2013 IEEE 13th International Conference on Data Mining Workshops

Dissimilarity Features in Recommender Systems

Christos Zigkolis, Savvas Karagiannidis, Athena Vakali
Aristotle University of Thessaloniki
54124 Thessaloniki, Greece
chzigkol@csd.auth.gr, skaragiann@gmail.com, avakali@csd.auth.gr

***Abstract*—In the context of recommenders, providing suitable suggestions requires an effective content analysis where information for items, in the form of features, can play a significant role. Many recommenders suffer from the absence of indicative features capable of capturing precisely the users' preferences which constitutes a vital requirement for a successful recommendation technique. Aiming to overcome such limitations, we introduce a framework through which we extract dissimilarity features based on differences in preferences of items' attributes among users. We enrich the representations of items with the extracted features for the purpose of increasing the ability of a recommender to highlight the preferred items. In this direction, we incorporate the dissimilarity features into different types of classifiers/recommenders (C4.5 and *lib*-SVM) and evaluate their importance in terms of precision and relevance. Experimentation on real data (Yahoo! Music Social Network) indicates that the inclusion of the proposed features improves the classifiers' performance, and subsequently the provided recommendations.**

***Keywords*—Recommender Systems, Dissimilarity Features**

I. INTRODUCTION

Recommenders search in collections of items and suggest the most relevant ones to users [1]. The relevance of an item for a user can be merely expressed with a decision of whether the user likes it or not. Providing recommendations through such decisions can be approached from a machine learning perspective where classifiers are trained to offer effective solutions to this binary classification problem (i.e. *like/dislike*). Classifiers have played an active role in the field of recommenders. Probabilistic methods have been used in both content-based [2] and collaborative-filtering [3] recommenders as well as in hybrid solutions [4]. Furthermore, other types of classifiers such as Decision Trees [5] and Support Vector Machines [6] have also been utilized in recommendation methods.

A classifier's performance is basically affected by the discriminating power of the included features to precisely separate samples belonging to different classes. In a classifier that acts as a recommender, this discriminating power can be expressed by the ability of features to effectively capture users' preferences by including indicative information that leads to the preferred items. Therefore, the challenge is to present features capable of capturing different aspects of the analysed content. Solutions to this challenge answer to the *limited content analysis problem* [1] that many recommenders suffer from in cases where there is either a limited number of features or these features fail to be indicative enough.

In this paper, we offer a solution to the very same problem by introducing a framework through which we extract novel dissimilarity features for items' attributes (e.g. music artists, movie genres). The notion of dissimilarity is represented by the rationale that we focus on differences in preferences in such attributes expressed by members of user communities. In our case, each user is attached to a community defined by users who share similar preferences, and the dissimilarity features are based on information originating from members of such communities. The understanding of users' rating behaviour constitutes an essential requirement for a recommender to provide suitable suggestions to users. Therefore, we offer a new way of approaching users' preferences by enriching the representations of items with the extracted features which may lead us to valuable insights about them. The main contribution of this paper is summarized in the following points:

User communities through pairwise similarities
We propose two methods of calculating pairwise user similarities through which user communities are formed. The first method takes pairs of users that share common rated items and assesses a similarity value by utilizing preference information originating from these items, while the second method focuses on preferences of the attributes that two users share in common (i.e. common attributes included to their rated items). Common attributes are not necessarily found only in common rated items. It is likely for two users to share the former without sharing the latter (e.g. users rated different movies of the same actors). This justifies our decision to use both methods.

Dissimilarity-based feature extraction
Through the introduction of a novel metric, we take users' preferences into account along with the preferences originating from communities of similar users to extract the new dissimilarity features. For each attribute of an item one dissimilarity feature is extracted. Thus, apart from its set of already available attributes, an item is also characterized by a set of our features. Such enrichments in items' representations provide additional information that helps us deal with the limited content analysis problem.

Infusion to feature-based recommenders
The extracted features are not applicable only to specific recommenders, but they can be used by any method that works with item features in order to provide suggestions. This can be proved useful especially in content-based recommenders where additional information regarding items can be an asset.

We also present a classification framework where the dissimilarity features are incorporated into several types of recommenders/classifiers. We make use of a decision tree learner along with a SVM-based method. The former can be executed with both nominal and continuous features as opposed to the latter which cannot support the first type. Therefore, we convert the nominal values of items' attributes into continuous ones with the use of Naive Bayes [7]. Both classifiers utilize our features along with the converted values of items' attributes.

To ascertain the usability of our features we apply two evaluation processes. To begin with, we measure the performance of the classifiers in terms of precision to evaluate the contribution of the dissimilarity features in solving the binary classification problem. Furthermore, we apply feature selection techniques to evaluate all features regarding their relevance in the classification process. The experimentation has been carried out on data from Yahoo! Music social network service[1] which contains user ratings on songs.

The rest of the paper is structured as follows. Section II briefly discusses several related studies. In Section III, we present our feature extraction framework, while in Section IV we describe how the dissimilarity features are utilized in the classification process. Section V shows our experimental results and in Section VI the conclusions and future work are highlighted.

II. Related Work

Feature extraction can be achieved by attaching information from external resources and/or by analysing user ratings. In this section, we present a number of the most representative research efforts.

For the purpose of infusing exogenous information, semantic technologies have played a significant role. For instance, in [4] the authors create sense-based item representations by using WordNet ontology. They make use of synonyms to formulate matching criteria between items and user profiles that help them find the most appropriate suggestions. Moreover, content from external resources has also proved to be useful in enriching item representations. Katz et al. [8] use Wikipedia to enrich items' content for a more precise pairwise item similarities that lead to improved recommendations. Studies similar to the aforementioned ones are based on text semantics and/or on external content which, in our case, are not available. Our dissimilarity features are extracted by analysing only the user ratings.

