Chapter 17
Map Based Visualization of Product Catalogs

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

17.1 Introduction

Traditionally, recommender systems present recommendations in ranked lists to the user. In content- and knowledge-based recommender systems, these lists are often sorted on some notion of similarity with a query, ideal product specification, or sample product. However, a lot of information is lost in this way, since two products with the same similarity to a query can differ from this query on a completely different set of product characteristics. When using a two dimensional map based visualization of the recommendations, it is possible to retain part of this information. In the map, we can then position recommendations that are similar to each other in the same area of the map.

A domain in which this map based approach can be very useful is electronic commerce, since electronic commerce stores sell a large number of products, often described by a large number of characteristics, but that are, in certain product categories, easily recognizable by an image of the product at hand. E-commerce domains for which this holds are, for example, consumer electronics and real estate. In the industry, one tries nowadays to improve usability of this kind of websites, using a trend called visual shopping. An example is CrispyShop.com[1], which compares products on one characteristic and price using a chart. Another approach taken in visual shopping is selecting items based on visual similarity (like color and shape), such as is done by Like.com[2] and Modista[3] which is especially useful in fashion related fields. An approach more closely related to the map based approach, is taken

Martijn Kagie · Michiel van Wezel · Patrick J.F. Groenen
Econometric Institute, Erasmus University Rotterdam, The Netherlands, e-mail: `martijn@kagie.net`, `mvanwezel@acm.org`, `groenen@ese.eur.nl`

[1] `http://www.crispyshop.com`

[2] `http://www.like.com`

[3] `http://www.modista.com`

DOI 10.1007/978-0-387-85820-3_17, © Springer Science+Business Media, LLC 2011

at BrowseGoods.com[4], where the products are shown in a map ordered as a department store. Both Musicovery[5] and LivePlasma[6] show a map of songs, where the latter also created a similar map for movies. Finally, YouTube[7] has an option called Warp!, which recommends movies similar to a movie you watched and shows them in a map.

Despite this wide list of commercial map based interfaces, these interfaces lack a, publicly available, scientific foundation. In this chapter, we discuss several issues in product catalog map interfaces which we have published earlier in [21, 22, 23, 24]. This chapter combines these issues and shows applications on a real e-commerce product catalog.

In the next section, we first review some scientific methods that can be used to create such maps that are used in the e-commerce domain or in related fields. Two of these methods, namely multidimensional scaling (MDS) [3] and nonlinear principal components analysis (NL-PCA) [13, 31, 34], are discussed in more detail in Section 17.3, where they are used in two different approaches to create a product catalog map interface. The first which is based on MDS uses a flexible dissimilarity measure between products to create the map. This approach has been published earlier in [21]. The second approach based on NL-PCA and published earlier in [24] has the advantage that it also shows points representing category values (i.e. the possible values of a categorical (nominal) attribute) of attributes that can be used for navigation over the map.

One problem in both map based visualization and content-based recommendation is that different characteristics of products are not considered to be equally important by users. In Section 17.4, we introduce a way to determine attribute weights using clickstream data. We counted how often products were sold. Based on the assumption that attributes that have a high influence on sales of products are attributes that are considered to be important by users, we estimate a Poisson regression model [32, 35] and derive weights from that. This approach has earlier been described in [22].

In Section 17.5, we describe one way in which map based visualization can be combined with a recommender system. The system we propose, the graphical shopping interface, does this by combining MDS based maps with the idea of recommending by proposing [45]. In an iterative process, each time a map containing a limited set of products is shown to the user in which she can select the product she likes most. Then, in the next map, the products that are more similar to this product are shown . This recommender system approach was introduced in [23].

All these methods are applied to a new real commercial product catalog of MP3 players in Section 17.6. Prototypes of these methods are available at `http://www.browsingmap.com/mapvis.html`. This product catalog was provided by Compare Group, owner of the Dutch price comparison site `http://`

[4] `http://www.browsegoods.com`

[5] `http://www.musicovery.com`

[6] `http://www.liveplasma.com`

[7] `http://www.youtube.com`

`www.vergelijk.nl`. Finally, we draw conclusions and give directions for future research in Section 17.7.

17.2 Methods for Map Based Visualization

The use of maps displaying areas with similar items has become increasingly popular in fields where users need to browse through relatively large collections of data, such as, web searching, image browsing, and managing music playlists. Also, map based visualization fits in the trend of visual shopping interfaces introduced in industry. In this section, we discuss four scientific methods for map based visualizations in the fields mentioned above, where we focus on the visualization methods used and their advantages and disadvantages when used to browse through product catalogs. In this discussion, we specifically pay attention to the type of data normally contained in (commercial) product catalogs. These catalogs often contain numerical, categorical, and multi-valued categorical attributes. Also, there can be a lot of missing values, due to, for instance, different product specifications by different manufacturers. Note that, some of the visualization methods could also be used for visualizations in three dimensions. However, we think these 3D visualizations will be too complex for users and, therefore, we only discuss 2D visualizations.

To give somewhat more insight in the differences between the four visualization methods we discuss next, we apply them to a small product catalog. This product catalog, shown in Table 17.1, consists of ten popular MP3 players derived from the larger MP3 player product catalog we use later to show our applications on. Furthermore, we limited the number of attributes to four, which where all chosen to be numerical or binary, such that all four methods could handle them easily. Also, we selected products that did not have any missing values on these four attributes.

Table 17.1: Example data set

	Name	Price	HDD	Memory Size (GB)	Weight
1.	Apple iPod Nano	180.55	0	8	40.0
2.	Apple iPod Video (30GB)	248.41	1	30	130.0
3.	Apple iPod Video (60GB)	73.50	1	60	156.0
4.	Apple iPod Video (80GB)	324.00	1	80	156.0
5.	Apple iPod Shuffle	76.00	0	1	15.5
6.	Creative Zen Nano Plus	76.00	0	1	22.0
7.	Creative Zen Vision M (30GB)	199.50	1	30	163.0
8.	Creative Zen Vision M (60GB)	250.50	1	60	178.0
9.	Sandisk Sansa e280	129.99	0	8	75.2
10.	Sony NW-A1200	143.90	1	8	109.0

17.2.1 Self-Organizing Maps

In the field of web (search) visualization, Kohonen's Self-Organizing Map (SOM) [27, 28, 29] is among the most popular methods to create maps. Following the early work of Chen et al. [5], SOMs have, for example, been used in applications by Chung et al. [6, 7] and Yang et al. [50]. Ong et al. [36] use a SOM to create a map for online news and Van Gulik et al. [49] for a music collection on an MP3 player.

Self-organizing maps use an unsupervised neural network to cluster and reduce dimensionality at the same time. First, a grid needs to be chosen that represents the structure of the map. Often, this is a rectangular or a hexagonal grid. Informally, the SOM training process works in the following way. All grid points (often called models, weights, or prototype vectors in SOM literature) are initialized with their location on the grid and a vector in the original attribute space.

Then, following an incremental-learning approach items are randomly shown to the SOM. For an item, we first compute which model represents the item best. This model is called the winner or best matching unit (BMU) and is often determined using Euclidean distance. Then, the model and neighbors of the model, to a lesser extent, are adapted to better fit the item at hand. This is done for all items and after that the next iteration is started. During each iteration, a seen item gets less influence on both the model (grid point) and its neighbors, such that the SOM converges to a solution. There also exists a batch-learning algorithm [29] for SOM which is faster.

The main advantage of SOM is that it generally provides nice clustered maps, in which clusters are formed by a set of neighboring grid points to which items are connected and relative many empty grid points. Also, neighboring models are similar to each other providing a similarity interpretation of the map. However, this similarity interpretation is not always valid for the complete map. Although a neighboring model is generally similar to another model, it can be that there exists a model even more similar in another part of the map.

Another disadvantage is that items have the same position in the map, when they are assigned to the same grid point (model), although this can be overcome by specifying a larger grid or making a hierarchical SOM, in which a SOM is created for each model in the original SOM. Also, the original SOM is not able to handle missing values or (multi-valued) categorical attributes, but with an adaptation of the comparison metric used to determine the BMU this is possible [28].

A SOM based on the example product catalog of Table 17.1 and shown in Figure 17.1 was created using the SOM Toolbox for Matlab [1] with a 10×10 rectangular grid. This grid is also shown in the map. As can be seen, each of the products is assigned to one of the squares (grid points). The complete map is used, that is, all corners have a product assigned to them, but there seem not to be real clusters of products. Looking at the ordering of the products on the map, the vertical dimension clearly corresponds to memory size and whether the MP3 player has a hard disk drive (HDD). The horizontal dimension captures the other two attributes, but these effects are not consistent over the map, due to the nonlinear character of SOMs.

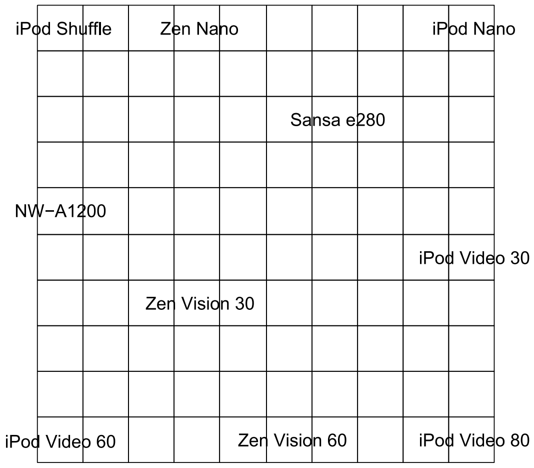


Fig. 17.1: Example of a self-organizing map on the product catalog shown in Table 17.1

17.2.2 Treemaps

Another popular visualization method which was also used to visualize web search results [48], but also to visualize the news in the commercial application NewsMap[8] of which a screenshot is shown in Chapter 15, is the treemap algorithm [44].

