Martijn Kagie *, Michiel van Wezel, Patrick J.F. Groenen

Econometric Institute, Erasmus University Rotterdam

A Graphical Shopping Interface based on Product Attributes

Econometric Institute Report EI 2007-02

Abstract

Most recommender systems present recommended products in lists to the user. By doing so, much information is lost about the mutual similarity between recommended products. We propose to represent the mutual similarities of the recommended products in a two dimensional space, where similar products are located close to each other and dissimilar products far apart. As a dissimilarity measure we use an adaptation of Gower's similarity coefficient based on the attributes of a product. Two recommender systems are developed that use this approach. The first, the graphical recommender system, uses a description given by the user in terms of product attributes of an ideal product. The second system, the graphical shopping interface, allows the user to navigate towards the product he wants. We show a prototype application of both systems to MP3-players.

Key words: Recommender Systems, Multidimensional Scaling, Similarity, Electronic Commerce, Case-Based Reasoning.

1 Introduction

In most electronic commerce stores, customers can choose from an enormous number of different products within a product category. Although one would assume that increased choice is better for customer satisfaction, the contrast is often the case [17,18]. This phenomenon is known as the paradox of choice: a large set of options to choose from makes it more difficult for the customer to find the product that she prefers most, that is, the product that is most similar to the customer's ideal product. When the amount of choice options

* Corresponding author. Phone: +31 10 4088951. Fax: +31 10 4089031.
Email addresses: kagie@few.eur.nl (Martijn Kagie), mvanwezel@few.eur.nl (Michiel van Wezel), groenen@few.eur.nl (Patrick J.F. Groenen).

increases, customers often end up choosing an option that is further away from the product they prefer most. Therefore, there is a need to help the customer to find this product and increase her satisfaction. Often, recommender systems are used to implement this process. These systems estimate user preferences and suggest products that match those preferences.

In many product categories, such as real estate and electronics, a consumer has to choose from a heterogenous range of products with a large amount of product attributes. Often, the customer has to make a selection based on a (limited) number of constraints on product attributes. The products that satisfy these constraints are usually shown in a list. A disadvantage of this approach is that customers can find these constraints too strict. In addition, product attributes can substitute each other, that is, a higher value on one attribute can compensate for a lower value on another. In this way, selection on pairs of attributes may not allow for attribute combinations that are preferred by a consumer. For example, a consumer who wants to buy an MP3 player can be equally satisfied with a cheaper MP3 player with less memory as with a more expensive MP3 player that has also more memory.

Another approach for creating a list of options to choose from, is to let the customer describe an ideal product based on her ideal values for the product attributes. Then, products are chosen such that they are most similar to the product.

A disadvantage of the usual approach of presenting the products in a list, which may be ordered on similarity to the ideal product in the second approach, is that no information is given on how similar the selected products are to each other. For example, two products that have almost the same similarity to the ideal product can differ from the ideal product on a completely different set of attributes and thus differ a lot from each other. Therefore, a recommender system should not only be based on the similarities of products to an ideal product, but also on the mutual similarities of the selected products.

In this paper, we propose a graphical recommender system (GRS) that visualizes the recommended products together with the ideal product in a two dimensional space using the mutual similarities. To do this, multidimensional scaling (MDS) [3] will be used. Since a consumer not always wants to specify her preferences, we also introduce a graphical shopping interface (GSI) that enables the customer to navigate through the products in a 2D space.

Similar graphical applications, so-called inspiration interfaces, are used in the field of industrial design engineering [11,19,20]. These applications are used to explore databases in an interactive way. At first, a small set of items is shown in a 2D space. Then, the user can click in any point in space and a new item that is closest to that point is added to the space. Our GSI differs from

these systems by recommending more items at a time and doing so using the similarities and not the distances in the 2D space.

The remainder of this paper is organized as follows. The next section gives a brief overview of the research in recommender systems. In Section 3, we give a description of the methodology used with an emphasis on the measure of similarity and MDS. In Section 4, the graphical recommender system is introduced and in Section 5 we extend the GRS to the graphical shopping interface, where the customer does not have to specify an ideal product, but can search through the product space in a graphical way. In Section 6, an application of both systems on MP3 Players is given, followed by an evaluation of our approach. Finally, we give conclusions and recommendations.

2 Recommender Systems

There is a wide literature on recommender systems. For an overview we refer to [1,15]. Shafer, Konstan, and Riedl [16] define recommender systems as: "[Systems that] are used by E-commerce sites to suggest products to their customers and to provide consumers with information to help them decide which products to purchase" [16, p. 116]. These suggestions can be the same for each customer, like top overall sellers on a site, but are often dependent on the user's preferences. These preferences can be derived in many ways, for example, using past purchases, navigation behavior, rating systems, or just by asking her preferences directly.

In this paper, we limit ourselves to the recommender systems where a personalized recommendation is given. Recent overview papers [1,15] make a distinction between three types of recommender systems based on how the recommendation is computed.

- *Content-based or knowledge based recommendation* [4] systems suggest products that are similar to the product(s) the customer liked in the past.
- *Collaborative filtering* [9] systems recommend products that other people with similar taste bought or liked in the past.
- *Hybrid approaches* [5] combine both content-based and collaborative methods.

Our systems belong to the category of content-based recommender systems. A large number of recommender systems in this group, including our approach, is based on case-based reasoning (CBR) [14]. The recommendation cycle used by case-based reasoning recommender system (CBR-RS) framework is shown in Figure 1.

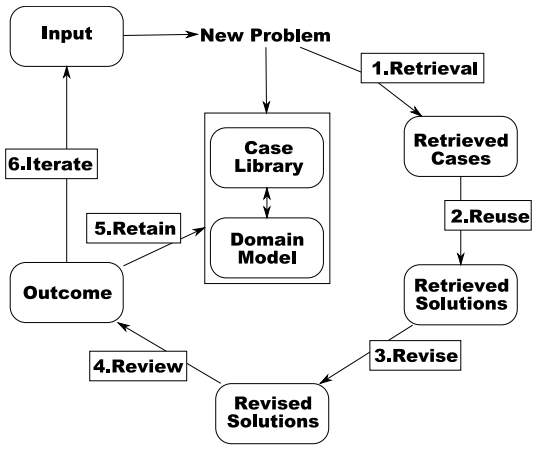

Figure 1. The steps in the CBR recommender system framework [14]. Not all steps have to be implemented by the CBR-RS.

