus determining the proper hypothesized partial preference. In this paper, a case is interpreted as a conjunction of constraints on the item's properties and the rating as the conclusion. The conjunction of constraints is represented as a set of constraints whereby each case has to be fulfilled.

In the following, the preprocessing step is introduced to extract hypothesized partial preferences from

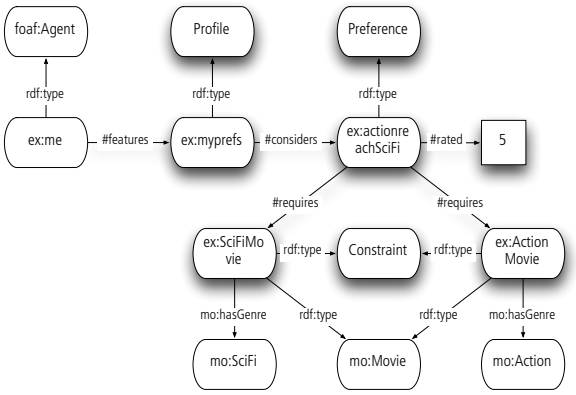

Fig. 2. Example of a user's partial preference using the YOULIKE ontology by [6].

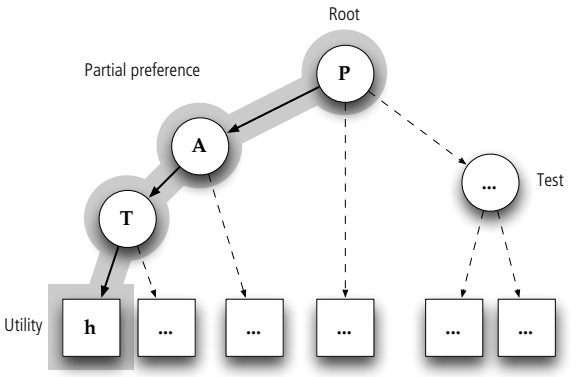

Fig. 3. Partial preferences encoded as branches from the root to the leaf of the decision tree.

a decision tree learner. A decision tree learner represents a hypothesis as a decision tree in which each node corresponds to a test of some property of a product. Each edge corresponds to a possible evaluation of such a test. A test in combination with an evaluation specifies a condition. Thus, each branch which starts at the root node and ends at a leaf corresponds to a conjunction of constraints with the leaf as the conclusion. Therefore, a hypothesized partial preference is encoded as a branch in a decision tree which starts at the root and ends in some leaf.

Figure 3 presents an exemplified decision tree representation of a users's preferences. The highlighted path in Figure 3 corresponds to a hypothesized partial preference and consists of the evaluated tests $\{P, A, T\}$ and its conclusion h.

To extract all hypothesized partial preferences, all all possible branches have to be parsed and extracted. In total, the number of hypothesized partial preferences corresponds to the number of leafs of the decision tree.

With Eq. 20, partial preference similarity between user a and b for the single preference $u_s^a(g)$ is approximated as:

$$sim\big(u_s^a(g), u^b(g)\big) \approx sim(h_s^a(g), h^b(g)) \qquad (23)$$

Hence, the partial preference similarity of hypothesized user preferences for $g \in G_a \cup G_b$ and g satisfying D_s^a is defined as:

$$\boxed{sim\big(a, b|g\big) \approx P\big(h_s^a(g) = h^b(g)\big)} \qquad (24)$$

Every machine learning algorithm whose model is representable as case distinctions (e.g., decision tree learner) can be applied to hypothesize a user's partial preferences.

6. Evaluation

The performance of the classification-based user preference similarity metrics (Section 5.1) and the partial user preference similarity metric (Section 5.2) is empirically evaluated with a movie data set and compared to state-of-the-art collaborative filtering approaches. The cold-start behavior of the presented similarity metric is evaluated with different data sets of different rating sparseness degrees.

6.1. Dataset

The MovieLens data set is used that has been collected by the GroupLens Research Project at the University of Minnesota. The data set provides 100 000 ratings of 943 users about 1 682 movies. The ratings are discrete values on a 1-to-5 integer scale with a rating mean of 3.53, a standard deviation of 1.13 and a rating median of 4. Each user provides at least 20 ratings. The genres of the movies are considered exclusively, which are provided by the MovieLens data set. The data set provides 18 different genres. Every movie is related to at least 1 genre and at most 6 genres. The median number of genres per movie is 2.

Based on the original data set, 10 different data sets have been created with an increasing rating sparseness degree of 10%. Specifically, for each of the 10 datasets additional 10% have been randomly removed of the ratings a user provides. Figure 4 shows the rating

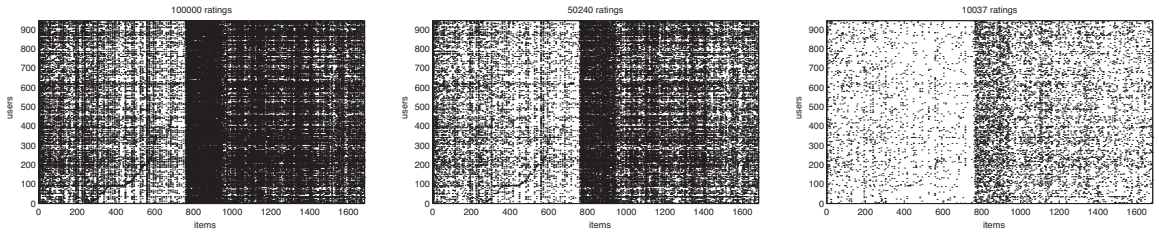

Fig. 4. MovieLens data set with increasing rating sparseness degree. (a) corresponds to the original MovieLens data set. In (b) and (c), 50% respectively 90% of the ratings are removed.

sparseness of the original data set and the two sparse data sets for which 50% and 90% of all ratings have been removed. Note that the 10th data set with a rating sparseness degree of 100% does not contain any ratings and thus is not considered in the evaluation. As is shown in Figure 4, the overall number of ratings decreases with each data set. However, the relative number of ratings per user and per item remains the same such that the rating distribution per user and item remains the same.