A treemap uses a tree, for example, resulting from a hierarchical clustering as its input. Each leaf node represents a single item. All nodes in the tree are labeled by a weight value. For non-leaf nodes, this weight value is the sum of weights of its children. When all leaf nodes are given a weight of 1, the weight value of all nodes represents the cluster size. Alternatively, items can be labeled by some measure of popularity to make these items more important in the visualization.

Starting at the root node, we divide the map vertically into as many parts as the root has children. The size of the resulting rectangles is relative to the weight values of the nodes. Then, we divide each of the rectangles horizontally using the children of the corresponding node. Alternating vertical and horizontal division, we can continue in this way until the bottom of the tree is reached and each item has a rectangle with an area relative to its weight on the map.

A main advantage of treemaps is that the complete map is filled, there are no spaces without any items and there is no overlap between items in the map. Also, its hierarchical structure is very useful for implementing zooming functionality. Since treemaps can be used in combination with hierarchical clustering algorithms, which are based on a similarity measure, they are also very flexible. A similarity measure can be chosen that is very suitable to the data at hand. For instance, when visualizing

[8] `http://marumushi.com/apps/newsmap/index.cfm`.

a product catalog, one could use the dissimilarity measure we introduce in the next section, that is able to handle missing values and mixed attribute types.

The treemap approach has two drawbacks. First of all, the original treemap algorithm often produces tall rectangles as results for small clusters or single items, which makes visualization of single products quite hard. This problem can be partly overcome by using squarified treemaps [4] or, even better, quantum treemaps [2], which guarantee a certain aspect ratio for the rectangles representing the items. The second drawback is that, although similar products are clustered together, there is no clear distance interpretation and ordering between clusters might be lost, that is, two quite similar products that are assigned to two different clusters might not be close to each other in the map.

A treemap based on the example product catalog of Table 17.1 is depicted in Figure 17.2. To create this map we first used Matlab's average linkage hierarchical clustering algorithm based on normalized Euclidean distances and visualized the resulting tree using a treemap algorithm for Matlab written by Hicklin [18]. In the map, wider lines correspond to higher level divisions of the data. Each rectangle, which all have the same area, represents a product. Compared to the SOM, the products that were at the top are on the right in the treemap. These products are the MP3 players without HDD. On the left side, we see now that the Apple iPod Video 60GB is closer to the Apple iPod Video 30GB, the Creative Zen Vision M 30GB, and the Sony NW-A1200 (since the line dividing the Apple iPod Video 80GB and the Creative Zen Vision M 60GB from the Apple iPod Video 60GB is wider than the line under it) than to the Apple iPod Video 80GB and the Creative Zen Vision M 60GB.

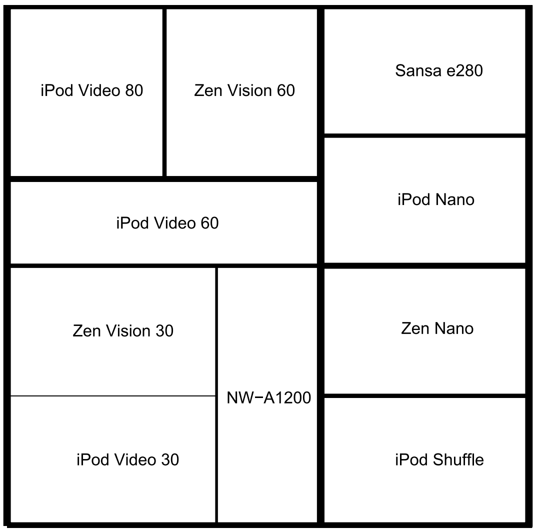


Fig. 17.2: Example of a treemap on the product catalog shown in Table 17.1

17.2.3 Multidimensional Scaling

A third class of applications uses multidimensional scaling (MDS) [3] algorithms to visualize relative large data collections. There exists a wide variety of MDS algorithms, such as classical scaling [47] (e.g. used to visualize music playlists by Donaldson [11]), nonmetric MDS [30], Sammon Mapping [42] (e.g. used to visualize a image collection by Pečenović et al. [37]), and SMACOF [10] (e.g. used to visualize a roller skates catalog by Stappers et al. [46]). However, they are all based on mapping a (symmetric) dissimilarity matrix into low dimensional Euclidean space, such that distances between pairs of points represent the dissimilarities as closely as possible. MDS is one of the methods we use to create product catalog maps and is better described in the next section.

The main advantage of MDS is that the distances in the map really correspond to the similarity between items. Secondly, when using a flexible dissimilarity measure MDS can handle missing values and mixed attribute types. Disadvantages of MDS compared to SOMs and TreeMap are that there may be a lot of empty space in and/or around the map and very similar items may (partially) overlap each other in the map. Empty spaces are a disadvantage, since more zooming is necessary on specific parts of the map to be able to use the map in a satisfying way.

In Figure 17.3, a map based on the example product catalog made using MDS is shown. We created this map using PROXSCAL, available under SPSS Categories [33], which uses the SMACOF algorithm. The dissimilarity matrix used was computed based on normalized Euclidean distances. As was expected, the corners of the map are more empty compared to the SOM. However, the positions of the products on the map (after rotation) do not differ much. Nevertheless, the MDS configuration maps the dissimilarities better, but on the other hand this makes the map less organized. Note that in most of the situations it is impossible to map all dissimilarities perfectly in two dimensions. Therefore, the solution is a compromise in which some dissimilarities are mapped better than others.

17.2.4 Nonlinear Principal Components Analysis

Probably the most well-known method to perform dimension reduction and, thus, for creating maps is principal components analysis (PCA). PCA has a numerical data matrix as input and linearly transforms the data, such that the first dimension, normally plotted in horizontal direction, explains the variance in the data as much as possible. The next dimensions (in case of a map only the second) try to do the same for the remaining variance. Also, all dimensions are uncorrelated with each other. Nonlinear principal components analysis (NL-PCA) [13, 31, 34] is a method that does the same as PCA, but is also able to handle categorical attributes and missing values. Also, using ordinal transformation, numerical attributes can be transformed nonlinearly. Additionally, the categories of the attributes also have a location in the map, that is in the center of the items belonging to that category. These category

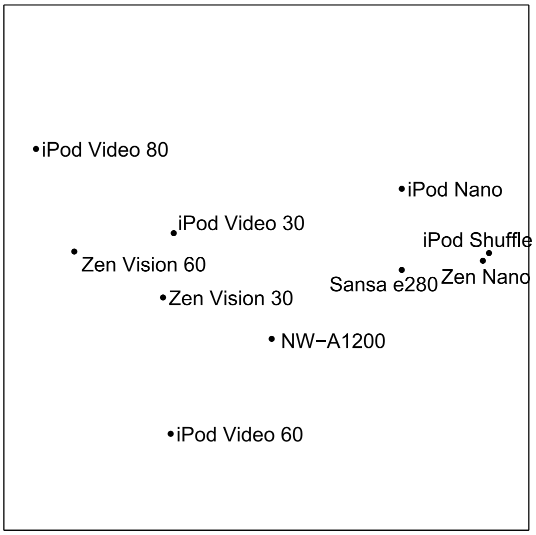


Fig. 17.3: Example of a map made with multidimensional scaling on the product catalog shown in Table 17.1

points can be used to give an interpretation to the map, but also as a navigation tool in the map based interface as is done in our product catalog map using NL-PCA, which is introduced in the next section.

Figure 17.4 shows the NL-PCA configuration of the example product catalog which was made using CATPCA also available under SPSS Categories [33]. All numerical attributes where treated as ordinal variables, such that nonlinear transformation was possible. However, due to the large flexibility of NL-PCA and the small data set, the resulting map has a lot of overlapping products. In fact, NL-PCA created clearly three clusters. Although the products in the clusters were also close together in the maps of the other three methods, they did not really form clusters. This example shows, in extreme, the ability of NL-PCA to cluster products having the same category value together.

17.3 Product Catalog Maps

In this section, we introduce two ways to create a product catalog map. We start by describing a product catalog map based on multidimensional scaling (MDS), which is combined with a *k*-means clustering algorithm to highlight some prototypical products. Thereafter, a product catalog map based on nonlinear principal components analysis (NL-PCA) is introduced which uses the category points to provide a navigation mechanism. These two methodologies were published earlier in [21] and [24].

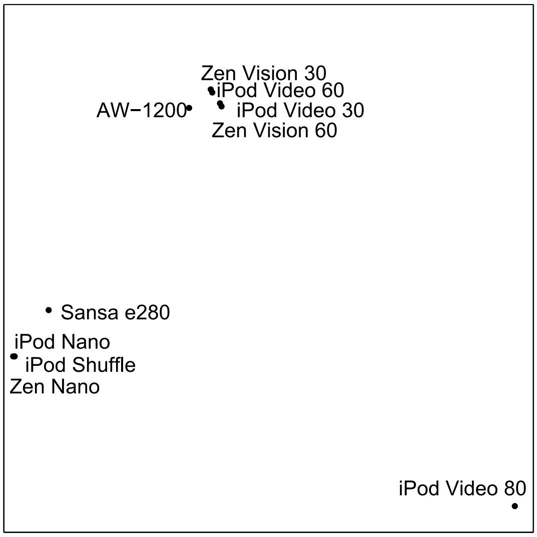


Fig. 17.4: Example of a map made with nonlinear principal components analysis on the product catalog shown in Table 17.1

In the description of these methods (and in the remainder of this paper), we use the following mathematical notation. A product catalog D contains I products $\mathbf{x}_i$ having K attributes $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{iK})$. The final two-dimensional map can be described by an $I \times 2$ matrix $\mathbf{Z}$ containing the coordinates of the products in the map.