The data used in this kind of systems is stored in the *case library*. All cases stored in this case library or case base have the same *domain model*, that is, the same feature space. The domain model consists of features describing at least one of the following submodels:

- *Content model:* Describes the product using product's attributes.
- *User model:* Describes the user by personal information like age, name, address, and past system usage.
- *Session model:* This model collects information about the recommendation session.
- *Evaluation model:* The evaluation model describes whether the recommendation was appropriate to the customer or not.

In many cases, including our approach, only the content model is part of the used domain model. The case library is merely the product catalog in this case.

In a CBR-RS, a recommendation is given based on the similarity between cases in the case library and the problem at hand. This problem is retrieved from the *input* of the customer, for example, an ideal product description or a sample product. Then, a CBR-RS implements some of the six steps described in [14] and shown in Figure 1. The two systems, the GRS and GSI, we introduce in this paper also belong to the CBR-RS family. Therefore, we use this framework in Sections 4 and 5 to describe our systems.

3 Methodology

An important part of the GSI is the similarity measure that is used to find cases that are recommended to the user. This similarity measure will be used for the selection of products and for visualizing the similarities between all of the recommended products. The method used for creating these 2D spaces is called multidimensional scaling (MDS) [3] and discussed in Subsection 3.1.

To define the measure of similarity between products, we introduce some notation. Consider a data set D, which contains products $\{\mathbf{x}_i\}_1^n$ having K attributes $\mathbf{x}_i = (x_{i1}, x_{i2} \dots x_{iK})$. In most applications, these attributes have mixed types, that is, the attributes can be numerical, binary, or categorical. The most often used (dis)similarity measures, like the Euclidean distance, Pearson's correlation coefficient, and Jaccard's similarity measure, are only suited to handle one of these attribute types.

One similarity measure that can cope with mixed attribute types is the general coefficient of similarity proposed by Gower [10]. Define the similarity s_{ij} between products i and j as the average of the nonmissing similarity scores s_{ijk} over the K attributes

$$s_{ij} = \sum_{k=1}^{K} m_{ik} m_{jk} s_{ijk} \Big/ \sum_{k=1}^{K} m_{ik} m_{jk} \ , \tag{1}$$

where m_{ik} is 0 when the value for attribute k is missing for product i and 1 when it is not missing.

The exact way of computing the similarity score s_{ijk} depends upon the type of attribute. However, Gower proposed that for all types it should have a score of 1 when the objects are completely identical on the attribute and a score of 0 when they are as different as possible. For numerical attributes s_{ijk} is based on the absolute distance divided by the range, that is,

$$s_{ijk}^N = 1 - \frac{|x_{ik} - x_{jk}|}{\max(\mathbf{x}_k) - \min(\mathbf{x}_k)}, \tag{2}$$

where $\mathbf{x}_k$ is a vector containing the values of the k^{th} attribute for all n products. For binary and categorical attributes the similarity score is defined as

$$s_{ijk}^C = 1(x_{ik} = x_{jk}), \tag{3}$$

implying that objects having the same category value get a similarity score of 1 and 0 otherwise.

To use Gower's coefficient of similarity in our system, three adaptation have to be made. First, the similarity has to be transformed to a dissimilarity, so that it can be used in combination with MDS. Second, we want to have the possibility to make some variables more important than others. Therefore, we need to incorporate weights into the coefficient. Third, the influence of categorical and binary attributes on the general coefficient turns out to be too large. The reason for this is that the similarity scores on binary or categorical attributes always have a score of 0 or 1 (that is, totally identical or totally different), whereas the similarity scores on numerical attributes almost always have a value between 0 and 1. Thus, the categorical attributes dominate the similarity measure. There is no reason to assume the categorical attributes are more important than numerical ones and we want to compensate for this. Therefore, we propose the following adaptations.

Both types of dissimilarity scores are normalized to have an average dissimilarity score of 1 between two different objects. Since the dissimilarity between the object and itself (δ_{ii}) is excluded and $\delta_{ij} = \delta_{ji}$, dissimilarities having $i \geq j$ are excluded from the sum without loss of generality. The numerical dissimilarity score becomes

$$\delta_{ijk}^{N} = \frac{|x_{ik} - x_{jk}|}{\left(\sum_{i<j} m_{ik} m_{jk}\right)^{-1} \sum_{i<j} m_{ik} m_{jk} |x_{ik} - x_{jk}|}. \tag{4}$$

The categorical dissimilarity score becomes

$$\delta_{ijk}^{C} = \frac{1(x_{ik} \neq x_{jk})}{\left(\sum_{i<j} m_{ik} m_{jk}\right)^{-1} \sum_{i<j} m_{ik} m_{jk} 1(x_{ik} \neq x_{jk})}. \tag{5}$$

Let C be the set of categorical attributes and N the set of numerical attributes. Then, the combined dissimilarity measure δ_{ij} is defined as

$$\delta_{ij} = \sqrt{\frac{\sum_{k \in C} w_k m_{ik} m_{jk} \delta_{ijk}^{C} + \sum_{k \in N} w_k m_{ik} m_{jk} \delta_{ijk}^{N}}{\sum_{k=1}^{K} w_k m_{ik} m_{jk}}}. \tag{6}$$

Here, a vector with weights $\mathbf{w}$ is incorporated to emphasize attributes differently. In (6), the overall square root is taken, since these dissimilarities can be perfectly represented in a high dimensional Euclidean space, when there are no missing values [10]. We use (6) as dissimilarity measure in the remainder of this paper.

3.1 Multidimensional Scaling

The dissimilarities discussed above are used in the GRS and GSI to create the 2D space where products are represented as points. A statistical technique for doing this, is multidimensional scaling (MDS) [3]. Its aim is to find a low dimensional Euclidean representation such that distances between pairs of points represent the dissimilarities as closely as possible. This objective can be formalized by minimizing the raw Stress function [12]

$$\sigma_r(\mathbf{Z}) = \sum_{i<j}(\delta_{ij} - d_{ij}(\mathbf{Z}))^2, \tag{7}$$

where the matrix $\mathbf{Z}$ is the $n \times 2$ coordinate matrix representing the n products in two dimensions, δ_{ij} is the dissimilarity between objects i and j forming the symmetric dissimilarity matrix $\mathbf{\Delta}$, and $d_{ij}(\mathbf{Z}) = \left(\sum_{s=1}^{2}(z_{is} - z_{js})^2\right)^{1/2}$ is the Euclidean distance between row points i and j.