In the direction of using ratings for feature extraction, Kim et al. [9] present a method called data-blurring that provides features which bring closer users who rated different items with the same attributes. This rationale resembles our user similarity method based on common rated attributes. However, in [9] binary types of preferences (i.e. preferred or unknown) are directly used on attributes, while we provide a more sophisticated approach where popularity and rating information are combined to calculate users' preferences in attributes. In another work, Symeonidis et al. [10] form feature-weighted user profiles with correlations between users and attributes. From these profiles similar users are discovered through which top-N lists of recommendations are extracted. In our work, user profiles contain preferences in attributes which can be seen as another type of user-attribute correlations, while we also make use of our profiles to form user neighbourhoods. However, in our case, information from such neighbourhoods does not directly result in recommendations, but it is used in the calculation of the dissimilarity features which affect the classifiers' performance and subsequently the final recommendations.

[1]Yahoo! R2 dataset in http://webscope.sandbox.yahoo.com

III. Dissimilarity Feature Extraction Framework

Aiming to precisely capture users' preferences, we introduce a feature extraction framework through which we calculate novel dissimilarity features for the attribute values of rated items. The nature of dissimilarity lies in the fact that a feature expresses the level of difference in preferences in an attribute value between a user who rated an item, to which this value belongs, and a group of similar users. The main idea behind this rationale can be shown by the following scenario.

Consider a band and a list of their songs which we are about to suggest to a user. Instead of taking solely into account the user's opinion on this band, we discover her similar users (i.e. regarding bands in general) and we make use of their opinions as well. Strong differences between these opinions may result in a different decision as opposed to the one we may have taken if we have only considered the opinion of the user in question. On the other hand, the credibility of the user's opinion may be enhanced in cases of consensus with other opinions. We argue that such differences in opinions, either strong or weak, offer indicative information regarding a user's preferences. Therefore, we quantify this information through the dissimilarity features.

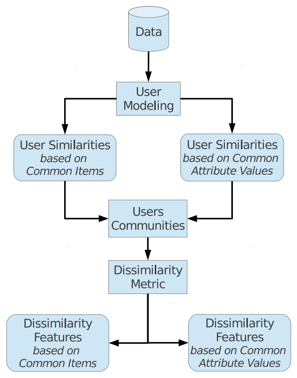

Data
User Modeling
User Similarities based on Common Items
User Similarities based on Common Attribute Values
Users Communities
Dissimilarity Metric
Dissimilarity Features based on Common Items
Dissimilarity Features based on Common Attribute Values

Fig. 1: Dissimilarity Feature Extraction Framework

As Figure 1 shows, we begin by formulating user profiles that maintain preferences in attribute values which represent the levels of interest a user shows in these values. These user profiles are utilized to calculate pairwise user similarities through which we get groups of top-k similar users (i.e. user communities) for each user. Here, we propose two methods of extracting similarities between users. Considering pairs of users, the first method takes into account the common rated items two users share, while the second focuses on the attribute values they both have expressed opinion (i.e. values belonged to their rated items). Then, we introduce a dissimilarity metric through which preference information coming from sets of similar users is utilized to calculate the dissimilarity features. Each user similarity extraction method results in a separate set of dissimilarity features. Each step is described in detail.

A. *Creating User Profiles*

We consider a set of items $I = \{item_1, item_2, ..., item_K\}$ along with their attributes $A = \{A_1, A_2, ..., A_L\}$. Each attribute is a finite set of attribute values, thus $A_i = \{a_1^i, a_2^i, ..., a_{l_i}^i\}$ where $i = [1, 2, ..., L]$ and l_i defines the number of values for the i-th attribute. Considering such values, we describe each item with a L-tuple which contains one value from each of the available attributes, thus $item_i = \{a_{x_1}^1, a_{x_2}^2, ..., a_{x_L}^L\}$ where $i = [1, 2, ..., K]$ and $a_{x_j}^j \in A_j$ with $1 \leq x_j \leq l_j$. Moreover, we maintain both a set of users $U = \{u_1, u_2, ..., u_N\}$ and their preferences $P = \{p_1, p_2, ..., p_M\}$. Each preference is a triplet with a rating of a user on an item, thus $p_i = \{u_a, item_b, r_i\}$ where $i = [1, 2, ..., M]$, $u_a \in U$, $item_b \in I$ and $r_i \in \Re$.

The presented user modeling facilitates the extraction of the dissimilarity features which, as mentioned earlier, are based on levels of interest in attribute values coming from groups of users. User profiles maintain these levels of interest, while they also contain other parts that are utilized in the process of forming these groups. In this paper, we extend the modeling process of Zigkolis et al. [11] work to meet the needs of calculating users' levels of interest in more than one attributes as well as supporting our user similarities extraction methods.

We begin by focusing on a single user's preferences in P represented by the subset $P(u_a) = \{\forall p_i \in P \mid p_i = \{u_a, item, r_i\}\}$ where $item \in I$. From these preferences, we extract a user's rated items given by $I(u_a) = \{\forall item \in I \mid p_i = \{u_a, item, r_i\}\}$ where $p_i \in P(u_a)$. This set is used in the user similarities extraction method presented in Section III-B1. Furthermore, we calculate the average rating of a user by the formula:

$$avg(u_a) = \frac{\sum_{i=1}^{|P(u_a)|}\{d(u_a) \times r_i\}}{|P(u_a)|} \tag{1}$$

$\forall p_i \in P(u_a)|p_i = \{u_a, item, r_i\}$. This average value will be utilized in our classification framework (see Section IV) as a threshold on ratings to label rated items.