17.3.1 Multidimensional Scaling

The first approach to create a product catalog map that we describe is based on MDS. As was mentioned in Section 17.2.3, the basis of an MDS map is a dissimilarity matrix. To compute a dissimilarity matrix $\boldsymbol{\Delta}$ from the product catalog, we need a dissimilarity measure. This measure should be able to cope with the specific data contained in a product catalog, that is, it should be able to handle missing values and numerical, categorical, and multi-valued categorical attributes.

Many popular (dis)similarity measures such as the Euclidean distance, Pearson's correlation coefficient, and the Jaccard similarity measure are not able to handle all of these attribute types. Moreover, they can not handle missing values naturally. Therefore, we use a dissimilarity measure which is an adaptation of the general coefficient of similarity proposed by Gower [14] and was introduced in [23]. Note that the MDS based product catalog map approach can also be used with other dissimilarity measures, such as co-purchases or item-item correlations.

The dissimilarity δ_{ij} between products i and j is defined as the square root of the average of nonmissing dissimilarity scores δ_{ijk} on the K attributes

$$\delta_{ij} = \sqrt{\frac{\sum_{k=1}^{K} w_k m_{ik} m_{jk} \delta_{ijk}}{\sum_{k=1}^{K} w_k m_{ik} m_{jk}}} \ , \tag{17.1}$$

where w_k is the weight of attribute k and m_{ik} and m_{jk} are binary indicators having a value of 0 when attribute k is missing for product i or j respectively and 1 otherwise. The weights w_k can be used to make some attributes more important than others. In Section 17.4, we show how these weights could be assigned automatically to match users' general preferences. The definition of the dissimilarity score δ_{ijk} depends on the type of attribute k. For all attribute types we use the same kind of normalization that ensures that the average nonmissing dissimilarity score for each attribute is equal to 1 in the product catalog. This normalization is used, to make the dissimilarity scores equally important and independent of the number of missing values.

The numerical dissimilarity score is based on the absolute distance

$$\delta_{ijk}^{N} = \frac{|x_{ik} - x_{jk}|}{\left(\sum_{i<j} m_{ik} m_{jk}\right)^{-1} \sum_{i<j} m_{ik} m_{jk} |x_{ik} - x_{jk}|} \ . \tag{17.2}$$

The dissimilarity score for categorical attributes is computed as

$$\delta_{ijk}^{C} = \frac{1(x_{ik} \neq x_{jk})}{\left(\sum_{i<j} m_{ik} m_{jk}\right)^{-1} \sum_{i<j} m_{ik} m_{jk} 1(x_{ik} \neq x_{jk})} \ , \tag{17.3}$$

where $1()$ is the indicator function returning a value of 1 when the condition is true and 0 otherwise.

To handle categorical attributes for which a product can belong to multiple categories (multi-valued categorical attributes), this dissimilarity framework was extended in [22]. There, the dissimilarity score for multi-valued categorical attributes was defined as

$$\delta_{ijk}^{M} = \frac{|x_{ik} \Delta x_{jk}|}{\left(\sum_{i<j} m_{ik} m_{jk}\right)^{-1} \sum_{i<j} m_{ik} m_{jk} (|x_{ik} \Delta x_{jk}|)} \ , \tag{17.4}$$

where both x_{ik} and x_{jk} are sets of values and Δ is the symmetric difference set operator. This measure counts how many categories there are to which one of the products at hand belongs and the other not.

As mentioned in Section 17.2.3, the aim of MDS is to map a dissimilarity matrix $\boldsymbol{\Delta}$ (having elements δ_{ij} and in our approach computed using (17.1)) as good as possible to distances between points in a low dimensional Euclidean space. This objective can be formalized by minimizing the raw Stress function [30]

$$\sigma_r(\mathbf{Z}) = \sum_{i<j} w_{ij} (\delta_{ij} - d_{ij}(\mathbf{Z}))^2 \ . \tag{17.5}$$

In this equation, $\mathbf{Z}$ is an $I \times 2$ coordinate matrix which forms the basis for the map, δ_{ij} is the dissimilarity between items i and j and $d_{ij}(\mathbf{Z})$ is the Euclidean distance between the coordinates of i and j, that is

$$d_{ij}(\mathbf{Z}) = \sqrt{\sum_{s=1}^{2} (z_{is} - z_{js})^2} \ . \tag{17.6}$$

Also, weights w_{ij} can be specified to force some dissimilarities to be fit better than others.

The dissimilarity measure we use is able to handle missing values. However, dissimilarities based on only a couple of nonmissing (and maybe even unimportant) attributes are more unreliable than dissimilarities for which no dissimilarity scores were missing. Therefore, the latter should receive higher weights. This can be done by defining the weights to be used in (17.5) as the weighted proportion of nonmissing attributes used for pair ij

$$w_{ij} = \frac{\sum_{k=1}^{K} w_k m_{ik} m_{jk}}{\sum_{k=1}^{K} w_k} \ . \tag{17.7}$$

We minimize $\sigma_r(\mathbf{Z})$ using the SMACOF algorithm [10] which is 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 and the difference between subsequent coordinate matrices $\mathbf{Z}$ converges to zero [10]. This property has an important and vital consequence for dynamic visualizations: the algorithm produces smooth changes to the points in the display leading to a (local) minimum solution of (17.5). In effect, the objects follow a smooth trajectory on the screen.

The resulting maps may look overwhelming to the user when the number of products is large. To make the map more appealing to the user, a small number of products is highlighted by showing larger sized and full color images, while other products are represented by a smaller monochrome image. These highlighted products are helpful to the user to get a quick overview of the map. Therefore, it is nice when these products represent different groups of products in this map. This was done, by first clustering the products in the map using k-means clustering [17]. We decided to perform a k-means clustering on the map $\mathbf{Z}$ instead of a hierarchical clustering procedure on the original dissimilarities for two reasons. First, this procedure is faster and, second, it is consistent with the visualization, that is, there is no overlap between the clusters in the map. In each cluster, one product is chosen to be highlighted, that is, the product closest to the cluster center based on Euclidean distance. In Section 17.6, we show a prototype of this approach using a product catalog of MP3-players.

17.3.2 Nonlinear Principal Components Analysis

The second approach for creating a product catalog map discussed here is based on NL-PCA. In this approach, a map is created in which not only the products are plotted, but also the category values of the attributes. These can then be used for navigation and selection. NL-PCA is a generalization of ordinary principal components analysis to ordinal (nonlinearly ordered) and categorical attributes. When only having numerical attributes, NL-PCA simplifies to ordinary PCA and when all attributes are categorical and a so-called multiple nominal transformation is chosen, then NL-PCA is identical to homogeneity or multiple correspondence analysis.

In homogeneity analysis, the $I \times K$ data matrix $\mathbf{X}$ is modeled by an indicator matrix $\mathbf{G}_k$ for every attribute. Let L_k denote the number of categories of attribute k. Every category $\ell, \ell = 1, \ldots, L_k$ has its own column in the $I \times L_k$ matrix $\mathbf{G}_k$, in which a 1 denotes that the object belongs to this category and a 0 that it does not. Multi-valued categorical attributes are modeled using an $I \times 2$ indicator matrix for every category. Missing values are incorporated using an $I \times I$ binary diagonal matrix $\mathbf{M}_k$ for all attributes having a value of 1 on the diagonal for nonmissing values for attribute k and 0 otherwise. Using this data representation, we can define the following loss function for homogeneity analysis [13, 34] by

$$\sigma(\mathbf{Z}; \mathbf{Y}_1, \ldots, \mathbf{Y}_K) = K^{-1} \sum_{k=1}^{K} \mathrm{tr}(\mathbf{Z} - \mathbf{G}_k\mathbf{Y}_k)'\mathbf{M}_k(\mathbf{Z} - \mathbf{G}_k\mathbf{Y}_k) \ , \tag{17.8}$$

where $\mathbf{Z}$ is a $I \times 2$ matrix representing the objects in 2D Euclidean space and the matrix $\mathbf{Y}_k$ is the 2 dimensional representations of the category values of the attribute k. Both $\mathbf{Z}$, the coordinates of objects, and all $\mathbf{Y}_k$'s, the coordinates of the attribute category values, can be plotted in a joint space which is called a biplot [15]. Essentially, $\mathbf{Z} - \mathbf{G}_k\mathbf{Y}_k$ gives the differences (or error) between the position of the individual products and the positions of the category centroids they belong to for variable k. Ideally, no error would exist and all products in the same category would be fully homogeneous and coincide with the position of their category. As there are more attributes and the products fill in different categories on the different attributes, (17.8) simply measures the squared distances of the products relative to their category centroids, hence how homogeneous the products are. The matrix $\mathbf{M}_k$ removes the contribution to the error for any product having a missing value for attribute k. Equation (17.8) can be minimized using an alternating least squares (ALS) procedure called Homals [13, 34].

From Homals, it is an easy change to NL-PCA by imposing extra restrictions on the category points. Numerical and ordinal attributes can be incorporated in the Homals framework when we impose a rank-1 restriction of the form

$$\mathbf{Y}_k = \mathbf{q}_k\mathbf{a}'_k \tag{17.9}$$

for the numerical and ordinal attributes, where $\mathbf{q}_k$ is a L_k dimensional column vector of category quantifications and $\mathbf{a}_k$ is a 2 dimensional column vector of weights.

Using this restriction the category points are forced to be on a line. However, this restriction is not sufficient to preserve the order for ordinal attributes or even the relative distance for numerical attributes. Therefore, $\mathbf{q}_k$ is constrained every ALS iteration to satisfy such restrictions. In the case of ordinal attributes this transformation is done by a weighted monotone regression [9] and in the case of numerical attributes this results in replacing $\mathbf{q}_k$ by the attribute's original values $\mathbf{x}_k$. A detailed description of the ALS algorithm for NL-PCA can be found in [13, 31, 34].