To minimize $\sigma_r(\mathbf{Z})$, we use the SMACOF algorithm [6,7,8] based on majorization. One of the advantages of this method is that it is reasonable fast and that the iterations yield monotonically improved Stress values which is important to visualize the iterations to the user by a smooth dynamic GRS and GSI.

4 Graphical Recommender System

To discuss our graphical recommender system (GRS), we first give a specification of the GRS and then implement the system.

The *input* in the GRS is the ideal product described by the customer in terms of product attributes. The customer also can provide weights to the different attributes based on how important she finds them. With the user's input a *new problem* is constructed, that is, a sample case. In the *retrieval* phase, a set of cases from the *case library* that is most similar to the input is selected and these are directly *reused* as solutions and shown in the 2D space to the customer. When the results are not satisfying to the user, she can adapt her product description or the weights of the attributes to start the process again in the *iterate* step. The recommendation cycle of the GRS is visualized in Figure 2. Note that not all steps of the CBR-RS approach are implemented in our GRS, in particular, the *revise*, *review*, and *retain* step.

As mentioned before, the input of the GRS is the vector of product attributes of the ideal product, $\mathbf{x}^*$, and the weights of the attributes, $\mathbf{w}$. Further, we use

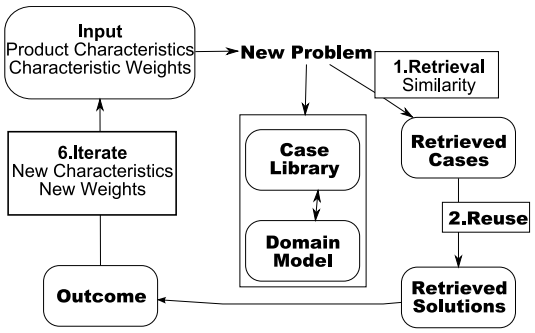

Figure 2. The CBR-RS steps implemented by the graphical recommender system.

the case library, that is, data set D. The implementation of the GRS is as follows.

We start by computing the weighted dissimilarities δ_{i*} between $\mathbf{x}^*$ and all the products $\mathbf{x}_i$ in data set D using the dissimilarity measure introduced in Section 3. This step leads to n values δ_{i*}.

Then, $p-1$ products are selected that are most similar to $\mathbf{x}^*$, by sorting the δ_{i*}'s and selecting the first p products. We combine $\mathbf{x}^*$ and the $p-1$ selected products in one data set D^*. We use (6) again to compute all the mutual dissimilarities of the selected objects and the ideal product and gather the dissimilarities in the symmetric matrix $\mathbf{\Delta}^*$ of size $p \times p$. This dissimilarity matrix $\mathbf{\Delta}^*$ is the input for the multidimensional scaling algorithm discussed in Section 3.1. The algorithm returns the $p \times 2$ coordinate matrix $\mathbf{Z}$ which is used to create the 2D space.

5 Graphical Shopping Interface

To facilitate selection of products by costumers who do not have a clear idea about what they are looking for, we propose the graphical shopping interface. The idea is to let the customer navigate through the complete product space, and each time a set of products is represented in a two dimensional space. The user can select a product and then a new set of products, including the selected product, is produced and visualized by MDS. First, a specification of this system is given and followed by its implementation.

The *input* of the GSI is a product selected by the user in the 2D space created in the previous iteration. Based on this previous selected product a *new problem* is constructed (that is, a sample case). In the *retrieval* phase, a large set of cases from the *case library* that is most similar to the input is selected and these are directly *reused* as solutions. In the *revise* stage, a smaller subset of products is chosen from the larger set specified in the previous steps. This

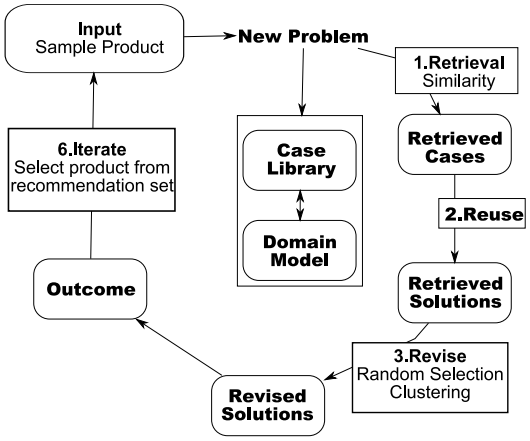

Figure 3. The CBR-RS steps implemented by the graphical shopping interface.

step can be implemented by, for example, random selection or clustering. This smaller set is shown in the 2D space. Then, the user can select the product that she likes most and this product will be the input for the next *iteration*. The recommendation cycle of the GRS is visualized in Figure 3. Also in this case the *review* and *retain* step are not implemented in the system.

The implementation of the GSI is not straightforward. The reason is that the revise step is not trivial and different implementations are possible. We analyze three different approaches: the random system, the clustering system, and the hierarchical system.

The random system uses random selection as its revise phase. The first iteration of the GSI is an initialization iteration. We refer to this iteration as iteration $t = 0$. The input of the user is unknown in this iteration, because the first input of user will be given after this iteration. Therefore, the product set D_t in this iteration will contain the complete case library, that is, $D_0 = D$. Then, p products are selected at random (without replacement) from D_0 and stored in the smaller set D_0^*. Using the dissimilarity metric proposed in Section 3, we compute the dissimilarity matrix $\mathbf{\Delta}_0^*$, given D_0^*. With the use of MDS we then create a 2D space $\mathbf{Z}_0$ containing these random selected products and show this to the customer.

The process starts when the customer selects one of the shown products. In every iteration, the selected product is treated as the new input $\mathbf{x}_t^*$. Then, we compute the dissimilarities between $\mathbf{x}_t^*$ and all other products in D. Based on these dissimilarities we create a set D_t with the $\max(p-1, \alpha^t n - 1)$ most similar products, where the parameter α with $0 < \alpha \leq 1$ determines how fast the data set selection is decreased each iteration. The smaller set D_t^* shown to the user, consists of product $\mathbf{x}_t^*$ and $p-1$ products that are randomly selected from the large product set. We again compute dissimilarity matrix $\mathbf{\Delta}_t^*$ and create the 2D space $\mathbf{Z}_t$ using MDS. The procedure terminates when D^* does

not change anymore. This happens when the size of D^* has decreased to p and the same product is chosen as was chosen in the last iteration.