Then, we filter $P(u_a)$ according to attribute values and we have $P(u_a, a_x^j) = \{\forall p_i \in P(u_a) \mid p_i = \{u_a, a_x^j \in item, r_i\}\}$ where $item \in I$ and $a_x^j \in A_j$. From such sets, we can take the values of an attribute included in u_a's items. Thus, $AV_{A_j}(u_a) = \{\forall a_x^j \in A_j \mid |P(u_a, a_x^j)| > 0\}$ which is used in our second user similarities extraction method presented in Section III-B2. Also, we calculate the average rating of a user in an attribute value $a_x^j \in A_j$ as follows:

$$avg(u_a, a_x^j) = \frac{\sum_{i=1}^{|P(u_a, a_x^j)|}\{d(u_a) \times r_i\}}{|P(u_a, a_x^j)|} \tag{2}$$

$\forall p_i \in P(u_a, a_x^j)|p_i = \{u_a, item, r_i\}$.

In (1) and (2), $d(u_a)$ defines a normalization factor that characterizes a user's level of pessimism (or optimism) [12] by comparing her ratings with all the ratings in data:

$$d(u_a) = \frac{\sum_{p_j \in P}\{r_j\}/|P|}{\sum_{p_i \in P(u_a)}\{r_i\}/|P(u_a)|} \tag{3}$$

$\forall p_j \in P|p_j = \{u, item, r_j\}$ and $\forall p_i \in P(u_a)|p_i = \{u_a, item, r_i\}$. Before utilizing a rating, we multiple it first with the normalization factor of the user who gave it. This eliminates the variance of ratings by bringing them closer to the total average rating in order to provide a common base for all users [13]. The average ratings in (1) and (2) are the normalized versions of the simple average ones.

Inspired by the work of Braak et al. [14], we capture users' interest in an attribute value with the combination of two factors, the popularity and the likeness. Popularity is determined by the percentage of items with a particular value in all user's rated items, while Likeness is extracted by user's average rating in an attribute value divided by the summarized average ratings in all values of an attribute:

$$Pop(u_a, a_x^j) = \frac{|P(u_a, a_x^j)|}{|P(u_a)|} \tag{4}$$

$$Lik(u_a, a_x^j) = \frac{avg(u_a, a_x^j)}{\sum_{i=1}^{l_j} avg(u_a, a_i^j)} \tag{5}$$

Contrary to the combination presented in [14] where both factors share the same importance, we favoured an f-measure equation in order to calculate the overall interest of a user in an attribute value. Therefore, we can put more emphasis on either popularity or likeness:

$$AI(u_a, a_x^j) = \frac{(1 + \beta^2) \times Lik(u_a, a_x^j) \times Pop(u_a, a_x^j)}{\beta^2 \times Lik(u_a, a_x^j) + Pop(u_a, a_x^j)} \tag{6}$$

where $\beta > 0$ parameter depicts to which factor we put more emphasis on (i.e. as β increases, popularity is emphasized against likeness). Through (6) it can be seen not only how well the attribute value is rated, but also how often items of that value are rated by the user. Its flexibility of favouring factors enables us to test which one leads to improved results. Finally, we proceed to a normalization step to calculate the normalized user interest in an attribute value:

$$nAI(u_a, a_x^j) = \frac{AI(u_a, a_x^j)}{\sum_{i=1}^{l_j} AI(u_a, a_i^j)} \tag{7}$$

B. Estimating User Similarities

As noted earlier, a dissimilarity feature is based on preference information originating from groups of users. Different user selection criteria lead to different groups which eventually affect the dissimilarity feature itself. One criterion could be to select all users that have rated items with an attribute value and use their preferences to extract the dissimilarity feature for this value. However, we consider this solution too generalized and prone to include noisy information. Hence, we suggest a more focused selection process by calculating pairwise user similarities through which we take only the top similar users into account.

A user similarity value is based on two factors. We initially create two vectors (i.e. one for each user) that contain their preference information regarding an attribute and then we calculate their correlation by using the Pearson formula. Also, we use the Sørensen index to measure the similarity over their sets of common elements (i.e. rated items or attribute values). This index acts as a boosting factor to the final similarity since it favours pairs that share more common elements compared with others. The generalized version of the similarity function is defined as:

$$Sim_{A_j}(u_a, u_b) = pC(\mathbf{v}_{u_a}^{A_j}, \mathbf{v}_{u_b}^{A_j}) \times SI(A, B) \tag{8}$$

where $A_j \in A$ and the Pearson Correlator is defined as follows:

$$pC(\mathbf{v}_{u_a}^{A_j}, \mathbf{v}_{u_b}^{A_j}) = \frac{\sum_{i=1}^{n}(\mathbf{v}_{u_a}^{A_j}(i) - \bar{\mathbf{v}}_{u_a}^{A_j})(\mathbf{v}_{u_b}^{A_j}(i) - \bar{\mathbf{v}}_{u_b}^{A_j})}{\sqrt{\sum_{i=1}^{n}(\mathbf{v}_{u_a}^{A_j}(i) - \bar{\mathbf{v}}_{u_a}^{A_j})^2}\sqrt{\sum_{i=1}^{n}(\mathbf{v}_{u_a}^{A_j}(i) - \bar{\mathbf{v}}_{u_a}^{A_j})^2}}$$

where $\mathbf{v}_{u_a}^{A_j}$ and $\mathbf{v}_{u_b}^{A_j}$ are the vectors for the two users created for the attribute A_j, while $\bar{\mathbf{v}}_{u_a}^{A_j}$ and $\bar{\mathbf{v}}_{u_b}^{A_j}$ are the averages of their values. In addition, the Sørensen index is represented by the formula:

$$SI(A, B) = \frac{2 \times |A \cap B|}{|A| + |B|}$$

where A and B represent either the users' rated items (i.e. $I(u_a)$ and $I(u_b)$) or the rated values of an attribute (i.e. $AV_{A_j}(u_a)$ and $AV_{A_j}(u_b)$). Note that, $Sim_{A_j}(u_a, u_b) \in [-1, 1]$ since $pC(\mathbf{v}_{u_a}^{A_j}, \mathbf{v}_{u_b}^{A_j}) \in [-1, 1]$ and $SI(A, B) \in [0, 1]$.