NL-PCA solutions have a number of advantageous properties that make it very suitable to create maps that contain both products and attribute category values.

- NL-PCA provides a joint space of object and attribute category points, called a biplot [15], while other visualization methods only provide the object points. The category points can be used to provide an easier interpretation of the map and to navigate through the map by selecting subsets of products.
- For categorical attributes, the category points are located in the centroids of the object points belonging to that category. This implies that when a certain attribute is well represented in the map, the object points are clustered around their corresponding category value of that attribute and a selection of all points belonging to this category will lead to a selection of a subspace of the map. For ordinal attributes, the category point will be the point on the line closest to the centroid of the object points.
- The third advantage is shared by NL-PCA and most other visualization methods. That is, the distance between object points in the map is, in general, related to dissimilarity between objects.

In Section 17.6, we also show an application of this approach to the MP3 player data set.

17.4 Determining Attribute Weights using Clickstream Analysis

Both product catalog map approaches introduced in the previous section consider all attributes of a product as equally important to the user. However, we showed that attribute weights can be incorporated in the dissimilarity measure used for MDS and also NL-PCA can be adapted to incorporate different weights for attributes. This holds for most visualization methods, including self-organizing maps and treemaps which were described in Section 17.2.

In this section, we will introduce an approach to determine these weights automatically using clickstream data. For every product, we count how often it was sold during some period. In our application, shown in Section 17.6, we actually counted outclicks, which are clicks out of the site (we used data of a price comparison site) to a shop where the product can be bought. For ease of readability, we use the term sales instead of outclicks during the remainder of this paper. Using these counts and the product attributes, we estimate a Poisson regression model, which is a model belonging to the class of generalized linear models. Using the coefficients of this model and their corresponding standard errors, we compute t-values which form the basis of our attribute weights. This approach was described earlier in [22].

17.4.1 Poisson Regression Model

A collection of models frequently used in the field of statistics are the generalized linear models (GLM) [32, 35]. To model a dependent count variable being discrete and nonnegative, such as sales in our domain, we use an appropriate member of the GLM family, that is, the Poisson regression model. In Poisson regression, we cannot use (multi-valued) categorical attributes directly, so we have to create dummy attributes instead. Therefore, every categorical attribute is represented by $L_k - 1$ dummies $x_{ik\ell}$, which are 1 for the category where the item belongs to and 0 for all other attributes, where L_k is the number of different categories for attribute k. When an item belongs to the last category L_k all dummies for this attribute will be 0. This representation is chosen to avoid multicollinearity. For multi-valued categorical attributes the same approach is used, only now all categories are represented by, in total, L_k dummies. For numerical attributes, we can just use the attribute itself. Hence, $x_{ik} = x_{ik1}$ and $L_k = 1$. We collect all $x_{ik\ell}$ for item i in vector $\mathbf{x}_i$. Also, an intercept term x_{i0} is incorporated in this vector, which equals 1 for all items. Hence, $\mathbf{x}_i = (x_{i0}, x_{i11}, \ldots, x_{iKL_K}$ Furthermore, we have the dependent count variable value y_i for all I items. Now, we can express the Poisson regression model as

$$y_i \approx \exp(\mathbf{x}_i'\mathbf{b}) \quad , \tag{17.10}$$

where $\mathbf{b}$ is a vector of regression parameters to be estimated. Furthermore, it is assumed that y_i is Poisson distributed having expectation $E(\exp(\mathbf{x}_i'\mathbf{b}))$. The Poisson regression model can be fitted by maximizing its loglikelihood function, which is often done by an iteratively reweighted least squares algorithm. Besides the model parameters $b_{k\ell}$, also standard errors σ_{kl} can be derived by this algorithm.

17.4.2 Handling Missing Values

Unfortunately, the Poisson regression model cannot cope with missing values in an integrated way. Missings are in general a problem in GLMs and Imbrahim et al. [20] recently compared different techniques that can be used to handle missing values in combination with GLMs. One of the best methods (leading to unbiased estimates and reliable standard errors) in their comparison was multiple imputation (MI) [41]. MI methods create a number of 'complete' data sets in which values for originally missing values are drawn from a distribution conditionally on the non-missing values. These imputed data sets can be created using two different methods: data augmentation [43] and sampling importance/resampling [26]. Although both methods lead to results having the same quality, the second method computes these results much faster. Therefore, an algorithm based on the second approach, namely the Amelia algorithm [26] which is available as a package [19] for the statistical software environment R, is used in our approach. For a discussion about how the regression coefficients and standard errors of these imputed datasets can be used to estimate the parameters and standard errors of the Poisson regression model, we refer to [22].

17.4.3 Choosing Weights Using Poisson Regression

There are three reasons why we cannot use the coefficients $b_{k\ell}$ resulting from the Poisson regression model as weights directly. The first reason is that dissimilarity scores and attributes do not have the same scale. Second, uncertainty about the correctness of the coefficient is not taken into account when using $b_{k\ell}$ directly. Although the value of a coefficient can be relatively high, it can still be unimportant. Consider, for example, a dummy attribute having very few 1's and a high coefficient value. Then, this high impact of the coefficient is only applicable to a limited number of items and its total importance is limited. By taking the uncertainty we have into account, we can correct for this. Third, weights should always be equal to or larger than 0, while $b_{k\ell}$ can also be negative.

The first two problems can be overcome by using the t-value

$$t_{k\ell} = \frac{b_{k\ell}}{\sigma_{kl}} \tag{17.11}$$

of coefficient $b_{k\ell}$ as a basis to compute weights. Due to standardization, all $t_{k\ell}$'s are on the same scale and, since these are standardized by division by the corresponding standard error $\sigma_{k\ell}$, uncertainty is taken into account. When we use $|t_{k\ell}|$ instead of $t_{k\ell}$ we guarantee the weights to be larger than or equal to 0.

For numerical attributes, we can map $|t_{k1}|$ directly to a 'pseudo' absolute t-value v_k for attribute k, that is, $v_k = |t_{k1}|$. Then, including a normalization of the weights (for ease of interpretability), we can compute w_k using

$$w_k = \frac{v_k}{\sum_{k'=1}^{K} v_{k'}} \ . \tag{17.12}$$

For (multi-valued) categorical attributes, we first have to compute v_k using the L_k values of $t_{k\ell}$. This is done by taking the average of the absolute $t_{k\ell}$ values

$$v_k = \frac{1}{L_k} \sum_{\ell=1}^{L_k} |t_{k\ell}| \ . \tag{17.13}$$

These v_k's can then be used to compute the weights for (multi-valued) categorical attributes using (17.12).

17.4.4 Stepwise Poisson Regression Model

The t-values $t_{k\ell}$ can be compared to a t-distribution to determine whether these values are significantly different from 0. In a so-called stepwise model, this significance testing is used for attribute selection, finally resulting in a model only having significant attributes. The stepwise model approach starts off with a model containing all attributes. Then, each time the most insignificant attribute is deleted from the model, until the model only contains significant attributes. Note that this stepwise approach leads to a different model when all insignificant attributes are deleted at once. In fact, due to collinearity, significance of an attribute may change when another attribute is deleted. When using the stepwise model to determine weights w_k, we consider L_k to be the number of dummies incorporated in the final model. This stepwise model is used in the application shown in Section 17.6.

17.5 Graphical Shopping Interface

The idea of map based visualization can be combined with recommender systems [39]. The most direct way for doing this is by just representing recommendations of the system in a map instead of a list. This approach is taken in the graphical recommender system introduced in [23]. Another system introduced in that paper, the graphical shopping interface (GSI), implements a more interactive recommendation process for users who do not have a clear idea about what they are looking for and need to shape their preferences. The GSI implements an approach which in recommender system literature is called recommendation by proposing [45] or inspiration seeking [40]. The idea is to let the user navigate through the complete product catalog in steps, where at each step a set of products is represented in a map. In this map, the user can select a product and then a new set of products, generally more similar to the selected product, is produced and visualized by MDS in a similar way as is done by the MDS based product catalog map introduced in Section 17.3.1.

In [23], three different types of the GSI were proposed, the random system, the clustering system, and the hierarchical system. Since the random system performed best in a simulation study in that paper, we only consider the random system here.

In this system, each time a small set of products is randomly selected to be shown to the user out of a larger set. This larger set contains products that are similar to a product selected by the user. First, the GSI needs to be initialized. We refer to this initialization as iteration $t = 0$. In this initialization, the larger set of products D_t is set to be the complete product catalog, that is, $D_0 = D$. Out of set D_0, p products are selected at random (without replacement). These products form together the smaller set D_0^*. Then, dissimilarity matrix $\boldsymbol{\Delta}_0^*$ is computed using the adapted Gower coefficient introduced in Section 17.3.1 and D_0^*. Finally, a map $\mathbf{Z}_0$ in which these randomly selected products are mapped is created using MDS and shown to the user.

The iterative process starts when the user selects one of the shown products we denote by $\mathbf{x}_t^*$. Then, the dissimilarities between $\mathbf{x}_t^*$ and all other products in D are computed. To create D_t, we select the $\max(p-1, \alpha^t I - 1)$ products that are most similar to $\mathbf{x}_t^*$. The parameter α, where $0 < \alpha \leq 1$, determines how much the size of D_t is decreased each iteration. Thereafter, the process is almost identical to the steps taken in the initialization. We create a small set D_t^* consisting of $\mathbf{x}_t^*$ and $p-1$ products randomly selected from D_t and compute a dissimilarity matrix $\boldsymbol{\Delta}_t^*$ based on these products. This matrix is the input for the MDS algorithm returning the new map $\mathbf{Z}_t$.