6.2. Experimental setting

6.2.1. Performance metrics

As suggested in [15], the *mean absolute error* (MAE) and *root mean square error* (RMSE) are used as performance metrics to measure the rating prediction accuracy. The MAE is the mean of the absolute difference between the user's rating r_i and the corresponding predicted rating $\widehat{r}_i$:

$$MAE = \frac{1}{n} * \sum_{i=1}^{n} |r_i - \widehat{r}_i|$$

Similarly, the RMSE is the root of the mean squared error. The RMSE complements the MAE because it is sensitive to large prediction errors rather then small prediction errors. The RMSE is computed as follows:

$$RMSE = \sqrt{\frac{1}{n} * \sum_{i=1}^{n} (r_i - \widehat{r}_i)^2}$$

The prediction errors are normally distributed. Therefore, the absolute prediction errors are folded normally distributed. For that reason, the non-parametric Wilcoxon signed-rank test for dependent samples is used to test the pairwise the MAE and RMSE of the recommendation approaches for significance. The significance level is set to $\alpha = 0.01$. The Bonferroni correction is applied to control the *family-wise error*.

The filtering effectiveness of relevant items is evaluated to complement to the evaluation of the prediction accuracy. The filtering effectiveness is measured by means of $Precision$ and $Recall$. The value of the ratings are 1-to-5 integer scale. Since a binary sale is required, an item is classified as relevant for user i, if he rated the item above the overall rating mean that is 3.53. Additionally, the F_1-$score$ is computed, which accounts for the well-known trade-off between $Precision$ and $Recall$. The accuracy of how well items are filtered is measured by the *area under the ROC curve* (AUC), which is equally to the probability that a randomly selected relevant item is higher ranked then a randomly selected not relevant item.

The $Precision$ corresponds to the relation of recommended relevant items and recommended items:

$$Precision = \frac{|rel.\ items \cap rec.\ items|}{|rec.\ items|}$$

The $Recall$ corresponds to the relation of recommended relevant items and all relevant items:

$$Recall = \frac{|rel.\ items \cap rec.\ items|}{|rel.\ items|}$$

The F_1-$score$ is the harmonic mean of the $Precision$ and $Recall$:

$$F_1\text{-}score = \frac{2 * Precision * Recall}{Precision + Recall}$$

The $Precision$ and $Recall$ values are based on a sequence of Bernoulli experiments and are thus binomial distributed. As stated by the *central limit theorem*, a binomial distribution approximates a normal distribution for large amount of Bernoulli experiments. Hence, paired-samples t-test is used to test for significance. The significance level is set to $\alpha = 0.01$ and Bonferroni correction is applied.

The experimental setup consisted of a 5-fold cross validation for each of the 10 data sets. For this purpose, the ratings for each user are divided into 5 folds of equal size. For each user, 4 folds are hold out which are used as the training set. The remaining fold is used as the test set. That results in a total of 50 experiments per evaluated approach.

6.2.2. Recommendation approaches for comparison

For the approaches presented in this paper the machine learning algorithms *J48*, *SMO*, and *Naïve Bayes* are chosen from the Weka library [30] to hypothesize user preferences and to evaluate the *probabilistic classification similarity* and the *correlation-based user preference similarity*. As suggested in Section 5.2, *J48* is used to hypothesize user preferences because each path from the root to the leaf in a decision tree is equivalent to a self-containing individual preference. These similarity metrics are applied to the general framework for k-nearest neighbor-based collaborative filtering [14] that is explained in Eq. 1. As evaluated in [14], $k = 10$ is used as the neighborhood size.

Since the approaches presented in this paper rely on machine learning algorithms, we evaluate the performance of approximating users' preferences, with the respective machine learning algorithms. Note that this corresponds a content filtering approach. The first is the decision tree learner *J48*, the second is the support vector machine *SMO* with a linear kernel, the third is *Naïve Bayes*, and the last is *Bayesian Network*. These four approaches are evaluated to to investigate how the performance of the user preference models correlated

As state-of-the-art collaborative filtering, *single value decomposition* (SVD) [28] and nearest neighbor-based collaborative filtering with Pearson correlation [14] (kNN Rating Pearson Corr.) are evaluated. Finally, the average item rating is used as the general baseline.

6.3. Results

6.3.1. Recommendation accuracy

Regarding to MAE and $RMSE$, collaborative filtering approaches (SVD, kNN Pearson corr., kNN J48 class., kNN J48 Pearson corr., kNN Naive Bayes class., kNN Naive Bayes Pearson corr., kNN SMO Pearson corr., and kNN Partial J48 class. sim.) perform significantly better than the content filtering approaches (J48, SMO, Naïve Bayes, and Bayes Net) in general as it is presented in Figure 5 and Table 1. For readability reason, Figure 5 presents a selection of the evaluated approaches that are presented in Table 1.