The proposed methods have different starting points. Focusing on common rated items is different than working with common rated attribute values. The first method guarantees that users who share a number of common rated items, they also share some common attribute values (i.e. the values belonged to the common items). However, as said earlier, there might be other common attribute values coming from different rated items. Note that, the presented methods extract similarities for each one of the available attributes which leads us to the formation of L symmetric matrices. This attribute-level approach seems more appropriate due to the fact that our dissimilarity features are also attribute-oriented. Both methods are described in detail and for the rest of the paper, we will refer to them as CI and CAV respectively.

1) Similarities based on common rated items: In CI method, a pairwise user similarity value is assessed by users' preference information derived from their common rated items. Considering the users u_a and u_b, the common items are given by the intersection of their rated items, thus we have $CI(u_a, u_b) = I(u_a) \cap I(u_b)$. We apply a threshold t_{CI} to the number of common items and we calculate similarity values only for users who share an adequate number (i.e. $|CI(u_a, u_b)| \geq t_{CI}$).

Considering an attribute $A_j \in A$, we start by formulating two vectors (i.e. $\mathbf{v}_{u_a}^{A_j}$ and $\mathbf{v}_{u_b}^{A_j}$) with users' normalized interest in the attribute values of A_j that belong to the items of $CI(u_a, u_b)$. In particular, for each common rated item, we take its attribute value $a_{x_i}^j$ and we update the two vectors with the values $nAI(u_a, a_{x_i}^j)$ and $nAI(u_b, a_{x_i}^j)$ respectively. Thus, we have $\mathbf{v}_{u_a}^{A_j}(i) = nAI(u_a, a_{x_i}^j)$ and $\mathbf{v}_{u_b}^{A_j}(i) = nAI(u_b, a_{x_i}^j)$ where $a_{x_i}^j$ is an attribute value of the i-th common rated item with $i = [1, 2, ..., |CI(u_a, u_b)|]$ and $1 \leq x_i \leq l_j$. Then, we calculate the users' similarity value for the attribute A_j by adjusting the (8) to meet the needs of this method:

$$SimCI_{A_j}(u_a, u_b) = pC(\mathbf{v}_{u_a}^{A_j}, \mathbf{v}_{u_b}^{A_j}) \times \frac{2 \times |CI(u_a, u_b)|}{|I(u_a)| + |I(u_b)|} \tag{9}$$

We apply (9) to all pairs of users for each attribute and from the populated L symmetric similarity matrices we extract L sets of top-k similar users for each available user. Thus, we have $topK_{A_j}^{CI}(u_a)$ for each $A_j \in A$. The algorithmic procedure is presented by Algorithm 1.

Algorithm 1 User Similarities Based On Common Items

```
Require: $U$, $A$, $nAI$ of attribute values for all users in $U$
Ensure: $SimCI_{A_j}$ user similarities matrix $\forall A_j \in A$
  for $a = 1 \rightarrow |U|$ do
    for $b = i + 1 \rightarrow |U|$ do
      if $|CI(u_a, u_b)| \geq t_{CI}$ then
        for all $A_j \in A$ do
          for $i = 1 \rightarrow |CI(u_a, u_b)|$ do
            $a_{x_i}^j \in item_i$ in $CI(u_a, u_b)$
            $\mathbf{v}_{u_a}^{A_j}(i) = nAI(u_a, a_{x_i}^j)$
            $\mathbf{v}_{u_b}^{A_j}(i) = nAI(u_b, a_{x_i}^j)$
          end for
          $SimCI_{A_j}(u_a, u_b) = pC(\mathbf{v}_{u_a}^{A_j}, \mathbf{v}_{u_b}^{A_j}) + \frac{2 \times |CI(u_a, u_b)|}{|I(u_a)| + |I(u_b)|}$
          $SimCI_{A_j}(u_b, u_a) = SimCI_{A_j}(u_a, u_b)$
        end for
      end if
    end for
  end for
```

The computational complexity for the extraction of the L similarity matrices is affected by the number of users as well as by the total average number of the common items they share. In particular, we have:

$$O(pairs_{tCI} \times (L \times \mu_{CI}) \times e_1 + e_2)$$

where $pairs_{tCI}$ defines the pairs of users that have passed the tCI threshold (i.e. $pairs_{tCI} < {}^{(|U|^2 - |U|)}/_2$) for each of which we create L vectors by looping through their common rated items. The average number of common items that all users share is represented by μ_{CI}. The e_1 cost represents the total time for the calculation of Pearson Correlation and for the final computation of a similarity value. Lastly, the e_2 cost defines the time we need to sort the similarity values and get the top-k similar users. The complexity of the latter is affected by the sorting algorithm one chooses to deploy. In our case, we make use of merge sort which is one of the most efficient sorting algorithms.

2) Similarities based on common attribute values: In CAV method, we focus on common rated attribute values between users to gauge a similarity value. Considering u_a and u_b, the set of the common rated attribute values is defined by the intersection of their values in each attribute $A_j \in A$, thus we have $CAV_{A_j}(u_a, u_b) = AV_{A_j}(u_a) \cap AV_{A_j}(u_b)$. Note that, in this method we have L sets of common elements as opposed to CI where we have only the set of common rated items.