The parameter α determines how large the influence of the selections of the user are on the complete process. When $\alpha = 1$, this influence is very small, since each time a completely random selection is shown to the user except for the product selected by the user in the last iteration. When α is lower, this influence is higher, but the variance in D_t also decreases more quickly. The random system is summarized in Figure 17.5.

```
procedure RANDOMgSI($D, p, \alpha$)
    $D_0 = D$.
    Generate random $D_0^* \subset D_0$ with size $p$.
    Compute $\boldsymbol{\Delta}_0^*$ given $D_0^*$ using (17.1).
    Compute $\mathbf{Z}_0$ given $\boldsymbol{\Delta}_0^*$ using MDS.
    $t = 0$.
    repeat
        $t = t + 1$.
        Select a product $\mathbf{x}_t^* \in D_{t-1}^*$.
        Get $D_t \subset D$ containing $\max(p-1, \alpha^t I - 1)$ products most similar to $\mathbf{x}_t^*$ using (17.1).
        Generate random $D_t^* \subset D_t$ with size $p - 1$.
        $D_t^* = D_t^* \cup \mathbf{x}_t^*$.
        Compute $\boldsymbol{\Delta}_t^*$ given $D_t^*$ using (17.1).
        Compute $\mathbf{Z}_t$ given $\boldsymbol{\Delta}_t^*$ using MDS.
    until $D_t^* = D_{t-1}^*$.
end procedure
```

Fig. 17.5: Pseudocode of the graphical shopping interface.

In Section 17.6, we also show an application of the GSI.

17.6 E-Commerce Applications

In this section, we describe three working prototypes that are available online. Two of them implement the product catalog maps based on MDS and NL-PCA and the third prototype is a graphical shopping interface (GSI). Both the MDS based product catalog map and the GSI also have an option to use weights determined by the method described in Section 17.4. All prototypes are using a product catalog of MP3 players that is described next.

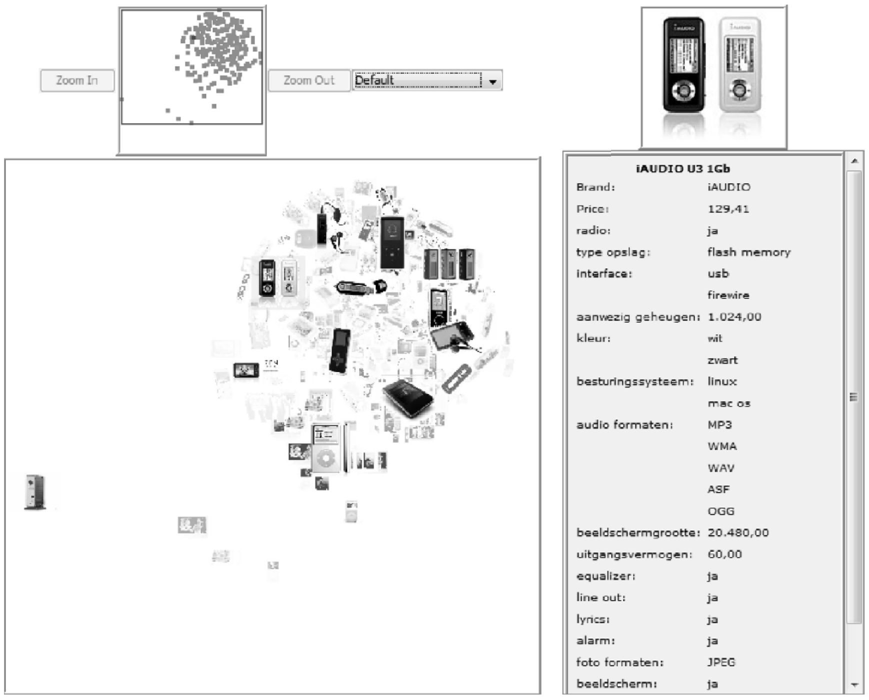
The product catalog of MP3 players was made available to us by Compare Group. Compare Group hosts, among other European price comparison sites, the Dutch price comparison site `http://www.vergelijk.nl`. The product catalog used is based on a database dump of this site from October 2007. At that moment, there were 225 MP3 players listed on the site. In total, these products were described using 45 attributes of which there were 16 numerical, 20 categorical, and 45 multi-valued categorical. Of these 45 attributes, 26 attributes were missing for more than half of the MP3 players. The other attributes are listed in Table 17.2. Note that although it also concerns MP3 players, it is different from that used in [21, 23, 24]. To be able to determine the weights automatically as described in Section 17.4, we matched the product catalog to a clickstream log of the same website logged during a period of two months from July 15 until September 15, 2007.

Table 17.2: Attribute characteristics of the MP3 player data. Only the attributes having less than 50% missing values are listed. For (multi-valued) categorical attributes only the three most occurring values are shown.

Attribute	% Missing Values	Mean
	Numerical Attributes	
Price	0.0%	134.54
Height	40.9%	63.65
Width	40.9%	48.87
Weight	44.0%	71.21
Depth	40.9%	16.54
Memory Size	3.1%	8635.30
Battery Life	44.4%	17.26
	Categorical Attributes	
Brand	0.0%	Samsung (12.0%), Creative (9.8%),Philips (8.4%)
Radio	32.0%	yes (68.6%), no (31.4%)
Extendable Memory	44.9%	yes (17.7%), no (82.3%)
Equalizer	39.6%	yes (85.3%), no (14.7%)
Screen	30.2%	yes (99.4%), no (0.6%)
Battery Type	44.4%	li-ion (44.8%), 1×AAA (33.6%), li-polymer (20.8%)
Voice Memo	24.4%	yes (81.2%), no (18.8%)
	Multi-Valued Categorical Attributes	
Storage Type	42.7%	flash memory (69.0%), hdd (21.7%), sd card (10.1%)
Interface	4.0%	usb (63.4%), usb 2.0 (31.5%), hi-speed usb (7.4%)
Color	38.2%	black (68.3%), white (20.1%), silver (16.5%)
Operating System	31.1%	windows (78.7%), mac os (34.2%), windows xp (32.3%)
Audio Formats	1.8%	MP3 (98.6%), WMA (90.0%), WAV (48.0%)

17.6.1 MDS Based Product Catalog Map Using Attribute Weights

The MDS based product catalog map prototype can be visited online at `http://www.browsingmap.com/mapvis.html`, where it is available as a Java applet. A screenshot of this prototype is shown in Figure 17.6. The GUI of the prototype mainly consists of a large focus map which gives a detailed view of a part of the complete product catalog map. In this map, a couple of products, in this case 12, are represented by a larger full color image, the other products are visualized as smaller monochrome images. By default, the monochrome images are colored according to the cluster they belong to. Alternatively, the user has the option to color them by product popularity (based on the clickstream data) or by attribute value. The small overview map at the top always shows the complete product catalog map. The user can navigate over the map in several ways. By selecting a rectangular subspace in the focus or overview map, the user can zoom in on a specific part of the map. Zooming is also possible using the mouse wheel. Finally, the user can move over the map using dragging. Since the map itself is static, that is, the coordinates of products are fixed, the computation of the map can be done offline, decreasing online computation time. Of course, it is straightforward to add traditional search technology, such as performing crisp queries, to the map based interfaces. All remaining products satisfying the constraints can then be displayed using a map. This would require the map to adapt when the product set changes and in this case client computations may be necessary.



iAUDIO U3 1Gb

Brand:	iAUDIO
Price:	129,41
radio:	ja
type opslag:	flash memory
interface:	usb
	firewire
aanwezig geheugen:	1.024,00
kleur:	wit
	zwart
besturingssysteem:	linux
	mac os
audio formaten:	MP3
	WMA
	WAV
	ASF
	OGG
beeldschermgrootte:	20.480,00
uitgangsvermogen:	60,00
equalizer:	ja
line out:	ja
lyrics:	ja
alarm:	ja
foto formaten:	JPEG
beeldscherm:	ja

Fig. 17.6: Screenshot of the MDS based product catalog map prototype.

The map of the prototype has been made using the attribute weight determination technique described in Section 17.4. These weights were determined using the data and product catalog described above. To be able to impute the missing values, we excluded attributes having more than 50% missing values and categories of (multi-valued) categorical attributes that were observed less than 10 times. The weights determined in this way are shown in Table 17.3. Note that attributes not mentioned in the table have a weight of zero and, hence, have no influence in the MDS procedure. According to our method, Brand and Memory Size are the most important attributes determining popularity of MP3 players and receive the highest weights.

Table 17.3: Weights determined using stepwise Poisson regression for the MP3 player catalog. Only selected attributes are shown.

Attribute	Weight
Brand	0.242
Memory Size	0.176
Audio Formats	0.131
Battery Type	0.106
Width	0.090
Operating System	0.086
Color	0.084
Storage Type	0.084

In Figure 17.7, we labeled the product points of both a map made by MDS using equal weights for all attributes and a map made using our weighting approach by their corresponding Brand. The second map was rotated using Procrustean transformation [16] to best match the first map. As can be seen in Figure 17.7, the products having the same brand are more clustered in the second map where brand had been made more important.

Figure 17.8a shows the second most important attribute, i.e., Memory Size. This map shows that the MP3 players are generally ordered from cheap at top left to expensive at the bottom right in the map. The most striking attribute absent in Table 17.3 and, thus, receiving a weight of zero, was Price. However, when we look at Figure 17.8b where we have made a map labeled by Price, there is a clear pattern in the map. Both effects exist, since Price highly correlates with other features of MP3 players, such as the Memory Size. These features are, hence, the features explaining most of the price that is given to a product.

More generally, we can see that the MDS produces a kind of circular shaped map with some outliers that represent products very different from all other ones. Due to this, large parts of the map are unused. On the other hand, the map provides at first sight a good interpretation, where the important attributes show a clear pattern.