The kNN-based collaborative filtering approaches rely on the performance of the user preference similarity metric. The novel similarity metrics based on hypothesized user preferences (kNN J48 class. sim., kNN J48 Pearson corr., kNN Naïve Bayes class. sim., kNN Naïve Bayes Pearson corr., kNN SMO Pearson corr., kNN Partial J48 class. sim.) perform significantly better then the Pearson correlation of users' item rating vectors (kNN Pearson corr.) for data sets with high degree of sparseness. However, the performance of the hypothesized user preference similarity metrics are limited by the learning effect cap of the underlying machine learning algorithm such that these similarity metrics do not improve much with more data. Note that the correlation of the content filtering approaches and the kNN-based collaborative filtering approach based on hypothesized user preferences is a causal relationship.

In contrast, kNN Pearson correlation benefits more from much data such that it significantly outperforms all hypothesized user preference similarity metrics although the difference is marginally small especially compared with kNN Naïve Bayes Pearson corr.

Based on the evaluation results, it is concluded that Naïve Bayes and SMO perform best as the underlying machine learning algorithm for hypothesized user preferences similarity metrics. Both perform significantly better then hypothesized user preferences with J48 in general. Probabilistic classification performs significantly better then correlation-based classification similarity with J48. However, this does not hold for Naïve Bayes where the correlation-based classification similarity performs significantly better with sparse data but significantly worse with much data. However, it is shown that the probabilistic classification similarity reaches faster the learning effect cap whereas the correlation-based similarity benefits more from more data.

Though partial preference similarity metric with J48 performs significantly better then correlation-based classification similarity with J48, it performs significantly worse then the probabilistic classification similarity metric with J48. To conclude, partial preference

Table 1
Recommendation accuracy for different degree of rating sparseness.

Metric	Algorithm	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%
MAE	J48	0.881	0.883	0.890	0.896	0.910	0.910	0.920	0.929	0.954	1.014
	SMO	0.854	0.860	0.864	0.874	0.882	0.888	0.900	0.909	0.934	1.010
	Naïve Bayes	0.869	0.872	0.871	0.877	0.888	0.888	0.833	0.898	0.909	0.969
	Bayes Net	0.887	0.890	0.890	0.897	0.909	0.913	0.918	0.920	0.939	0.999
	Average	0.817	0.818	0.819	0.820	0.824	0.827	0.827	0.936	0.846	0.884
	SVD	0.817	0.879	0.819	0.820	0.825	0.829	0.828	0.839	0.849	0.898
	kNN Pearson corr.	**0.765**	0.770	0.879	0.776	0.782	0.793	0.802	0.822	0.874	0.907
	kNN J48 class. sim.	0.789	0.788	0.773	0.790	0.795	0.800	0.802	0.807	0.829	0.876
	kNN J48 Pearson corr.	0.803	0.805	0.790	0.807	0.816	0.824	0.830	0.839	0.859	0.900
	kNN Naïve Bayes class. sim.	0.775	0.775	0.808	0.776	**0.755**	0.788	**0.790**	**0.797**	**0.812**	**0.866**
	kNN Naïve Bayes Pearson corr.	0.766	**0.767**	0.777	**0.768**	0.775	**0.784**	0.792	0.800	0.820	0.884
	kNN SMO Pearson corr.	0.769	0.768	**0.770**	0.776	0.780	0.791	0.794	0.805	0.836	0.873
	kNN Partial J48 class. sim.	0.799	0.799	0.792	0.795	0.809	0.810	0.815	0.817	0.841	0.896
RMSE	J48	1.249	1.252	1.261	1.266	1.275	1.277	1.290	1.298	1.328	1.388
	SMO	1.213	1.221	1.227	1.236	1.241	1.248	1.262	1.272	1.301	1.375
	Naïve Bayes	1.233	1.237	1.235	1.239	1.249	1.247	1.255	1.261	1.274	1.336
	Bayes Net	1.254	1.256	1.257	1.262	1.272	1.273	1.283	1.285	1.304	1.366
	Average	1.025	1.026	1.028	1.029	1.034	1.038	1.040	1.051	1.067	1.119
	SVD	1.051	1.057	1.056	1.065	1.079	1.099	1.094	1.177	1.212	1.470
	kNN Pearson corr.	**0.977**	0.984	0.988	0.991	0.999	1.015	1.032	1.064	1.143	1.162
	kNN J48 class. sim.	1.001	1.002	1.005	1.004	1.011	1.018	1.021	1.028	1.062	1.135
	kNN J48 Pearson corr.	1.020	1.023	1.028	1.028	1.038	1.050	1.062	1.073	1.102	1.162
	kNN Naïve Bayes class. sim.	0.985	0.986	0.990	0.988	0.997	**1.002**	**1.007**	**1.015**	**1.039**	**1.113**
	kNN Naïve Bayes Pearson corr.	0.981	**0.983**	**0.986**	**0.986**	**0.994**	1.005	1.015	1.027	1.062	1.148
	kNN SMO Pearson corr.	0.984	0.985	0.992	0.995	1.001	1.014	1.015	1.035	1.079	1.144
	kNN Partial J48 class. sim.	1.023	1.025	1.024	1.024	1.037	1.041	1.047	1.054	1.086	1.166

similarity with J48 does not provide a significant improvement but it performs similar to the hypothesized user preference similarity with J48.