When we set $\alpha = 1$, the system always returns a complete random selection at each stage and the user's input is almost completely ignored, that is, only the selected product is kept and $p - 1$ new random products are positioned in a new 2D space together with the kept product. When α is lower, we have more confidence in the selection of the user, but we also more quickly decrease the variance in D_t. The random system is summarized in Algorithm 1.

Algorithm 1 GSI implementation using random selection

procedure RANDOM_GSI(D, p, α)
  $D_0 = D$.
  Generate random $D_0^* \subset D_0$ with size p.
  Compute $\mathbf{\Delta}_0^*$ given D_0^* using (6).
  Compute $\mathbf{Z}_0$ given $\mathbf{\Delta}_0^*$ using MDS.
  $t = 0$.
  repeat
    $t = t + 1$.
    Select a product $\mathbf{x}_t^* \in D_{t-1}^*$.
    Get $D_t \subset D$ containing $\max(p - 1, \alpha^t n - 1)$ products most similar to $\mathbf{x}_t^*$ using (6).
    Generate random $D_t^* \subset D_t$ with size $p - 1$.
    $D_t^* = D_t^* \cup \mathbf{x}_t^*$.
    Compute $\mathbf{\Delta}_t^*$ given D_t^* using (6).
    Compute $\mathbf{Z}_t$ given $\mathbf{\Delta}_t^*$ using MDS.
  until $D_t^* = D_{t-1}^*$.
end procedure

A disadvantage of the random system is that is very hard to find a small group of products that are very different from all other products. There is only a small probability of selecting such a product in D_0^* and it is likely that this product is not in D_t the second time. For this reason, it can be advantageous to have the products selected in D_t^* represent the different groups of products in D_t. By decreasing the size of D_t each time these groups will become more similar to each other and finally become individual products.

The clustering system is quite similar to the random system, the only difference being that the random selection procedure is replaced by a clustering algorithm. In principle, every clustering algorithm can be used that can cluster a dissimilarity matrix. We use the average linkage method (see for example [21] or [13]) that is a hierarchical clustering method, yielding a complete tree (dendrogram) T of cluster solutions. Since this clustering method is based on a dissimilarity matrix, the dissimilarity matrix $\mathbf{\Delta}_t$ based on D_t is computed first using (6). Then, the average linkage algorithm is performed on $\mathbf{\Delta}_t$ resulting in dendrogram T_t. The system only uses the solution with p clusters D_t^c.

Each of the p clusters is then represented by one prototypical product in the product set D_t^*. For an easy navigation, the product selected in the previous iteration will always represent the cluster it belongs to. For the other clusters, we determine the product with the smallest total dissimilarity to the other products in the cluster. Define $\mathbf{\Delta}^c$ as the dissimilarity matrix (with elements δ_{ij}^c) between all products in cluster D^c, n_c as the size of this cluster, and i_c as the index of the prototypical product, we can define this ideal index as follows

$$i_c = \arg\min_i \sum_{j=1}^{n_c} \delta_{ij}^c. \tag{8}$$

The resulting product set D_t^* is used in the same way as in the random system to compute $\mathbf{\Delta}_t^*$ and $\mathbf{Z}_t$. The clustering system is summarized in Algorithm 2.

Algorithm 2 GSI implementation using clustering

procedure CLUSTERING_GSI(D, p, α)
  $D_0 = D$.
  Compute $\mathbf{\Delta}_0$ given D_0 using (6).
  Compute T_t given $\mathbf{\Delta}_0$ using average linkage.
  Find p clustering solution in T_0.
  Determine prototypical products of clusters using (8).
  Store prototypical products in D_0^*.
  Compute $\mathbf{\Delta}_0^*$ given D_0^* using (6).
  Compute $\mathbf{Z}_0$ given $\mathbf{\Delta}_0^*$ using MDS.
  $t = 0$.
  repeat
    $t = t + 1$.
    Select a product $\mathbf{x}_t^* \in D_{t-1}^*$.
    Get $D_t \subset D$ containing $\max(p-1, \alpha^t n - 1)$ products most similar to $\mathbf{x}_t^*$ using (6).
    Compute $\mathbf{\Delta}_t$ given D_t using (6).
    Compute T_t given $\mathbf{\Delta}_t$ using average linkage.
    Find p clustering solution in T_t.
    Determine prototypical products of clusters using (8).
    Store prototypical products in D_t^*.
    Compute $\mathbf{\Delta}_t^*$ given D_t^* using (6).
    Compute $\mathbf{Z}_t$ given $\mathbf{\Delta}_t^*$ using MDS.
  until $D_t^* = D_{t-1}^*$.
end procedure

Clustering (and especially hierarchical clustering) becomes quite slow as the product space gets larger. Since D_t is interactively selected each time the clustering has to be done each time as well. Therefore, our third implementation, the hierarchical system, does not create a set D_t each time, but uses only one hierarchical clustering result. We start with performing the average linkage algorithm to the complete case library D. To do this, we first compute dissimilarity matrix $\mathbf{\Delta}$ using (6). We will use the dendrogram T, created by

this clustering algorithm, to navigate through the product space. In the first iteration (the initialization), we start by setting $T_0 = T$. An iteration starts at the root of T_t. We will go down T_t until we find the p cluster solution. If this clustering does not exist, the largest possible clustering solution is chosen which is equal to the number of products in the previous selected cluster. This solution exists of p clusters D_t^c, where each of the p clusters is represented by one prototypical product in the product set D_t^*. The procedure for determining the prototypical products is the same as in the clustering system. Then, the dissimilarity matrix $\mathbf{\Delta}_t^*$ of D_t^* is computed using (6) and used as input for the MDS algorithm to compute the two dimensional representation $\mathbf{Z}_t$. When a product $\mathbf{x}_t^*$ is selected from this space, the cluster it represents is used as the root of T_{t+1}. The procedure is terminated when a selected cluster only contains a single product. This product is the final recommendation of the system.