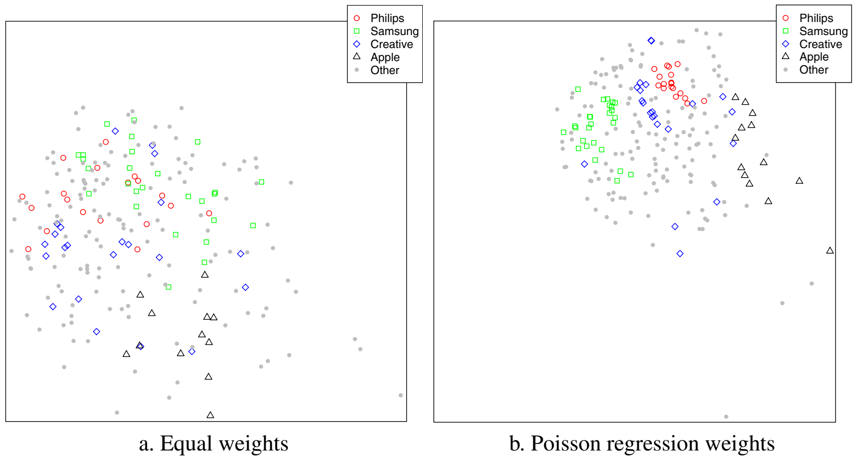


Fig. 17.7: Product catalog maps based on MDS labeled by brand.

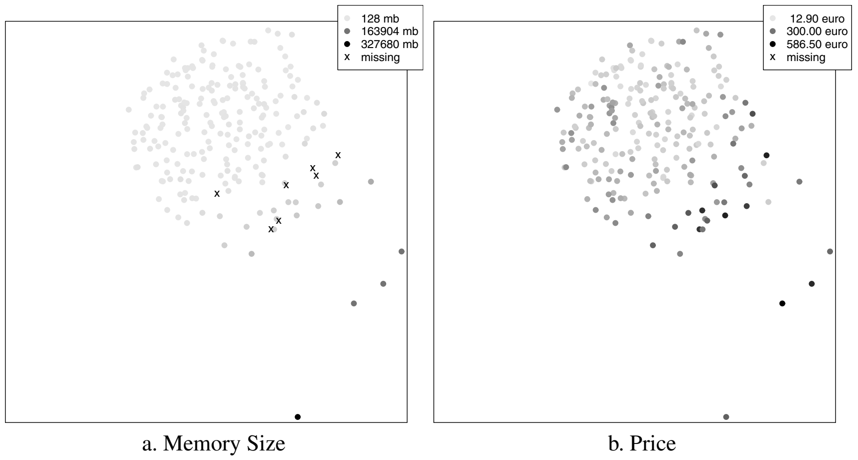
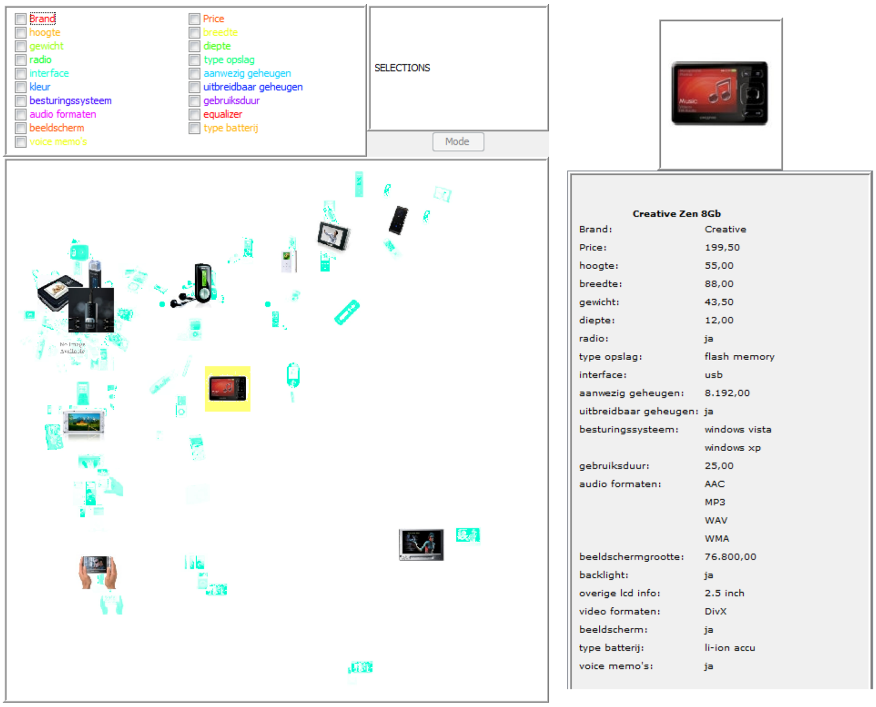


Fig. 17.8: Product catalog maps based on MDS labeled by Memory Size and Price.

17.6.2 NL-PCA Based Product Catalog Map

A screenshot of the prototype using NL-PCA is shown in Figure 17.9. This application is available online at `http://www.browsingmap.com/mapvis.html`. The main part of the GUI is the product catalog map. Initially, this map shows all products. A couple of products, in this case 12, are highlighted using a larger full color image, the other products are represented by a smaller monochrome image. For the initial map, the products that are highlighted are selected using a *k*-means clustering procedure identical to the one used in the MDS based product catalog map.



Creative Zen 8Gb

Brand:	Creative
Price:	199,50
hoogte:	55,00
breedte:	88,00
gewicht:	43,50
diepte:	12,00
radio:	ja
type opslag:	flash memory
interface:	usb
aanwezig geheugen:	8.192,00
uitbreidbaar geheugen:	ja
besturingssysteem:	windows vista
	windows xp
gebruiksduur:	25,00
audio formaten:	AAC
	MP3
	WAV
	WMA
beeldschermgrootte:	76.800,00
backlight:	ja
overige lcd info:	2.5 inch
video formaten:	DivX
beeldscherm:	ja
type batterij:	li-ion accu
voice memo's:	ja

Fig. 17.9: Screenshot of the NL-PCA based product catalog map prototype.

Above the product catalog map, there is a list with all product attributes. Each attribute can be selected by clicking on its check box. If one selects a certain attribute, the category points of this attribute are added to the map. The category points of the selected attributes not only help the user to interpret the map, they are also tools to navigate through the map. By clicking on a category point on the map this category is added to the selection list.

The selection list, which is shown to the right of the list of attributes, determines the products that are highlighted on the map. The set of highlighted products is determined as follows. For each of the selected attributes, a shown product should belong to at least one of the selected categories.

When the selection list has been adapted by adding or removing a category point, a couple of things are modified in the visualization of the map. The first thing is that all products satisfying the new constraints are colored red, while all other products are colored blue. Furthermore, a number of products is randomly selected from this set to be highlighted.

Since a selection will often lead to a subspace of the map, it is also possible to zoom in on this part of the map. However, there is no guarantee that all points in this subspace satisfy all constraints imposed by the selection list. We have chosen not to delete these product points, since these may be interesting to the user. Although these products do not satisfy all the demands of the user, they are very similar to the products that do and may have some appealing characteristics the user until then did not think of.

In both the complete and the zoomed in map, the user can click on the highlighted products to get a full product description of this selected product, which is shown at the right side of the application. However, by moving over both a full color or a monochrome image, a tooltip is shown containing an image of the product and the values of some of its most important attributes. Furthermore, the values for the attributes in the selection list are shown in this tooltip, colored green when they match the preferred category value and red when they do not. Since the GUI is based on a single NL-PCA map, this map too can be computed offline just as the MDS product catalog map.

Since the quality of NL-PCA maps may become quite poor when having a lot of missing values, we removed attributes having more than 50% missing values. Then, we also removed products having more than 50% missing values on this limited set of attributes. This resulted in a set of 189 MP3-players described by 19 attributes, namely the attributes shown in Table 17.2.

In the NL-PCA algorithm, we will treat all numerical attributes as being ordinal, because of two reasons. In the first place, many of the numerical attributes do not have a linear interpretation for users, such as, for example, the memory size. The second advantage of using the ordinal transformation is that due to the underlying monotone regression procedure some adjacent categories can be merged into a single category point. Since a numerical attribute has a lot of categories (i.e. all unique values in the data set), visualizing all these categories may become unclear and selection using these category values may become useless, since a lot of category values only belong to a single object. Using an ordinal transformation this becomes less of a problem, since categories with a small number of objects are often merged with their neighbors.

In Figure 17.10, two biplots are shown resulting from the NL-PCA algorithm. A biplot visualizes both the products labeled by their attribute value and the category points also labeled by their value. Also, the origin is plotted in the biplots. By the design of the NL-PCA method, ordinary products (products having attribute values that are similar to many other products) should be close to the origin, while more distinct products should be far away. This also holds for the categories. Since both biplots in Figure 17.10 are plots of numerical attributes, the category points are on a line. Like in the MDS map, also here both Price and Memory Size correlate with each other and are well represented, in this case on the second dimension.

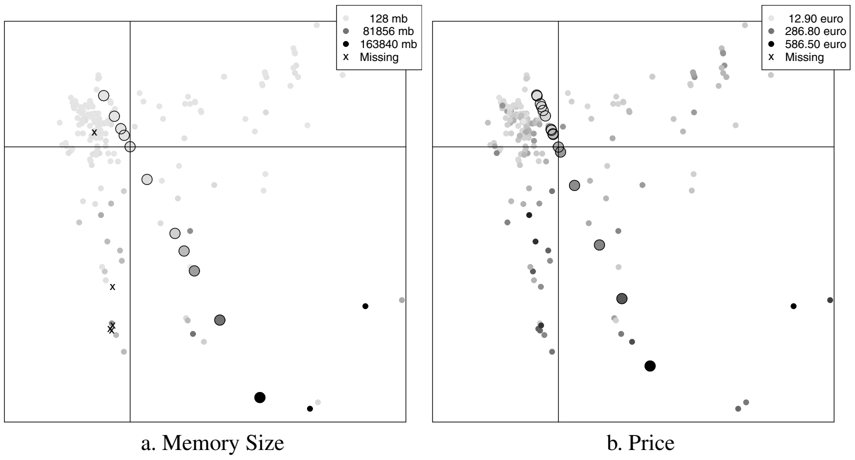


Fig. 17.10: Biplots based on NL-PCA labeled by Memory Size and Price. Large circles are category points and small dots represent object points.