The only exception is SVD which performs significantly worse in terms of RMSE at a sparseness degree of 90%. In other words, SVD is vulnerable to poor recommendations when the sparseness degree is extremely high. This suggests to avoid SVD in *new-system cold-start* situations.

Within the content filtering approaches, the Naïve Bayes algorithm performs significantly best with high degree of sparseness. However, the learning effect of SMO is higher then the one of Naïve Bayes such that SMO hypothesizes a user's preferences significantly better then Naïve Bayes with lower degree of sparseness. Generally, the average rating significantly outperforms SVD even though the recommendation accuracy is similar.

In conclusion, retrieving like-minded individuals based on the comparison of hypothesized user preferences is the better method, especially in a cold-start situation. In the case when users provide many ratings, this approach is comparable to the traditional approach based on ratings for common rated items.

6.3.2. Comparison of collaborative filtering approaches

In the following, both collaborative filtering approaches, namely kNN Naïve Bayes Pearson correlation and kNN kNN Pearson corr., are compared. The first approach is chosen because it is the best performing approach that is based on the formal framework of Section 5.

The MAE and $RMSE$ measure the overall performance and thus, do not measure how well a recommender system performs in each of the three cold-start problems presented in Section 2.4. For that reason, the predicted recommendations are grouped according to the number of ratings a user provides (rating effort) and the number of times an item has been rated (item popularity). For this goal, the users and

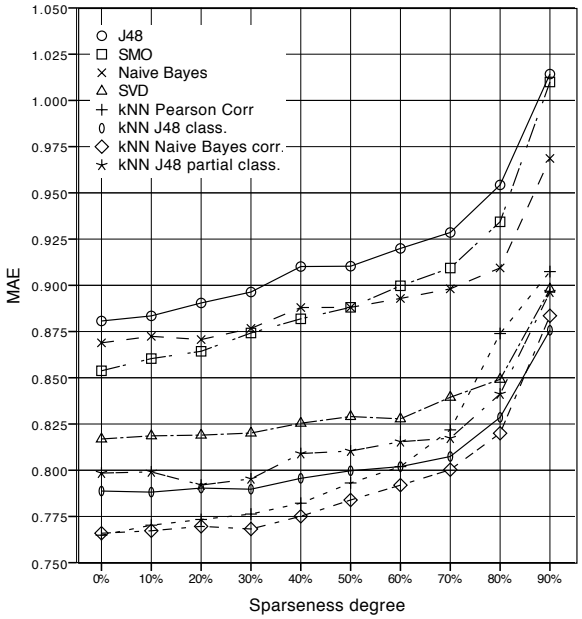

Fig. 5. Behavior of prediction accuracy of recommendations with increasing degree of rating sparseness from 0% (original data set) to 90%. The rating prediction accuracy is measured in terms of MAE and RMSE.

items have been binned to 20-quantiles. The behavior of kNN Naïve Bayes Pearson corr. is exemplified in Figure 6.

Referring to Figure 6, the highest and red colored peaks represent poor recommendation accuracy respectively high MAE. In contrast, lowest and blue colored sinks represent good recommendation accuracy respectively low MAE. The area with unpopular items and little rating effort of the user defines the the new-system cold-start area, the area with unpopular item defines the new-item cold-start area and the area with little rating effort of a user defines the new-user cold-start area.

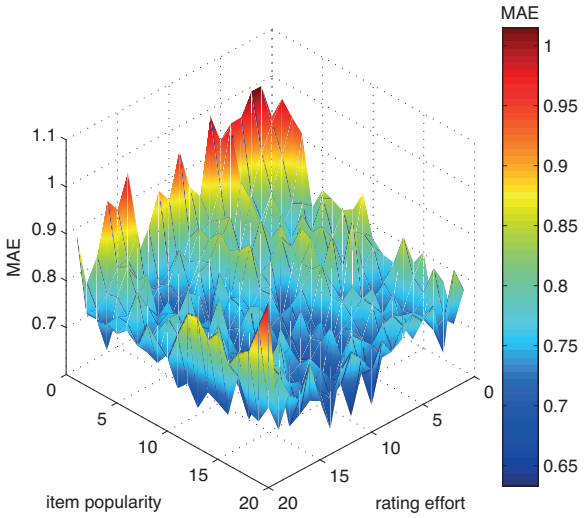

Fig. 6. Recommendation accuracy behavior of kNN Naïve Bayes Pearson corr. with respect to item popularity and a user's rating effort for the original MovieLens dataset (0% of sparseness degree).

As it is shown in Figure 6, collaborative filtering approaches perform poorly in the new-system cold-start area. Additionally, nearest neighbor-based collaborative filtering is shown to perform poorly in the new-item cold-start area independent of the number of ratings a user provides. The reason is that the number of potential neighbors with respect to a given item is very small, such that the k-nearest neighbors consists of less like-minded users. In contrast, the number of potential neighbors for popular items is much higher such that the k-nearest neighbors consists most likely of like-minded users. Interestingly, the number of ratings a user provides does not have that much impact on the overall recommendation accuracy because machine learning algorithms quickly approximate user preferences. To summarize, the item popularity has a bigger impact on the recommendation accuracy of nearest neighbor-based collaborative filtering then the number of ratings a user provides.