Note that there is no convergence parameter α in this approach. Since a cluster is selected every step and products outside this cluster are not considered anymore in the remainder of the recommendation procedure, this approach converges quickly to a recommended product. As a consequence, it may lead to worse recommendations. The hierarchical system is summarized in Algorithm 3.

Algorithm 3 GSI implementation using hierarchical clustering.

procedure HIERARCHICAL_GSI(D, p)
  Compute $\mathbf{\Delta}$ given D using (6).
  Compute T given $\mathbf{\Delta}$ using average linkage.
  $T_0 = T$.
  $t = 0$.
  $n_c = size(D)$.
  repeat
    Find $\min(n^c, p)$ clustering solution in T_t.
    Determine prototypical products of clusters using (8).
    Store prototypical products in D_t^*.
    Compute $\mathbf{\Delta}_t^*$ given D_t^* using (6).
    Compute $\mathbf{Z}_t$ given $\mathbf{\Delta}_t^*$ using MDS.
    Select $\mathbf{x}_t^* \in D_t^*$ en determine cluster D_t^c it represents.
    $n_c = size(D_t^c)$.
    D_t^c is root of T_{t+1}.
    $t = t + 1$.
  until $n^c \leq p$.
end procedure

Table 1
Description of the MP3-player data set. The data set describes 321 MP3-players using 22 product attributes.

Categorical Characteristics	Missing	Levels (frequency)
Brand	0	Creative (53), iRiver (25), Samsung (25), Cowon (22), Sony (19), and 47 other brands (207)
Type	11	MP3 Player (254), Multimedia Player (31) USB key (25)
Memory Type	0	Integrated (231), Hard Disc (81), Compact Flash (8), Secure Digital (1)
Radio	9	Yes (170), No (139), Optional (3)
Audio Format	4	MP3 (257), ASF (28), AAC (11), Ogg Vorbis (9), ATRAC3 (5), and 4 other formats (6)
Interface	5	USB 2.0 (242), USB 1.0/1.1 (66), Firewire (6), Bluetooth (1), Parallel (1)
Power Supply	38	AAA x 1 (114), Lithium Ion (101), Lithium Polymeer (45), AA x 1 (17), AAA x 2 (4), Ni Mh (3)
Remote Control	9	No (289), In Cable (13), Wireless (10)
Color	281	White (7), Silver (5), Green (5), Orange (4), Purple (4), Red (4),Pink (4), Black (4),Blue (3)
Headphone	15	Earphone (290), Chain Earphone (8), Clip-on Earphone (2), Earphone With Belt (2), No Earphone (2), Minibelt Earphone (1), Collapsible Earphone (1)

Numerical Characteristics	Missing	Mean	Stand. Dev.
Memory Size (MB)	0	6272.10	13738.00
Screen Size (inch)	264	2.16	1.04
Screen Colors (bits)	0	2.78	5.10
Weight (grams)	66	83.88	84.45
Radio Presets	9	3.06	7.84
Battery Life (hours)	40	18.63	12.56
Signal-to-Noise Ratio (dB)	247	90.92	7.32
Equalizer Presets	0	2.60	2.22
Height (cm)	28	6.95	2.48
Width (cm)	28	5.57	2.82
Depth (cm)	28	2.18	4.29
Screen Resolution (pixels)	246	31415.00	46212.00

6 A Prototype Application to MP3 Players

In this section, we show a prototype of the graphical shopping interface on a data set containing MP3 players. This data set consists of 22 attributes of 321 MP3-players collected from the Dutch website `http://www.kelkoo.nl` during June 2006. The data set is of a mixed type, which means that we have both categorical and numerical attributes and contains quite some missing values. An overview of these data is given in Table 1.

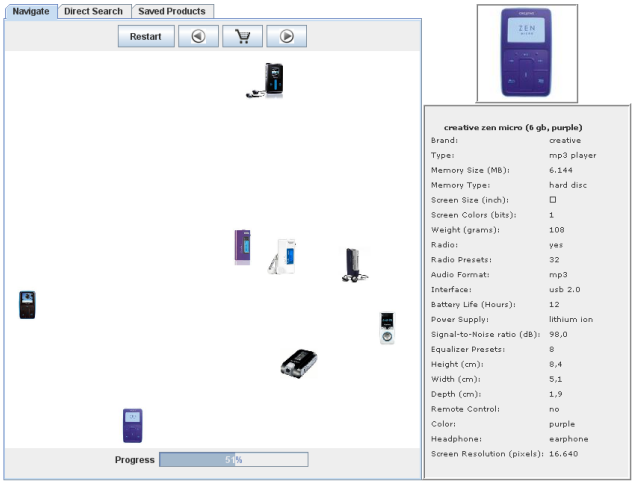

Figure 4. Screenshot of the graphical user interface of the prototype of the GSI for MP3-players

A prototype of our GUI is shown in Figure 4. A version of this prototype implementing the random GSI is available at `http://people.few.eur.nl/kagie/gsi.html`. The prototype is implemented as a Java Applet, which means that can be used in a web environment. The interface uses three tabs, each containing a 2D space and some buttons: The Navigate tab implementing the graphical shopping interface (GSI), the Direct Search tab implementing the graphical recommender system (GRS), and a Saved Products tab to save products in. In a 2D space, each product is shown as a point in the space and represented by a thumbnail picture. To add a selected product to the shopping basket the user presses the *Save* button represented by a shopping cart in the GSI or the GRS tab. The *Recommend* (or play) button uses the selected product for the next step in the GSI and by pressing the same button in the GRS tab recommendations are given based on the ideal values for the product attributes and weights specified by the user. The ideal product is represented by the symbol $\otimes$ in the GRS space. The products in the Saved Products space are also represented using MDS like in the two other spaces. With the *Delete* button saved products can be removed from this space. A panel at the right shows the product attributes of a selected product together with a picture of this product.

The transition between two steps in the GSI is implemented in a smooth way. After the selection of a product by the user, the new products are added to the space at random positions. Then, the space is optimized using MDS. This optimization is shown to the user. When the optimization has converged, the old products are gradually made less important (using a weighted version of MDS) until they have no influence anymore. Finally, the old products are removed and the space of new products is optimized. This implementation yields smooth visual transitions, which are important for an effective GUI.

Figure 5. An example of the GRS.