Where the second dimension tells something about the memory size and attributes correlating with that, the first dimension seems to separate older from newer products. This can, for instance, be seen in Figure 17.11, where we labeled products by the Interface attributes. Since Interface is a multi-valued categorical attribute, we have chosen to label each product only by a single value, namely the one most frequent in the product catalog. Also, we only show the category points of the three most occurring categories, since all products belong to at least one of them. As said, there seem to be two groups, the older MP3 players supporting USB and the newer ones supporting USB 2.0. For the operating systems attribute one can observe a similar pattern.

More generally, the NL-PCA approach creates a map in which more of the available space is used than in the MDS approach. However, most of the map is used to visualize the special, not that common products, while the ordinary products are cluttered together near the right top corner of the map. Also, the map only shows those products and attributes that do not have too many missing values.

17.6.3 Graphical Shopping Interface

A screenshot of the graphical shopping interface prototype is shown in Figure 17.12. This prototype is available at `http://www.browsingmap.com/mapvis.`

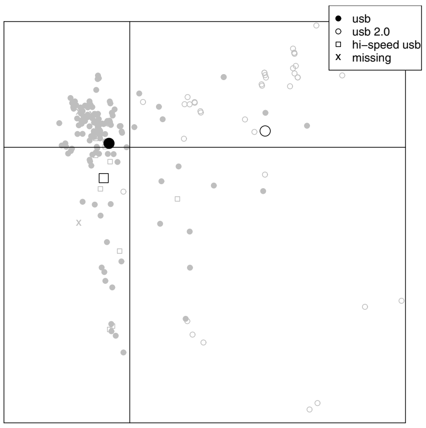


Fig. 17.11: Biplot based on NL-PCA labeled by Interface. Large shapes indicate category points and small transparent shapes represent object points.

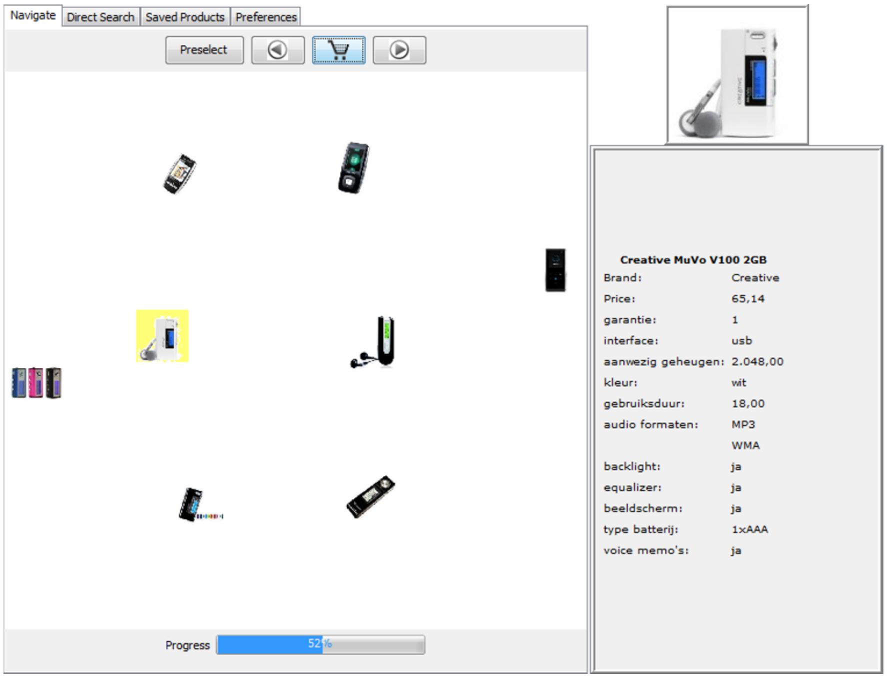
`html`. The interface uses four tabs: The Navigate tab implementing the graphical shopping interface (GSI), the Direct Search tab implementing the graphical recommender system (GRS) [23], a Preferences tab to set weights, and a Saved Products tab to save products in.

The graphical recommender system creates a map of products given a search query. The represented products are those most similar to the query. Representing these results in a map instead of, as is usually done, a list has the advantage that it makes clear which search results are similar to each other and which differ from the query on a different set of attributes.

In both the GSI and GRS map, products are represented by a thumbnail picture. By clicking, products can be selected and detailed information is shown right from the map. Above the map in the GSI tab, there are some buttons to go to the next iteration, to go one iteration back, to restart the process, and to save a product.

In the Preferences tab the user can set the weights used to compute dissimilarity in the GRS and GSI. Also, the user can deselect certain attributes, which means that their weights are set to 0. When weights are changed, the maps are immediately adapted. This tab also has a 'smart weights' button which can be used to set the weights to the values determined using the approach discussed in Section 17.4. Hence, the weights are then set to the the values shown in Table 17.3.

There is a smooth transition between two steps in the GSI. After the selection of a product by the user, the new products are added to the map at random positions. Then, the map is optimized using MDS. The optimization process is shown to the user. When the optimization has converged, the old products are gradually made less important (using the weighted version of MDS) until their influence is zero. Finally, the old products are removed from the map and the new map is optimized. This implementation yields smooth visual transitions, which are important for an effective GUI.



Creative MuVo V100 2GB

Brand:	Creative
Price:	65,14
garantie:	1
interface:	usb
aanwezig geheugen:	2.048,00
kleur:	wit
gebruiksduur:	18,00
audio formaten:	MP3
	WMA
backlight:	ja
equalizer:	ja
beeldscherm:	ja
type batterij:	1xAAA
voice memo's:	ja

Fig. 17.12: Screenshot of the MDS based product catalog map prototype.

When starting the application, the GSI tab first shows a tree of attributes and attribute values that can be used to preselect a subset of products to be used. In this way, the user can rule out products of which she is certain she is not interested in.

Since the maps of the GSI are dynamically generated, that is, they are dependent on user input and the randomization process, we do not show one here. Using another MP3 player catalog, it was shown in [23] that the theoretical quality (in terms of Stress, see (17.5)) of these maps is high, mainly since they only show about eight products. Also, the recommendation algorithm was tested in a simulation study, showing promising results.

In [23], also a usability study among 71 respondents was reported. Results of this study showed that compared to a traditional list-based interface, the users were not significantly less satisfied with the GSI, while they found it more complex. However, it was only the first time the respondents used the GSI. Some weaknesses of the GSI reported in that usability study have been taken into account in the GSI prototype discussed here. Since the graphical recommender tab was switched off during the experiment, users missed a way to formulate a query. Secondly, people missed a way to restrict the search space, which is facilitated by the selection tree in the new prototype.

However, the main disadvantage of the GSI approach seems to be its complexity. In that sense, the static product catalog maps might be a better alternative. Still, this should to be tested in a usability study.

17.7 Conclusions and Outlook

In this chapter, we have discussed how product catalogs can be visualized in a map to provide a way in which e-commerce website users may get a better overview of all products being available. In such a map, products that are similar in their attributes should be located close to each other, while dissimilar products should be located in different areas of the map.

In the framework presented in this chapter, two methods have been used to create such product catalog maps: Multidimensional scaling (MDS) and nonlinear principal components analysis (NL-PCA). MDS has the advantage that it is the only method providing a real distance interpretation. Similarity between products is matched as closely as possible to distances in a two dimensional space. We combined MDS with an adapted version of the Gower coefficient, which is very flexible, since it is able to handle mixed attribute types and missing values. The map made by MDS for the MP3 player application we have shown, seems to have a clear interpretation with a clustering of brands and a important price dimension. The main disadvantage of this map (and MDS in general) is that the map has a circular shape leaving the corners of the map open and positions outliers relatively far away from the rest of the map. However, using a weighting scheme emphasizing the small dissimilarities, this may be overcome.

NL-PCA has the advantage that it is the only method that is able to also visualize attribute categories next to the product visualization. These category points can be used to select subsets of products in the map as was shown in our prototype. In general, the interpretation of the NL-PCA map was in line with the interpretation of the MDS map. Although distinct products also take a large part of the map in the NL-PCA approach, the objects are more spread over the map. The main disadvantage of using the NL-PCA method on our product catalog was that we could not visualize all products, because NL-PCA may create poor maps when introducing objects with too many missing values. Another disadvantage is that the dissimilarity between products is not directly mapped to distances as is done in the MDS method. This can be done in NL-PCA by using a different normalization method. However, then interpretation of category points becomes more difficult which may mean that these cannot be used for navigation anymore.

Since users do usually not consider all product attributes to be equally important we have shown a method based on a Poisson regression model, which can determine attribute importance weights automatically based on counting product popularity in a clickstream log file. Since this method is independent from the visualization technique, it can be used with every technique allowing weights of attributes and it can be even applied in recommender systems which allow for attribute weights. However, the proposed method has some shortcomings. Weights for categorical attributes are determined in a quite heuristic way and interactions between attributes are ignored. Therefore, we are working on a different way to determine these weights using a more flexible model based on boosted regression trees [25].