The recommendation accuracy of kNN-based collaborative filtering with Pearson correlation of item rating vectors (kNN Pearson corr.) is similarly distributed as in Figure 6. For that reason, the MAE dif-

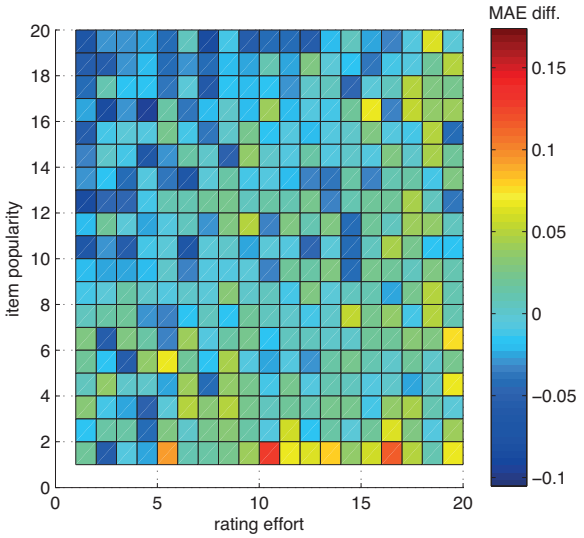

Fig. 7. Difference between kNN Naïve Bayes Pearson correlation (modeling hypothesized user preferences) and kNN Pearson correlation (using the item vector as user preferences) regarding to MAE. Numbers below 0 indicate a superiority of the hypothesized method.

ference of the kNN Naïve Bayes Pearson correlation and kNN Pearson Correlation is presented in Figure 7 to compare both. As the distribution of MAE differences shows, kNN Pearson correlation performs better than similarity metrics based on hypothesized user preferences when the rating effort of a user is high and the item popularity is low. In contrast, similarity metrics based on hypothesized user preferences perform comparably better for users providing few ratings and popular items. Therefore, the relative performance depends on item popularity and rating effort. The reason why kNN Pearson correlation benefits more from the rating effort is that the common rated item set is generally bigger and hence, the Pearson correlation of item rating vectors is more accurate. The reason why hypothesized user preference similarity benefits more from item popularity is that they do not stress preference similarity as much as Pearson correlation of item rating vectors.

To conclude, hypothesized user preferences model the user's preferences better then item rating vectors for sparse data and consequently perform significantly better. Especially Naïve Bayes approximates the user preferences best and performs significantly best as kNN Naïve Bayes Pearson correlation by means of MAE and $RMSE$ for mostly all sparseness degrees. Only for the sparseness degree of 0% (original MovieLens data set), the kNN Pearson correlation performs significantly better.

6.3.3. Filtering performance

The evaluation of the filtering performance is presented in Table 2 and confirms the evaluation results presented in Section 6.3.1 because the evaluated approaches behave similar.

Generally, content filtering approaches (J48, SMO, Naïve Bayes, and Bayes Net) have significantly lower $Precision$ then collaborative filtering approaches (kNN J48 class. sim., kNN J48 Pearson corr., kNN Naïve Bayes class. sim., kNN Naïve Bayes Pearson corr., kNN SMO Pearson corr., kNN Partial J48 class. sim.). In contrast, content filtering have partially higher $Recall$ then collaborative filtering approaches with the exception of the kNN SMO Pearson correlation. As the F_1-$score$ suggest, the correlation-based classification similarity with SMO performs best on every sparseness degree. Exceptionally, correlation-based classification similarity with Naïve Bayes provides the highest F_1-$score$ for the data set with a sparseness degree of 80%.

To complement the F_1-$score$ that combines $Precision$ and $Recall$, the AUC values are presented in Table 3. The hypothesized user preference similarity metrics with Naïve Bayes have a higher AUC value then others. The probabilistic classification similarity performs better in extreme cold-start settings with a high degree of sparseness whereas the correlation-based classification similarity approach performs better with much data.

7. Limitations, future work, and conclusions

This paper introduces the notion of using hypothesized user preferences—by the means of a machine learning algorithm—to model user preferences in the context of collaborative filtering. Second, it proposes to exploit the capabilities of some machine learning algorithms (such as decision trees) to partition the feature-space to model partial user preferences. Third, it introduces three novel similarity metrics—namely probabilistic classification similarity, correlation-based classification similarity, and partial similarity—operating on the hypothesized user preferences. These are then combined into hypothesized user preference based recommendation systems.

The paper then compares these systems to content based filtering approaches and state-of-the-art collab-

Table 2
Filtering performance of relevant items.