Considering an attribute A_j, we calculate a similarity value only if users share an adequate number of common rated attribute values of it. Between two users, we find the one that has rated the fewer attribute values and we extract the percentage of the common values with respect to this minimum number. We apply a threshold t_{CAV} to this percentage:

$$t_{CAV} \leq \frac{|CAV_{A_j}(u_a,u_a)|}{min(|AV_{A_j}(u_a)|,|AV_{A_j}(u_b)|)}$$

Then, we create the vectors $\mathbf{v}_{u_a}^{A_j}$ and $\mathbf{v}_{u_b}^{A_j}$ by utilizing the users' normalized interest in their common values of A_j. We have $\mathbf{v}_{u_a}^{A_j}(i) = nAI(u_a, a_{x_i}^j)$ and $\mathbf{v}_{u_b}^{A_j}(i) = nAI(u_b, a_{x_i}^j)$ where $a_{x_i}^j$ is the i-th common attribute value belonged to $CAV_{A_j}(u_a, u_b)$ with $i = [1, 2, ..., |CAV_{A_j}(u_a, u_b)|]$ and $1 \leq x_i \leq l_j$. After the creation of the two vectors, we can calculate the users similarity value for the attribute A_j by adjusting the (8) to meet the needs of this method:

$$SimCAV_{A_j}(u_a, u_b) = pC(\mathbf{v}_{u_a}^{A_j}, \mathbf{v}_{u_b}^{A_j}) \times \frac{2 \times |CAV_{A_j}(u_a, u_b)|}{|AV_{A_j}(u_a)| + |AV_{A_j}(u_b)|} \tag{10}$$

From the extracted L symmetric similarity matrices, we once again keep L sets of top-k similar users for each available user. Thus, we have $topK_{A_j}^{CAV}(u_a)$ for each $A_j \in A$. We describe the whole algorithmic procedure in Algorithm 1.

Algorithm 2 User Similarities Based On Common Attribute Values

```
Require: $U$, $A$, $nAI$ of attribute values for all users in $U$
Ensure: $SimCAV_{A_j}$ user similarities matrix $\forall A_j \in A$
  for $a = 1 \rightarrow |U|$ do
    for $b = i + 1 \rightarrow |U|$ do
      for all $A_j \in A$ do
        if $(\frac{|CAV_{A_j}(u_a,u_b)|}{min(|AV_{A_j}(u_a)|,|AV_{A_j}(u_b)|)}) \geq t_{CAV}$ then
          for $i = 1 \rightarrow |CAV_{A_j}(u_a, u_b)|$ do
            $a_{x_i}^j \in CAV_{A_j}(u_a, u_b)$
            $\mathbf{v}_{u_a}^{A_j}(i) = nAI(u_a, a_{x_i}^j)$
            $\mathbf{v}_{u_b}^{A_j}(i) = nAI(u_b, a_{x_i}^j)$
          end for
          $SimCAV_{A_j}(u_a, u_b) = pC(\mathbf{v}_{u_a}^{A_j}, \mathbf{v}_{u_b}^{A_j})$
          $+ \frac{2 \times |CAV_{A_j}(u_a,u_b)|}{|AV_{A_j}(u_a)|+|AV_{A_j}(u_b)|}$
          $SimCAV_{A_j}(u_b, u_a) = SimCAV_{A_j}(u_a, u_b)$
        end if
      end for
    end for
  end for
```

The computational complexity of this method is affected by the number of users as well as the total average number of their common rated attribute values. In particular, we have:

$$O((pairs \times \mu_L) \times \mu_{CAV} \times e_1 + e_2)$$

where $pairs = {(|U|^2 - |U|)}/{2}$ is the pairs of users for each of which we check to see for which attributes the threshold t_{CAV} is passed (i.e. $\mu_L \leq L$). For these attributes we create vectors by looping through users' sets of common rated attribute values. The average number of these sets between all pairs of users is represented by μ_{CAV}. Once again, the e_1 cost measures the total time for the calculation of Pearson Correlation and for the final computation of a similarity value, while the e_2 cost defines the time the merge sort needs to sort the similarity values and extract the top-k similar users.

C. Estimating Dissimilarity Features

A dissimilarity feature is calculated for each attribute value of a rated item resulting in L new features for it. Considering an item with $a_x^j \in A_j$ being one of each attribute values, the dissimilarity feature of a_x^j represents the difference in the interest shown by the user u_a who gave the rating and her most similar ones regarding the A_j. To quantify this difference, we propose a dissimilarity metric where we make use of users' normalized interest in a_x^j:

$$D_{u_a}^S(a_x^j) = \frac{\sum_{u_b \in U'} w_{ab} \times (nAI(u_a, a_x^j) - nAI(u_b, a_x^j))}{\sum_{u_b \in U'} w_{ab}} \tag{11}$$

where S indicates either CI or CAV method. In case of CI, the set $U' = topK_{A_j}^{CI}(u_a)$ and $w_{ab} = SimCI_{A_j}(u_a, u_b)$, while in case of CAV, the set $U' = topK_{A_j}^{CAV}(u_a)$ and $w_{ab} = SimCAV_{A_j}(u_a, u_b)$.

The more similar two users are, the more important we consider their difference in preference. Therefore, we make use of user similarities as weights in (11) to support this rationale. According to (8) and (11), both similarity values and dissimilarity features share the same range (i.e. $[-1, 1]$). At the end, we update an item's representation by adding its extracted dissimilarity features. For the user u_a, we have $I'(u_a) = \{item'_1, ..., item'_{|I(u_a)|}\}$ where $item'_i = \{\{a_{x_1}^1, ..., a_{x_L}^L\}, \{D_{u_a}^S(a_{x_1}^1), ..., D_{u_a}^S(a_{x_L}^L)\}\}$ is the augmented version of the i-th item of $I(u_a)$.

IV. Classification Framework

In this section, we present a classification framework through which we train classifiers to solve the binary classification problem of deciding which items users like or not. We start by converting the nominal attribute values of items into continuous ones and then, we label the items by using rating information. After that, we train classifiers by using users' items as samples and we test their performance with various evaluation metrics (see Section V-C).

Community-based Naive Bayes Conversion
The applied SVM-based classifier cannot handle nominal features, thus, the problem of converting them from qualitative to quantitative arises. Several techniques have addressed this issue such as binary flag fields, text categorization, principal component analysis [15] and bayesian network classifiers. In this work, we adopt the latter by using Naive Bayes method [7].