Introducing the graphical shopping interface, we have shown one way in which map based visualization could be combined with recommendation techniques, in this case with recommendation by proposing. However, we expect that map based visualization could also be successfully combined with other content- and knowledge-based recommendation techniques, such as critiquing (see [70] and Chapter 13).

Besides combinations with other types of recommendation, we think there are some more challenges in product catalog visualization. First of all, since determining which methods provides the best map is a matter of personal taste and subject to the data to be visualized, one could also try different visualization methods, such as independent component analysis [8] or projection pursuit [12]. A good idea would be to compare different visualization approaches in a user study. In this study, we used a data set of reasonable size. Using larger product catalogs, for instance, having thousands of products, means that both the algorithms used to create the map as well as the interface itself should be able to cope with these numbers. Besides visualizing a large catalog of a single product type, another challenge might be to create a map containing multiple types of products, for instance, different electronic devices.

Acknowledgements

We thank Compare Group for making their product catalog and clickstream log files available to us.

References

1. Alhoniemi, E., Himberg, J., Parviainen, J., Vesanto, J.: SOM Toolbox 2.0 for Matlab 5. Labatory of Computer and Information Science, Helsinki University of Technology, Helsinki, Finland (1999). Available at `http://www.cis.hut.fi/projects/somtoolbox/`
2. Bederson, B.B., Schneiderman, B., Wattenberg, M.: Ordered and quantum treemaps: Making effective use of 2D space to display hierarchies. ACM Trans. Graph. **21**(4), 833–854 (2002)
3. Borg, I., Groenen, P.J.F.: Modern Multidimensional Scaling, 2nd edn. Springer Series in Statistics. Springer, New York (2005)
4. Bruls, M., Huizing, K., Van Wijk, J.J.: Squarified treemaps. In: Proceedings of Joint Eurographics and IEEE TCVG Symposium on Visualization, pp. 33–42. IEEE Press (2000)
5. Chen, H., Houston, A.L., Sewell, R.R., Schatz, B.R.: Internet browsing and searching: User evaluations of category map and concept space techniques. J. Am. Soc. Inf. Sci. **49**(7), 582–603 (1998)
6. Chung, W.: Web searching in a multilingual world. Commun. ACM **51**(5), 32–40 (2008)

7. Chung, W., Bonillas, A., Lain, G., Xi, W., Chen, H.: Supporting non-English Web searching: An experiment on the Spanish business and the Arabic medical intelligence portals. Decis. Support Syst. **42**, 1697–1714 (2006)
8. Comon, P.: Independent component analysis, a new concept? Signal Process. **36**(3), 287–314 (1994)
9. De Leeuw, J.: Correctness of Kruskal's algorithms for monotone regression with ties. Psychometrika **42**(1), 141–144 (1977)
10. De Leeuw, J.: Convergence of the majorization method for multidimensional scaling. J. Classif. **5**, 163–180 (1988)
11. Donaldson, J.: Music recommendation mapping and interface based on structural network entropy. In: V. Oria, A. Elmagarmid, F. Lochovsky, Y. Saygin (eds.) Proceedings of the 23rd International Conference on Data Engineering Workshops, pp. 811–817. IEEE Computer Society (2007)
12. Friedman, J.H., Tukey, J.W.: A projection pursuit algorithm for exploratory data analysis. IEEE Trans. Comput. **22**, 881–890 (1974)
13. Gifi, A.: Nonlinear multivariate analysis. Wiley, Chichester, UK (1990)
14. Gower, J.C.: A general coefficient of similarity and some of its properties. Biometrics **27**, 857–874 (1971)
15. Gower, J.C., Hand, D.J.: Biplots, *Monographs on Statistics and Applied Probability*, vol. 54. Chapman & Hall, London, UK (1996)
16. Green, B.F.: The orthogonal approximation of an oblique structure in factor analysis. Psychometrika **17**, 429–440 (1952)
17. Hartigan, J.A., Wong, M.A.: A k-means clustering algorithm. Appl. Stat. **28**, 100–108 (1979)
18. Hicklin, J.: Treemap for Matlab. Mathworks (2007). Available at `http://www.mathworks.com/matlabcentral/fileexchange/17192`
19. Honaker, J., King, G., Blackwell, M.: Amelia II: A Program for Missing Data (2008). R package version 1.1-27, available at `http://gking.harvard.edu/amelia`
20. Ibrahim, J.G., Chen, M.H., Lipsitz, S.R., Herring, A.H.: Missing-data methods for generalized linear models: A comparative review. J. Am. Stat. Assoc. **100**(469), 332–346 (2005)
21. Kagie, M., Van Wezel, M., Groenen, P.J.F.: Online shopping using a two dimensional product map. Lect. Notes Comput. Sci. **4655**, 89–98 (2007)
22. Kagie, M., Van Wezel, M., Groenen, P.J.F.: Choosing attribute weights for item dissimilarity using clickstream data with an application to a product catalog map. In: Proceedings of the 2nd ACM Conference on Recommender Systems, pp. 195–202. ACM Press, New York (2008)
23. Kagie, M., Van Wezel, M., Groenen, P.J.F.: A graphical shopping interface based on product attributes. Decis. Support Syst. **46**(1), 265–276 (2008)
24. Kagie, M., Van Wezel, M., Groenen, P.J.F.: An online shopping interface based on a joint product and attribute category map. In: Proceedings of IUI Workshop on Recommendation and Collaboration ReColl 2008 (2008)
25. Kagie, M., Van Wezel, M., Groenen, P.J.F.: Determination of Attribute Weights for Recommender Systems Based on Product Popularity. Tech. rep. ERS-2009-022-MKT, Erasmus Research Institute in Management, Erasmus University Rotterdam (2009).
26. King, G., Honaker, J., Joseph, A., Scheve, K.: Analyzing incomplete political science data: An alternative algorithm for multiple imputation. Am. Polit. Sci. Rev. **95**(1), 49–69 (2001)
27. Kohonen, T.: The self-organizing map. Proc. IEEE **78**(9), 1464–1480 (1990)
28. Kohonen, T.: The self-organizing map. Neurocomputing **21**, 1–6 (1998)
29. Kohonen, T.: Self-Organizing Maps, 3rd edn. Springer Series in Information Sciences. Springer, New York (2001)
30. Kruskal, J.B.: Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. Psychometrika **29**(1), 1–27 (1964)
31. Linting, M., Meulman, J.J., Groenen, P.J.F., Van der Kooij, A.J.: Nonlinear principal components analysis: Introduction and application. Psychol. Methods **12**(3), 336–358 (2007)
32. McCullagh, P., Nelder, J.A.: Generalized Linear Models, *Monographs on Statistics and Applied Probability*, vol. 37, 2nd edn. Chapman & Hall, Boca Raton (1989)

33. Meulman, J.J., Heiser, W.J.: SPSS Categories 15. SPSS Inc. (2007)
34. Michailidis, G., De Leeuw, J.: The Gifi system of descriptive multivariate analysis. Stat. Sci. **13**(4), 307–336 (1998)
35. Nelder, J.A., Wedderburn, R.W.M.: Generalized linear models. J. R. Stat. Soc. Ser. A-Stat. Soc. **135**(3), 370–384 (1972)
36. Ong, T.H., Chen, H., Sung, W., Zhu, B.: Newsmap: a knowledge map for online news. Decis. Support Syst. **39**, 583–597 (2005)
37. Pečenović, Z., Do, M.N., Vetterli, M., Pu, P.: Integrated browsing and searching of large image collections. Lect. Notes in Comput. Sci. **1929**, 173–206 (2000)
38. Reilly, J., McCarthy, K., McGinty, L., Smyth, B.: Tweaking critiquing. Knowledge-Based Syst. **18**(4–5), 143–151 (2005)
39. Resnick, P., Varian, H.R.: Recommender systems. Commun. ACM **40**(3), 56–58 (1997)
40. Ricci, F., Wöber, K., Zins, A.: Recommendation by collaborative browsing. In: A.J. Frew (ed.) Information and Communication Technologies in Tourism 2005, pp. 172–182. Springer, Vienna (2005)
41. Rubin, D.B.: Multiple Imputation for Nonresponse in Surveys. Wiley, New York (1987)
42. Sammon, J.W.: A nonlinear mapping for data structure analysis. IEEE Trans. Comput. **18**(5), 401–409 (1969)
43. Schafer, J.L., Olsen, M.K.: Multiple imputation for multivariate missing-data problems: A data analyst's perspective. Multivariate Behav. Res. **33**(4), 545–571 (1998)
44. Schneiderman, B.: Tree visualizations with tree-maps: 2-d space filling approach. ACM Trans. Graph. **11**(1), 92–99 (1992)
45. Shimazu, H.: ExpertClerk: A conversational case-based reasoning tool for developing salesclerk agents in e-commerce webshops. Artif. Intell. Rev. **18**, 223–244 (2002)
46. Stappers, P.J., Pasman, G., Groenen, P.J.F.: Exploring databases for taste or inspiration with interactive multi-dimensional scaling. In: In Proceedings IEA 2000 / HFES 2000, Ergonomics for the new Millennium, pp. 3–575–3–578. Santa Monica CA (2000)
47. Torgerson, W.S.: Multidimensional scaling: I. Theory and method. Psychometrika **17**, 401–419 (1952)
48. Turetken, O., Sharda, R.: Developement of a fisheye-based information search processing aid (FISPA) for managing information overload in the web environment. Decis. Support Syst. **37**, 415–434 (2004)
49. Van Gulik, R., Vignoli, F., Van der Wetering, H.: Mapping music in the palm of your hand, explore and discover your collection. In: Proceedings of the 5th International Conference on Music Information Retrieval (2004)
50. Yang, C.C., Chen, H., Hong, K.: Visualization of large category map for Internet browsing. Decis. Support Syst. **35**, 89–102 (2003)