Metric	Algorithm	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%
Precision	J48	0.647	0.647	0.644	0.644	0.639	0.642	0.638	0.637	0.633	0.626
	SMO	0.648	0.648	0.648	0.645	0.645	0.645	0.640	0.641	0.638	0.627
	Naïve Bayes	0.648	0.647	0.648	0.645	0.643	0.642	0.639	0.639	0.635	0.628
	Bayes Net	0.649	0.648	0.649	0.645	0.643	0.641	0.638	0.641	0.635	0.625
	Average	0.691	0.691	0.689	0.689	0.686	0.686	0.686	0.683	0.679	0.647
	SVD	0.692	0.692	0.690	0.690	0.687	0.686	0.686	0.682	0.677	0.664
	kNN Pearson corr..	0.726	0.723	0.722	0.721	0.719	0.711	0.705	0.697	0.670	0.653
	kNN J48 class. sim.	0.717	0.719	0.716	0.717	0.714	0.712	0.710	0.708	0.698	0.684
	kNN J48 Pearson corr.	0.699	0.698	0.696	0.695	0.692	0.688	0.681	0.679	0.665	0.664
	kNN Naïve Bayes class. sim.	**0.727**	**0.727**	**0.727**	**0.727**	**0.722**	**0.720**	**0.716**	**0.719**	**0.708**	**0.691**
	kNN Naïve Bayes Pearson corr.	0.726	0.725	0.723	0.724	0.720	0.716	0.707	0.704	0.694	0.670
	kNN SMO Pearson corr.	0.720	0.721	0.714	0.714	0.714	0.708	0.703	0.695	0.679	0.671
	kNN Partial J48 class. sim.	0.712	0.709	0.714	0.714	0.710	0.707	0.705	0.702	0.695	0.673
Recall	J48	0.719	0.716	0.717	0.717	0.710	0.700	0.685	0.679	0.669	0.633
	SMO	**0.732**	0.729	0.725	0.719	0.717	0.709	0.687	0.683	0.677	0.635
	Naïve Bayes	0.731	**0.730**	**0.730**	0.725	0.725	**0.721**	0.706	0.710	**0.710**	0.678
	Bayes Net	0.718	0.716	0.717	0.708	0.708	0.703	0.687	0.692	0.688	0.662
	Average	0.696	0.694	0.699	0.701	0.699	0.689	0.689	0.688	0.675	0.688
	SVD	0.696	0.693	0.698	0.700	0.699	0.689	0.689	0.689	0.678	0.671
	kNN Pearson corr.	0.710	0.705	0.700	0.699	0.700	0.693	0.693	0.694	0.677	0.635
	kNN J48 class. sim.	0.677	0.680	0.678	0.681	0.677	0.676	0.668	0.673	0.670	0.663
	kNN J48 Pearson corr.	0.689	0.690	0.691	0.689	0.691	0.687	0.681	0.681	0.664	0.655
	kNN Naïve Bayes class. sim.	0.685	0.688	0.686	0.687	0.685	0.682	0.678	0.677	0.670	0.663
	kNN Naïve Bayes Pearson corr.	0.714	0.714	0.714	0.715	0.710	0.709	0.701	0.701	0.694	0.670
	kNN SMO Pearson corr.	0.729	**0.730**	0.724	**0.728**	**0.728**	**0.721**	**0.718**	**0.713**	0.700	**0.695**
	kNN Partial J48 class. sim.	0.688	0.683	0.696	0.689	0.680	0.683	0.676	0.687	0.672	0.660
F1-Measure	J48	0.681	0.680	0.679	0.678	0.673	0.670	0.661	0.657	0.650	0.629
	SMO	0.687	0.686	0.684	0.680	0.679	0.675	0.663	0.661	0.657	0.631
	Naïve Bayes	0.687	0.686	0.686	0.683	0.682	0.679	0.671	0.673	0.670	0.652
	Bayes Net	0.681	0.680	0.681	0.675	0.674	0.671	0.662	0.666	0.660	0.643
	Average	0.694	0.692	0.694	0.695	0.692	0.688	0.688	0.685	0.677	0.667
	SVD	0.694	0.693	0.694	0.695	0.693	0.688	0.688	0.685	0.678	0.668
	kNN Pearson corr.	0.718	0.714	0.711	0.710	0.710	0.702	0.699	0.695	0.673	0.644
	kNN J48 class. sim.	0.697	0.699	0.697	0.698	0.695	0.694	0.688	0.690	0.684	0.673
	kNN J48 Pearson corr.	0.694	0.694	0.693	0.692	0.692	0.688	0.681	0.680	0.665	0.659
	kNN Naïve Bayes class. sim.	0.705	0.707	0.706	0.706	0.703	0.701	0.697	0.698	0.688	0.677
	kNN Naïve Bayes Pearson corr.	0.720	0.720	**0.719**	0.720	0.715	0.713	0.704	0.702	**0.694**	0.670
	kNN SMO Pearson corr.	**0.724**	**0.726**	**0.719**	**0.721**	**0.721**	**0.714**	**0.710**	**0.704**	0.689	**0.683**
	kNN Partial J48 class. sim.	0.700	0.696	0.705	0.701	0.695	0.695	0.690	0.694	0.683	0.666

orative filtering approaches in a large experiment. It shows that the presented similarity metrics using hypothesized user preferences significantly outperform the compared collaborative filtering and content filtering approaches in cold-start situations with regards to prediction accuracy of recommendations and filtering performance. Furthermore, correlation-based classification similarity with Naïve Bayes as the user preference hypothesis learning algorithm performs similar to the best traditional collaborative filtering approach in non-sparse settings suggesting that it overall performs well and is the most robust against data sparseness.

Table 3
AUC for different sparseness degrees.