Figure 5 shows an example of a 2D space created by the GRS. The description of the ideal product is shown in Table 2. The MP3-players closest to the ideal product specification are the Samsung YH-820 and the Maxian MP2220 positioned above and below the ideal product respectively. It is maybe surprising that the distance between these two products is one of the largest in the space. However, when we have a closer look, we see that although both MP3-players are quite similar to our ideal product description, they are quite different from each other. The Samsung YH-820 is a small and light MP3 player with limited memory size and a smaller screen than we wanted. On the other hand, the Maxian has a large screen and memory size, but it is also larger and heavier than our ideal product. The Samsung YH-925 and the 20GB versions of the Cowon iAudio and the Samsung YH-J70 are all MP3-players having a memory of 20GB as we wanted, but having worse screens than the Maxian. Conversely, these players are somewhat smaller and lighter than the Maxian. The 30GB versions of the Cowon iAudio and the Samsung YH-J70 only differ in memory size from the 20GB versions. Therefore, these MP3-players are visualized somewhat further from the ideal product than the 20GB versions.

7 Evaluation of the Graphical Shopping Interface

To test our approach, the MP3 players data introduced in the previous section are used. We study the quality of the 2D spaces by considering the Stress values. Through a simulation study we evaluate how easily a customer can find the product she wants using the GSI.

Table 2
Description of ideal product used in Figure 5.

Characteristic	Value	Characteristic	Value
Brand	Samsung	Screen Size	1.8 inch
Type	Multimedia	Screen Colors	18 bits
Memory Type	Hard-disk	Weight	100 grams
Radio	No	Radio Presets	0
Audio Format	MP3	Battery Life	12 hours
Interface	USB2.0	Signal-to-Noise	95 dB
Power Supply	Lithium Ion	Equalizer Presets	5
Remote Control	No	Height	8 cm
Color	Black	Width	4 cm
Headphone	Earphone	Depth	1 cm
Memory Size	20 GB	Screen Resolution	60000 px

Representing recommended solutions in a 2D space only improves the system when the 2D representations are of a sufficient quality, that is, information is added to a regular list. Although this is largely a subjective matter, we can say something about the Stress of the obtained solutions. Solutions with a low Stress value represent, at least technically, the products in a good way. Since we also like to compare solutions with a different number of products, Stress is normalized by dividing by the sum of squared dissimilarities, that is,

$$\sigma_n = \frac{\sum_{i<j}(\delta_{ij} - d_{ij}(\mathbf{Z}))^2}{\sum_{i<j}\delta_{ij}^2}. \tag{9}$$

This normalized Stress can be interpreted as the proportion of the unexplained sum-of-squares of the dissimilarities [3].

To estimate the average quality of a fit in the GSI is practically impossible, because of the interactivity and randomness in the system. However, we can say something about the goodness of the representations in the GRS. Since the user has much freedom in specifying her ideal product, there is a very large number of possible plots in practice. We use a leave-one-out method to approximate the quality of the plots. Each time, we pick one product from the data set as the ideal product of the user and we use the other products as our case library. All attributes are used to compute the dissimilarities and all weights are set to 1. Then, the $p-1$ most similar products to this ideal product are selected and a 2D space of these p products is created using MDS. This procedure is repeated, until each product has functioned once as an ideal product description. This overall procedure is done for $p = 2$ until $p = 10$. The results for the average normalized Stress values are shown in Table 3.

It is no surprise that solutions with only two products have a Stress of (almost) zero, since there is only one dissimilarity in that case that can always be perfectly scaled on a line. Also the solutions with three points have an average

Table 3
Normalized Stress values for the experiments to determine the quality of the 2D representations in the GRS.

p	Mean Normalized Stress
2	$3.36 \cdot 10^{-32}$
3	$3.13 \cdot 10^{-4}$
4	$5.77 \cdot 10^{-3}$
5	$1.21 \cdot 10^{-2}$
6	$1.96 \cdot 10^{-2}$
7	$2.63 \cdot 10^{-2}$
8	$3.16 \cdot 10^{-2}$
9	$3.62 \cdot 10^{-2}$
10	$4.05 \cdot 10^{-2}$

Stress value very close to zero. When we increase p, the Stress values increase, as may expected because the problem gets more difficult. Since this increase seems almost linear, it is hard to identify the ideal product set size. If one wants near perfect solutions, p should be set to 3, but a space of three products is not very informative. For larger p, the Stress values are still acceptable, even for $p = 10$, 96% of the sum-of-squares in the dissimilarities is explained by the distances in the 2D space.

Apart from the quality of the 2D representations, the navigation was tested in the different implementations of the GSI. In an ideal system, the user will always find the product she likes most in a small number of steps. We expect that there will be a tradeoff between the number of steps that is necessary to find the product and the probability that the customer will find the product she likes.

To evaluate the navigation attributes of the different systems in a simulation, some assumptions need to be made about the navigation behavior of the user. First, we assume that the customer implicitly or explicitly can specify what her ideal product looks like in terms of its attributes. Second, we assume that the user compares products using the same dissimilarity measure as the system uses. Finally, it is assumed that in each step the customer chooses the product that is most similar to the ideal product of the customer. Note that we only evaluate the search algorithm in this way and not the complete interface, since the results would have been the same when the results were shown in a list.

We use a leave-one-out procedure to select the ideal product descriptions of the user. A random ideal product description is not used, since such a procedure will create a lot of ideal products that do not exist in reality. Each time, one product is selected as the ideal product of the customer and all other products are used as the case library. We repeat this until every product is left out once. We evaluate the three different implementations (the random, the clustering, and the hierarchical system) with p set to 4, 6, 8, and 10. For the random

Table 4
Results for different specifications of the random system.

p	α	Successes	In 5 Steps	Average number of Steps
4	0.2	16.8%	16.8%	5.02
	0.4	29.3%	19.3%	6.38
	0.6	41.7%	10.0%	9.12
	0.8	56.1%	5.9%	15.99
6	0.2	29.3%	28.7%	4.77
	0.4	42.7%	34.9%	5.83
	0.6	52.7%	17.1%	8.01
	0.8	70.1%	10.9%	13.12
8	0.2	43.0%	42.7%	4.34
	0.4	47.0%	43.6%	5.29
	0.6	66.4%	30.5%	6.96
	0.8	80.1%	16.2%	11.22
10	0.2	46.4%	46.4%	4.09
	0.4	58.9%	56.7%	4.82
	0.6	70.4%	37.1%	6.27
	0.8	83.8%	19.9%	9.92

and clustering system we also vary the parameter α by setting it to the values 0.2, 0.4, 0.6, and 0.8. Before starting a single experiment, we determine which product in the case library is most similar to the product we left out. During each step in a single experiment, we use the assumptions above to compute the product the user will select. We stop when the most similar product is in D_t^* that is shown to the user or when the system terminates.