For each user u_a, we take the similar users associated with all attributes. Thus, we define $topK_A^S(u_a) \uplus topK_{A_j}^S(u_a)$ for each $A_j \in A$ where S indicates CI or CAV method. Once this group of users has been formed, we gather all their samples in $I_A^S(u_a) \uplus I'(u_i)$ for each $u_i \in topK_A^S(u_a)$. Through $I_A^S(u_a)$ we formulate a sample-attribute matrix for u_a. Similarly to [7], we replace nominal attribute values with scores. However, instead of replacing all values of a sample with one score, we calculate a score for each of its values. In particular, from the user's sample-attribute matrix we take into account one attribute at a time through which posterior probabilities for its values are calculated. These probabilities lead us to the final scores. At the end, we update each item's representation by replacing the attributes with their corresponding scores. Thus, we have $I''(u_a) = \{item''_1, ..., item''_{|I(u_a)|}\}$ where $item''_i =$

$\{\{NB^S_{u_a}(a^1_{x_1}), ..., NB^S_{u_a}(a^L_{x_L})\}, \{D^S_{u_a}(a^1_{x_1}), ..., D^S_{u_a}(a^L_{x_L})\}\}$ is the updated version of the i-th sample of $I'(u_a)$.

This community-based approach of formulating a sample-attribute matrix for each user leads us to a matrix which captures more precisely the user's preferences which would not be the case if all users' samples are used. In the latter case, individual preferences would be lost since one score would be produced for each attribute value regardless the user's preferences. As a result, recommendation precision would be adversely affected. Note that, the samples of a user capture perfectly her preferences and they could be used to formulate the sample-attribute matrix. However, this approach is too biased and it is rejected.

Samples Labelling
A classifier needs labels on samples to be trained. Therefore, we label each sample of a user with one of our two classes, *like* or *dislike*, by comparing its normalized rating with the user's normalized total average rating of (1). The samples with ratings above the average are labelled with the *like* class whilst the others go to the *dislike* class. Therefore, we have:

$$sample_{label}(i) = \begin{cases} like & \text{if } d(u_a) \times r_i \geq avg(u_a) \\ dislike & \text{otherwise} \end{cases}$$

where $sample(i)$ is the i-th item of $I''(u_a)$ and r_i is u_a's rating on it.

Applying different thresholds to CI and CAV will result in different groups of similar users. This directly affects the dissimilarity features as well as the extracted scores. Hence, for each execution of these methods we obtain different samples for users which are used to train our classifiers. In particular, we formulate three different sets of features for each user's samples: the NB set which contains the scores of the attribute values, the DF set where the dissimilarity features of the updated item representations are included, and the NB+DF which is their combination. Considering a feature set, we train and test a classifier for each user as described in the Section V.

V. Experimentation

We carried out experiments on a real-world dataset to ascertain the usability of the proposed dissimilarity features in providing suitable recommendations through the use of classifiers. In this section, we describe the dataset and the classifiers we make use of, while we also evaluate our features in terms of discriminating ability.

A. Dataset Description

We make use of Yahoo! Music dataset which contains user ratings on songs accompanied by three types of attributes: genres, albums and artists. The dataset provides more than 700 million ratings, although we keep only a fragment of this vast amount of data. We omit the items that have missing attribute values and then, we maintain only the users that have rated more than 20 items. As a result, numbers of both items and users are greatly reduced (i.e. almost 92M ratings, 18K songs and 770K users). We also make use of a genre hierarchy to replace each genre with its connected one from the highest level. For instance, consider a song with a third-level genre "Southern Rock" connected with the first-level genre "Rock". In this case, we replace the former with the latter. Although with such replacements we overlook detailed genre information, we manage to decrease the sparsity of this attribute (i.e. more rated items share common genres). A similar approach cannot be applied to the other attributes due to lack of hierarchies.

We further proceed to a 5% random sampling on the remaining set of users which is adequate to evaluate the performance of our classification framework. From a statistical perspective, the error produced by our random sampling for a 99% confidence margin can be estimated by the formula $RSE = {}^{1.29}/\sqrt{|U|}$. In our case, the random sampling error rate is estimated to $\pm 0.6\%$ which is of no concern considering the results presented later. Table I contains the sizes of all entities we make use of.

$\lvert U\rvert$ users	$39,427$
$\lvert I\rvert$ items	$18,442$
$\lvert P\rvert$ preferences	$4,617,240$
$A_1 = genres$	$l_1 = 18$
$A_2 = albums$	$l_3 = 2,385$
$A_3 = artists$	$l_2 = 877$

TABLE I: Dataset Description

B. Classifiers as Recommenders

We evaluate our dissimilarity features by incorporating them into different recommenders/classifiers. Through Weka, we apply two representative classifiers from two widely used algorithmic families, the *C4.5* decision tree learner and the *lib*-SVM. Regarding the *C4.5* classifier, we make use of an unpruned decision tree with the default parameters of the Weka implementation. As for the *lib*-SVM, we make use of two types, the *nu*-SVC and the *C*-SVC, in order to manage two different groups of users. The *nu*-SVC is used for users that have a balanced number of like and dislike labelled items, while the *C*-SVC is used for the unbalanced ones. This separation is performed due to the inability of the former type to be applied to the group of the unbalanced users (i.e. nu cannot be estimated). In both types, a simple linear kernel is preferred with some changes in the default Weka parameter values[2].

C. Evaluation Metrics

To test the usability of our features we apply two evaluation processes. In the first process, we measure the precision of the applied classifiers through which we can assess the importance of our features in the classification tasks when they are used along with other features. Increased precision in such cases will indicate that the dissimilarity features can deal with our binary classification problem. In addition, we apply feature selection techniques which produce ordered lists of features regarding their relevance. In this case, the goal is for our features to be placed higher in these lists comparing to other features.