Algorithm	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%
J48	0.641	0.640	0.636	0.637	0.632	0.633	0.626	0.627	0.619	0.594
SMO	0.647	0.645	0.643	0.641	0.640	0.638	0.631	0.633	0.626	0.602
Naïve Bayes	0.647	0.645	0.645	0.644	0.640	0.639	0.634	0.636	0.628	0.609
Bayes Net	0.645	0.643	0.643	0.640	0.636	0.635	0.629	0.635	0.625	0.503
Average	0.710	0.711	0.709	0.709	0.706	0.703	0.702	0.697	0.686	0.655
SVD	0.711	0.711	0.709	0.710	0.705	0.702	0.702	0.697	0.688	0.657
kNN Pearson corr.	**0.755**	0.751	0.748	0.746	0.742	0.734	0.726	0.718	0.681	0.643
kNN J48 class. sim.	0.736	0.737	0.734	0.736	0.732	0.729	0.726	0.727	0.713	0.681
kNN J48 Pearson corr.	0.720	0.718	0.715	0.716	0.709	0.705	0.695	0.694	0.674	0.656
kNN Naïve Bayes class. sim.	0.747	0.747	0.746	0.747	0.741	0.739	**0.736**	**0.735**	**0.723**	**0.689**
kNN Naïve Bayes Pearson corr.	**0.755**	**0.754**	**0.752**	**0.752**	**0.748**	**0.742**	0.732	0.731	0.715	0.670
kNN SMO Pearson corr.	0.753	**0.754**	0.748	0.748	0.745	0.737	0.731	0.726	0.701	0.679
kNN Partial J48 class. sim.	0.732	0.729	0.737	0.733	0.724	0.725	0.720	0.721	0.707	0.669

Partial user preference similarity is shown to perform similarly as overall preference similarity with the same underlying machine learning algorithm.

It is shown that the recommendation performance of nearest neighbor-based collaborative filtering that retrieves like-minded users based on the the comparison of user preference models depends primarily on item popularity. The reason is that the set of users who rated an item defines the set of potentially like-minded people. The number of ratings a user provides, in contrast, is secondary because machine learning algorithms tend to take further ratings to verify and strengthen the current preference model.

Furthermore, nearest neighbor-based collaborative filtering approaches are only able to recommend items which have been at least rated ones. Indeed, the useful recommendations are limited to items that have been rated by like-minded users. Our approach, in contrast, does not depend on commonly rated items but on a similar hypothesis space, allowing the recommendation of items that have not been rated by like-minded users resulting in a bigger coverage.

A limitation of the presented similarity metrics is their dependency of the defined hypothesis space which limits the accuracy of the hypothesized user preferences. A trade-off exists between a large hypothesis space and hypothesized user preferences being a partial function and, thus, not able to classify all items. This limitation could be addressed by incorporating domain ontologies, which define semantical relationships among item properties to hypothesize user preferences. Here, general concepts of item properties are used to learn the preferences of users with few ratings and more specific item property concepts are used for users which provide many ratings using algorithms that can exploit such ontological relationships such as [7]. An additional source of ontological knowledge about items is the *Linked Data Cloud* that could be exploited analogously or even opportunistically to describe specific items in more detail or gather missing information.

Second, users' preferences are assumed to be stationary as well as users' rating being truthfully. This enables a valid computation of user preference similarity.

Third, the presented similarity metrics depend strongly on the set of rated items, which bias the comparison when a user provides much more ratings then another user. One approach to address this problem is to weight every rating of a user by the user's total number of ratings boosting the importance of ratings for users with fewer ratings.

Fourth, since a recommender system may not know all user ratings, a methodology needs to be developed to compare user preferences solely based on the syntactic representation and semantic meaning of the hypothesized user preferences.

Fifth, further research should investigate a methodology to select the proper recommendation approach for a given cold-start situation regarding to sparseness degree and the three types of cold-start problems.

Last but not least, as it is shown, partial user preference similarity performs similar to overall preference similarity. However, as discussed in [19], recommendation accuracy is only one quality aspect of a recommender system. Recommending diverse items may lead to higher user satisfaction, a better user experience, and satisfy a user's expectations. For that reason,

partial preference similarity needs further research regarding other quality aspects then simply recommendation accuracy.

In summary, the method presented here—the notion of using hypothesized user preferences by the means of a machine learning algorithm to model user preferences in the context of collaborative filtering—provides a highly promising avenue for improving the performance of recommendation systems and paves the way for the inclusion of background knowledge in the form of ontologies via the use of ontology-aware machine learning algorithms. In addition, it provides the foundations for recommender systems in the Semantic Web to incorporate collaborative filtering by interpreting preferences from machine learning models. As such, it provides an important building block for further improving automated recommendations both on the traditional, human and the semantic web.

References

[1] S. S. Anand, P. Kearney, and M. Shapcott. Generating semantically enriched user profiles for web personalization. *ACM Transactions on Internet Technology*, 7(4), 2007.

[2] M. Balabanovic and Y. Shoham. Fab: Content-based, collaborative recommendation. In *Communications of the ACM*, volume 40, pages 66–72, New York, NY, USA, March 1997. ACM.

[3] C. Basu, H. Hirsh, and W. Cohen. Recommendation as classification: Using social and content-based information in recommendation. In *In Proceedings of the 15th National Conference on Artificial Intelligence*, pages 714–720. AAAI Press, 1998.

[4] R. M. Bell and Y. Koren. Improved neighborhood-based collaborative filtering. In *The 13th International Conference on Knowledge Discovery and Data Mining in KDD Cup*, San Jose, California, August 2007.