For the different systems and specifications Tables 4, 5, and 6 show the percentages of success, the percentage of successes during the first five steps of the process, and the average number of steps the system uses before it stops. The random system in Table 4 performs better for larger p as expected: the percentage of successes is higher and the average number of steps lower. As the quality of MDS representations reduces with increasing p, it becomes more difficult for the user to get an overview of the product space. Also, there is a tradeoff between the number of steps necessary to find a product and the probability to find the product. For example, a random system with $p = 10$ and $\alpha = 0.8$ finds in 84% of the cases the correct product, but needs on average almost 10 steps to find it. In this case, after 5 steps only in 20% of the cases the correct product is found. However, a system with $p = 10$ and $\alpha = 0.4$ has a success after 5 steps in 57% of the cases, but the total success rate is 59%. The average number of steps for this system is also 4.8.

The clustering system perform overall worse than the random systems and seem, therefore, not to be an alternative. However, the hierarchical systems, especially the one with $p = 10$, show similar performance as the random systems with a small α.

Table 5
Results for different specifications of the clustering system.

p	α	Successes	In 5 Steps	Average number of Steps
4	0.2	14.6%	14.6%	4.85
	0.4	20.3%	12.2%	6.68
	0.6	19.3%	8.7%	8.90
	0.8	13.7%	10.0%	7.83
6	0.2	22.1%	21.2%	4.83
	0.4	32.7%	27.1%	6.17
	0.6	44.9%	19.0%	8.30
	0.8	25.9%	11.5%	9.10
8	0.2	31.5%	29.9%	4.57
	0.4	38.9%	35.5%	5.38
	0.6	60.4%	25.6%	7.34
	0.8	45.8%	13.7%	10.20
10	0.2	39.6%	38.9%	4.34
	0.4	50.5%	45.5%	5.04
	0.6	72.6%	29.6%	6.40
	0.8	68.2%	16.2%	11.02

Table 6
Results for different specifications of the hierarchical system.

p	Successes	In 5 Steps	Average number of Steps
4	47.4%	11.5%	8.03
6	47.7%	18.4%	5.69
8	48.9%	24.6%	5.02
10	52.3%	38.0%	4.10

Tables 4, 5, and 6 only showed the success rates, but did not say anything about the cases where the most similar product was not recommended to the user. Therefore, we have also counted how many times the second, third etc. most similar product was recommended. This information is summarized for the random and hierarchical system in Tables 7 and 8. They show that many misrecommendations of the systems are recommendations of products that are quite similar to the ideal product. Looking at the top 5 of most similar products, these products are recommended in up to 94% of the cases to the customer. In many systems that desire only a small number of steps, like the hierarchical system, products 2 until 5 are recommended in one fifth of the cases to the user. In many cases, these products have a not much higher dissimilarity than the most similar product and in some cases their dissimilarity is the same. Therefore, recommending one of these products instead of the most similar one, will not make the recommendation much worse.

Table 7
Proportions of cases that the ranking of the recommended product was in the specified ranges for the random system.

		ranking ranges							
p	α	1	≤ 2	≤ 3	≤ 5	≤ 10	≤ 25	≤ 50	≤ 100
4	0.2	16.8%	24.6%	29.9%	36.5%	47.0%	66.7%	84.4%	95.0%
	0.4	29.3%	36.5%	42.4%	49.5%	60.1%	78.5%	91.0%	99.4%
	0.6	41.7%	50.5%	58.9%	64.2%	75.7%	87.9%	94.4%	100.0%
	0.8	56.1%	67.3%	73.2%	80.4%	89.4%	96.0%	98.4%	99.7%
6	0.2	29.3%	40.8%	45.2%	49.2%	60.4%	75.1%	86.6%	96.0%
	0.4	42.7%	50.5%	57.9%	64.5%	74.1%	88.5%	96.0%	97.8%
	0.6	52.7%	64.8%	69.2%	75.7%	84.1%	91.9%	96.9%	97.8%
	0.8	70.1%	80.1%	82.2%	86.3%	94.1%	97.8%	99.7%	100.0%
8	0.2	43.0%	54.8%	59.5%	64.5%	75.4%	89.1%	95.3%	98.8%
	0.4	47.0%	58.6%	63.9%	71.3%	77.3%	90.0%	94.7%	98.1%
	0.6	66.4%	76.0%	79.4%	84.7%	91.0%	96.6%	97.5%	98.8%
	0.8	80.1%	87.2%	91.0%	93.2%	97.5%	99.4%	99.4%	99.4%
10	0.2	46.4%	56.1%	60.8%	67.0%	79.8%	92.5%	96.6%	99.1%
	0.4	58.9%	70.4%	74.8%	80.4%	91.0%	95.3%	98.8%	100%
	0.6	70.4%	76.0%	81.6%	87.5%	93.2%	97.2%	98.4%	98.8%
	0.8	83.8%	89.7%	92.2%	93.8%	96.0%	98.8%	99.1%	99.1%

Table 8
Proportions of cases that the ranking of the recommended product in the specified ranges for the hierarchical system.

	ranking ranges							
p	1	≤ 2	≤ 3	≤ 5	≤ 10	≤ 25	≤ 50	≤ 100
4	47.4%	56.4%	64.2%	71.3%	77.6%	85.4%	89.7%	95.6%
6	47.7%	55.1%	62.3%	70.1%	76.0%	85.4%	90.3%	95.6%
8	48.9%	56.4%	62.3%	70.4%	77.3%	86.9%	94.1%	97.5%
10	52.3%	58.6%	67.9%	74.8%	81.6%	90.0%	93.8%	98.8%

8 Conclusions

In this paper, we presented two recommender systems which both use two dimensional spaces to represent products in. Products that are similar to each other, based on their product attributes, are represented close to each other in the 2D representation. The difference between both systems is that the GRS uses explicit input from the user, where the GSI can be used to navigate through the product space. Both were combined in a prototype application for MP3 players. Some simulation tests were performed to evaluate the quality of the 2D representations and the navigation behavior in the different implementations of the GSI. The first type of tests showed that the quality of the 2D representations in the GRS is acceptable at least up to 2D spaces with 10 products, but as expected the quality of the representations became less when p was increased. The navigation tests showed that there is a tradeoff

between the number of steps a user needs to find the best product in a certain implementation of the GSI and the probability that the user will find the best product. Results of the implementation with a clustering in each step were worse than both the random system and the hierarchical system, implying that the clustering method is not good enough to be applied in practice. To be quite certain that the best product eventually will be found, the random system with a high value for α should be preferred. If a high probability of successes in the first few steps is preferred, then one should choose a random system with a low α or a hierarchical system.