[2]Normalization and Probability Estimation parameters are changed to true value

C4.5 Classifier						
	CI method			CAV method		
	$t_{CI} = 10$	$t_{CI} = 20$	$t_{CI} = 30$	$t_{CAV} = 0.1$	$t_{CAV} = 0.2$	$t_{CAV} = 0.3$
NB	67.62%	66.28%	66.28%	72.65%	72.58%	72.10%
DF	76.66%	78.98%	79.37%	71.21%	71.51%	73.93%
NB+DF	77.81%	79.75%	**79.77%**	74.27%	74.39%	**75.77%**
lib-SVM Classifier						
NB	64.49%	62.87%	62.06%	69.12%	69.02%	68.57%
DF	69.50%	**69.73%**	67.62%	65.10%	65.27%	66.60%
NB+DF	68.02%	68.21%	69.69%	**70.09%**	70.07%	70.03%

TABLE II: Weighted Precision Results

1) Precision Results: A classifier's performance is based on class predictions for samples against their actual classes. The more correct predictions, the more effective a classifier is. Here, we make use of the precision metric where for each class we count the ratio between the positive predictions and the negative ones. Although recall metric is often used to evaluate the performance of classifiers, we exclude it from our experiments due to the fact that we do not focus on top-n recommendations. On the contrary, we predict labels for all samples which results in no false negatives for the classes. Therefore, recall metric is not relevant in the context of our experimentation.

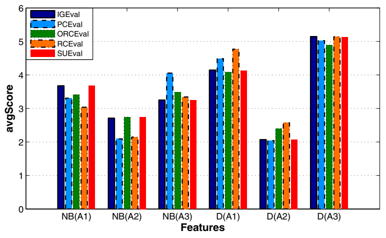
Focusing only on precision, we calculate a weighted precision for a classifier by combing the precisions of all classes the sizes of which are also taken into account. For each user we train and evaluate one classifier by using the 10-fold validation technique, and, at the end, we calculate the average weighted precision from all classifiers. Table II contains the results of the applied classifiers with respect to both CI or CAV methods executed with different t_{CI} and t_{CAV} thresholds. In all of our experiments, we define $\beta = 1$ for (7) and $k = 100$ for the top similar users.

As Table II depicts, the DF sets outperform the NB sets in almost all cases. Regarding the *C4.5* classifier, the improvement ranges from 1.83% to 13.09%, while for the SVM-based classifier we get from 5.01% to 6.86%. However, there are some cases where the NB sets produce better precision that the DF sets ranging from 1.07% to 4.02%. The latter does not indicate invaluable dissimilarity features and this can be further countered by the fact that the combined sets NB+DF produce improved results in all cases ranging from 0.97% to 13.49%. This endorses the reasoning that our features enhance the discriminating power of the applied classifiers. Furthermore, we observe that NB+DF sets produce better results when we increase the thresholds. This could be attributed to the fact that increased thresholds result in more suitable user communities regarding similarity, which propagates less "noisy" information in the calculation of the dissimilarity features.

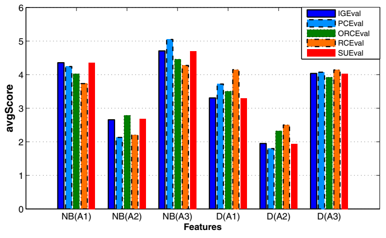
2) Ranking through Feature Selection: Apart from the evaluation based on precision, we also rank features according to five popular feature selection techniques[3] (i) information gain with respect to classes (IGEval), (ii) Pearson correlation between features and classes (PCEval), (iii) importance of features based on the One-Rule classification (ORCEval), (iv) using Relief criteria [16] (RCEval) and (v) features' symmetrical uncertainty with respect to classes (SUEval). Each technique analyses a user's samples and produces an ordered list of features where the places indicate the features' relevance regarding the classification. Scores are given to features between 1 and 6 where 1 indicates the lowest ranked feature and 6 the highest one. Considering all users and the ordered lists produced by a technique, we calculate an average score for each type of feature:

$$avgScore_C(F) = \frac{\sum_{u_a \in U} score_{u_a}^C(f)}{|U|} \in [1, 6]$$

where $score_{u_a}^C(f)$ is the score of criterion C given to the feature f belonged to u_a's NB+DF set. The set F contains 6 elements each one of which represents features from NB+DF sets of the same type. For instance, $NB(A_1)$ represents all converted values for A_1 attribute, while $D(A_1)$ is for all dissimilarity features of values of the same attribute.



(a) CI method with $t_{CI} = 30$



(b) CAV method with $t_{CAV} = 0.3$

Fig. 2: Features Ranking Points

Here, we present the average scores for the NB+DF features that produced the best precision results in our experiments. Figure 2a depicts the scores of features originating from the CI method with $t_{CI} = 30$, while Figure 2b shows the results when the CAV is applied with $t_{CAV} = 0.3$. Note that, the results for NB+DF features originating from both CI and CAV method executed with the other thresholds lead to similar conclusions, therefore they are not presented.

[3]All techniques are included in Weka software

Results indicate that our features perform adequately well in all evaluation criteria. In particular, when CI method is applied, $D(A_1)$ and $D(A_3)$ achieve the highest scores comparing to all others indicating that they play the most significant role in classifying users' samples. Regarding the CAV method, although the $NB(A_1)$ and $NB(A_3)$ get the best scores, the scores of $D(A_1)$ and $D(A_3)$ remain adequately high which infers that they also contribute in the classification tasks. Finally, both $NB(A_2)$ and $D(A_2)$ produce the lowest scores in all criteria which may lead to the conclusion that the attribute $A_2 = albums$ offers little to no additional discriminating power to the classifiers as opposed to the others due to its great sparsity (i.e. see Table I).

VI. Conclusions & Future Work

In this work, we introduce a dissimilarity feature extraction framework through which we augment items' representations with new features. When utilized in classification tasks, these enriched representations are proved to be effective in discovering preferred items, since they result in an improved performance in terms of precision. Presented results from the applied classifiers and the feature selection techniques endorse the usability of our features.