[5] S. Berkovsky, T. Kuflik, and F. Ricci. Mediation of user models for enhanced personalization in recommender systems. In *User Modeling and User-Adapted Interaction*, volume 18, pages 245–286, August 2007.

[6] A. Bouza. *Hypothesis-Based Collaborative Filtering — Retrieving Like-Minded individuals Based on the Comparison of Hypothesized Preferences*. Phd thesis, University of Zurich, 2012.

[7] A. Bouza, G. Reif, A. Bernstein, and H. Gall. Semtree: Ontology-based decision tree algorithm for recommender systems. In *7th International Semantic Web Conference (ISWC)*, October 2008.

[8] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *14th Conference on Uncertainty in AI*, pages 43–52. Morgan Kaufmann, July 1998.

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

[10] I. Cantador, A. Bellogín, and P. Castells. A multilayer ontology-based hybrid recommendation model. *AI Communcations*, 21(2-3):202–210, 2008.

[11] I. Cantador and P. Castells. Building emergent social networks and group profiles by semantic user preference clustering. In *2nd International Workshop on Semantic Network Analysis (SNA 2006), at the 3rd European Semantic Web Conference (ESWC 2006)*, pages 40–53, Budva, Motenegro, June 2006.

[12] J. Golbeck and J. Hendler. Filmtrust: Movie recommendations using trust in web-based social networks. In *In Proceedings of the IEEE Consumer Communications and Networking Conference*, volume 1, pages 282–286, Las Vegas, NV, January 2006.

[13] N. Good, J. B. Schafer, J. A. Konstan, A. Borchers, B. Sarwar, J. Herlocker, and J. Riedl. Combining collaborative filtering with personal agents for better recommendations. In *AAAI '99/IAAI '99: Proceedings of the 16th national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence*, number 16, pages 439–446, Menlo Park, CA, USA, 1999. American Association for Artificial Intelligence.

[14] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *IGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 230–237, New York, NY, USA, 1999. ACM.

[15] J. L. Herlocker, J. A. Konstan, L. G. Reveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5–53, January 2004.

[16] H. Kautz, B. Selman, and M. Shah. Referral web: Combining social networks and collaborative filtering. *Communications of the ACM*, 40(3):63–65, March 1997.

[17] J. A. Konstan. Introduction to recommender systems: Algorithms and evaluation. *ACM Transactions on Information Systems*, 22(1):1–4, 2004.

[18] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76–80, January/February 2003.

[19] S. M. McNee, J. Riedl, and J. A. Konstan. Being accurate is not enough: How accuracy metrics have hurt recommender systems. In *CHI '06: CHI '06 extended abstracts on Human factors in computing systems*, pages 1097–1101, New York, NY, USA, 2006. ACM.

[20] P. Melville, R. J. Mooney, and R. Nagarajan. Content-boosted collaborative filtering for improved recommendations. In *18th national conference on Artificial intelligence*, pages 187–192. American Association for Artificial Intelligence, 2002.

[21] S. E. Middleton, H. Alani, and D. C. de Roure. Exploiting synergy between ontologies and recommender systems. In *In International Workshop on the Semantic Web, Proceedings of the 11th International World Wide Web Conference*, Hawaii, May 2002.

[22] S. E. Middleton, N. R. Shadbolt, and D. C. de Roure. Ontological user profiling in recommender systems. In *ACM Transactions on Information Systems*, volume 22, pages 54–88, January 2004.

[23] T. M. Mitchel. *Machine Learning*. McGraw Hill Higher Education, October 1997.

[24] M. Qin, S. Buffett, and M. W. Fleming. Predicting user preferences via similarity-based clustering. In *Proceedings of the Canadian Society for computational studies of intelligence, 21st conference on Advances in artificial intelligence*, pages 222–233, 2008.

[25] A. M. Rashid, S. K. Lam, G. Karypis, and J. Riedl. Clustknn: A highly scalable hybrid model- and memory-based cg algorithm. In *WEBKDD*, August 2006.

[26] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *CSCW '94: Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186, New York, NY, USA, 1994. ACM.

[27] P. Resnick and H. R. Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.

[28] B. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl. Applications of dimensionality reduction in recommender systems - - a case study. In *ACM WebKDD 2000 Web Mining for E-Commerce Workshop*, 2000.

[29] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260, Tampere, Finlance, 2002. ACM.

[30] I. H. Witten and E. Frank. *Data Mining - Practical Machine Learning Tools and Techniques*. Second edition. Elsevier, San Francisco, 2nd edition edition, 2005.

[31] B. Xu, J. Bu, C. Chen, and D. Cai. An exploration of improving collaborative recommender systems via user-item subgroups. In *Proceedings of the 21st international conference on World Wide Web*, pages 21–30, 2012.

[32] L. Zhang, B. Xiao, J. Guo, and C. Zhu. A scalable collaborative filtering algorithm based on localized preference. *International Conference on Machine Learning and Cybernetics*, 1:160–167, July 2008.

[33] C.-N. Ziegler. Semantic web recommender systems. In *In Proceedings of the Joint ICDE/EDBT Ph.D. Workshop 2004*, March 2004.

[34] C.-N. Ziegler and G. Lausen. Analyzing correlation between trust and user similarity in online communities. In *Proceedings of the 2nd International Conference on Trust Management*, volume 2995, pages 251–265, 2004.

[35] C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen. Improving recommendation lists through topic diversification. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 22–32, New York, NY, USA, 2005. ACM.