Both systems show that it is possible to combine 2D product representations with recommender systems. However, there still are several possibilities to extend and possibly improve the systems.

The first extension that can be made is to extend the domain model with other submodels, like the user model or the evaluation model. In the user model, some user account information can be stored, what means that the similarity between users is used in the recommendation process. In the evaluation model, rating information could be incorporated. The ratings can be given directly by the user or can be based on saved or purchased products. To extend the domain model in this way, more CBR steps should be implemented by the system, that is, the reuse step and the retain step. The reuse step has to be implemented, since not all information should be shown to the user and the selection can contain duplicate products. In the retain step, the combination of the recommended product, user information, and rating can be stored. If the domain model is extended, the dissimilarity metric should be adapted. For example, the rating cannot be directly used as a dissimilarity score. Also, the dissimilarity metric should be flexible, because the user should not be forced to rate products or to have a user account.

An important issue for the computation of the dissimilarities is the weighting of the different features. When an evaluation model is incorporated in the recommender system these weights can be learned by the system for the total customer population or even for individual customers. In a CBR-RS for traveling [2], for example, a weighting approach based on frequencies of features previously used in a query was used.

The GSI can also be further improved by not only allowing the customer to select a product, but also by allowing the customer to select any point in the space. With the use of external unfolding [3] an invisible product can be found that is closest to the selected point and used in the next iteration. This technique is, for example, used in an application to compare roller skates in [20].

References

[1] G. Adomavicius, A. Tuzhilin, Towards the next generation of recommender systems: A survey of the state-of-the-art and possible extensions, IEEE Transactions on Knowledge and Data Engineering 17 (6) (2005) 734–749.

[2] B. Arslan, F. Ricci, N. Mirzadeh, A. Venturini, A dynamic approach to feature weighting, Management Information Systems 6 (2002) 999–1008.

[3] I. Borg, P. J. F. Groenen, Modern Multidimensional Scaling, Springer Series in Statistics, 2nd ed., Springer, New York, 2005.

[4] R. Burke, Knowledge based recommender systems, in: J. E. Daily, A. Kent, H. Lancour (eds.), Encyclopedia of Library and Information Science, vol. 69, Supplement 32, Marcel Dekker, New York, 2000.

[5] R. Burke, Hybrid recommender systems: Survey and experiments, User Modelling and User-Adapted Interaction 12 (2002) 331–370.

[6] J. De Leeuw, Applications of convex analysis to multidimensional scaling, in: J. R. Barra, F. Brodeau, G. Romier, B. van Cutsem (eds.), Recent Developments in Statistics, Noth-Holland, Amsterdam, The Netherlands, 1977, pp. 133–145.

[7] J. De Leeuw, Convergence of the majorization method for multidimensional scaling, Journal of Classification 5 (1988) 163–180.

[8] J. De Leeuw, W. J. Heiser, Convergence of correction-matrix algorithms for multidimensional scaling, in: J. C. Lingoes, E. E. Roskam, I. Borg (eds.), Geometric representations of relational data, Mathesis, Ann Arbor, MI, 1977, pp. 735–752.

[9] D. Goldberg, D. Nichols, B. M. Oki, D. Terry, Using collaborative filtering to weave an information tapestry, Communications of the ACM 35 (12) (1992) 61–70.

[10] J. C. Gower, A general coefficient of similarity and some of its properties, Biometrics 27 (1971) 857 – 874.

[11] I. Keller, MDS-i for 1 to 1 e-commerce: A position paper, in: Proceedings of the CHI 2000 Workshop on 1-to-1 E-commerce, 2000.

[12] J. B. Kruskal, Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis, Psychometrika 29 (1) (1964) 1–27.

[13] J. Lattin, J. D. Carrol, P. E. Green, Analyzing Multivariate Data, Duxbury Applied Series, Brooks/Cole, Pacific Grove, CA, 2003.

[14] F. Lorenzi, F. Ricci, Case-based recommender systems: A unifying view, in: B. Mobasher, S. S. Anand (eds.), Intelligent Techniques for Web Personalization, vol. 3169 of Lecture Notes in Computer Science, Springer, Berlin/ Heidelberg, 2005, pp. 89–113.

[15] B. Prasad, Intelligent techniques for e-commerce, Journal of Electronic Commerce Research 4 (2) (2003) 65–71.

[16] J. B. Schafer, J. A. Konstan, J. Riedl, E-commerce recommendation applications, Data Mining and Knowledge Discovery 5 (2001) 115–153.

[17] B. Schwartz, The Paradox of Choice: Why More Is Less, HarperCollins, New York, 2004.

[18] B. Schwartz, Can there ever be too many flowers blooming?, in: W. Ivey, S. J. Tepper (eds.), Engaging Art: The Next Great Transformation of America's Cultural Life, Routledge, New York, *in press*, available at `http://www.swarthmore.edu/SocSci/bschwar1/SchwartzCulture.pdf`.

[19] P. J. Stappers, G. Pasman, Exploring a database through interactive visualised similarity scaling, in: M. W. Altom, M. G. Williams (eds.), Human Factors in Computer Systems. CHI99 Extended Abstracts., 1999.

[20] P. J. Stappers, G. Pasman, P. J. F. Groenen, Exploring databases for taste or inspiration with interactive multi-dimensional scaling, in: Proceedings of the XIVth Triennial Congress of the International Ergonomics Association and 44th Annual Meeting of the Human Factors and Ergonomics Association, 'Ergonomics for the New Millennium', 2000.

[21] A. R. Webb, Statistical Pattern Recognition, 2nd ed., John Wiley & Sons, West Sussex, 2002.