In future work, we are interested in comparing our features with state-of-the-art research efforts such as matrix factorization techniques [17]. Such comparison studies will greatly enhance the importance of our work. Furthermore, considering the computational complexity of our feature extraction framework, we plan to provide a distributed implementation through which its execution time will be reduced. Finally, the proposed framework could be adjusted in order to provide suggestions in the context of social networks. Considering a community of users, we could calculate dissimilarity features for the items they have expressed their opinion. Then, these features could lead to dissimilarities between users which can be used for their separation into two poles. The one pole would contain the users with small pairwise dissimilarity weights (i.e. like-minded users), while the other pole would include the ones with high dissimilarity weights (i.e. opposite-minded user). Applications that seek a balance between accuracy and diversity could use both poles of users.

VII. Acknowledgments

Christos Zigkolis's work has been co-financed by the European Union (European Social Fund ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) – Research Funding Program: Heracleitus II. Investing in knowledge society through the European Social Fund. Furthermore, we would like to acknowledge Yahoo! Labs for providing this research endeavour with the dataset of Yahoo! Movies R2 via Webscope.

References

[1] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, Eds., *Recommender Systems Handbook*. Springer, 2011.

[2] R. J. Mooney and L. Roy, "Content-based book recommending using learning for text categorization," in *Proceedings of the fifth ACM conference on Digital libraries*, ser. DL '00. New York, NY, USA: ACM, 2000, pp. 195–204. [Online]. Available: http://doi.acm.org/10.1145/336597.336662

[3] K. Miyahara and M. J. Pazzani, "Collaborative filtering with the simple bayesian classifier," in *Proceedings of the 6th Pacific Rim international conference on Artificial intelligence*, ser. PRICAI'00. Berlin, Heidelberg: Springer-Verlag, 2000, pp. 679–689. [Online]. Available: http://dl.acm.org/citation.cfm?id=1764967.1765055

[4] M. Degemmis, P. Lops, and G. Semeraro, "A content-collaborative recommender that exploits wordnet-based user profiles for neighborhood formation," *User Modeling and User-Adapted Interaction*, vol. 17, no. 3, pp. 217–255, Jul. 2007. [Online]. Available: http://dx.doi.org/10.1007/s11257-006-9023-4

[5] M. Pazzani, J. Muramatsu, and D. Billsus, "Syskill &#38; webert: Identifying interesting web sites," in *Proceedings of the thirteenth national conference on Artificial intelligence - Volume 1*, ser. AAAI'96. AAAI Press, 1996, pp. 54–61. [Online]. Available: http://dl.acm.org/citation.cfm?id=1892875.1892883

[6] C. Bomhardt, "Newsrec, a svm-driven personal recommendation system for news websites," in *Web Intelligence, 2004. WI 2004. Proceedings. IEEE/WIC/ACM International Conference on*, 2004, pp. 545–548.

[7] N. Lee and J.-M. Kim, "Conversion of categorical variables into numerical variables via bayesian network classifiers for binary classifications," *Comput. Stat. Data Anal.*, vol. 54, no. 5, pp. 1247–1265, May 2010. [Online]. Available: http://dx.doi.org/10.1016/j.csda.2009.11.003

[8] G. Katz, N. Ofek, B. Shapira, L. Rokach, and G. Shani, "Using wikipedia to boost collaborative filtering techniques," in *Proceedings of the fifth ACM conference on Recommender systems*, ser. RecSys '11. New York, NY, USA: ACM, 2011, pp. 285–288. [Online]. Available: http://doi.acm.org/10.1145/2043932.2043984

[9] H. Kim, J. Kim, and J. Herlocker, "Feature-based prediction of unknown preferences for nearest-neighbor collaborative filtering," in *Proceedings of the Fourth IEEE International Conference on Data Mining*, ser. ICDM '04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 435–438. [Online]. Available: http://dl.acm.org/citation.cfm?id=1032649.1033499

[10] P. Symeonidis, A. Nanopoulos, and Y. Manolopoulos, "Feature-weighted user model for recommender systems," in *Proceedings of the 11th international conference on User Modeling*, ser. UM '07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 97–106. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-73078-1_13

[11] C. Zigkolis, S. Karagiannidis, I. Koumarelas, and A. Vakali, "Integrating similarity and dissimilarity notions in recommenders," *Expert Systems with Applications*, vol. 40, no. 13, pp. 5132 – 5147, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417413001681

[12] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen, "The adaptive web," P. Brusilovsky, A. Kobsa, and W. Nejdl, Eds. Berlin, Heidelberg: Springer-Verlag, 2007, ch. Collaborative filtering recommender systems, pp. 291–324. [Online]. Available: http://dl.acm.org/citation.cfm?id=1768197.1768208

[13] D. Lemire and A. Maclachlan, "Slope one predictors for online rating-based collaborative filtering," *CoRR*, vol. abs/cs/0702144, 2007.

[14] P. t. Braak, N. Abdullah, and Y. Xu, "Improving the performance of collaborative filtering recommender systems through user profile clustering," in *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology - Volume 03*, ser. WI-IAT '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 147–150. [Online]. Available: http://dx.doi.org/10.1109/WI-IAT.2009.422

[15] J. de Leeuw and P. Mair, "Gifi methods for optimal scaling in r: The package homals," *Journal of Statistical Software*, vol. 31, no. 4, pp. 1–21, 8 2009. [Online]. Available: http://www.jstatsoft.org/v31/i04

[16] M. Robnik-Sikonja and I. Kononenko, "An adaptation of relief for attribute estimation in regression," in *Proceedings of the Fourteenth International Conference on Machine Learning*, ser. ICML '97. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, pp. 296–304. [Online]. Available: http://dl.acm.org/citation.cfm?id=645526.657141

[17] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, Aug. 2009. [Online]. Available: http://dx.doi.org/10.1109/MC.2009.263