INTEGRATION OF MULTIPLE FEATURE TYPES WITH ADAPTIVE RETRIEVAL IN VECTOR SPACE MODEL

Anca Doloc-Mihu[1] Vijay V. Raghavan[1] Peter Bollmann-Sdorra[2]

[1] *University of Louisiana at Lafayette, The Center of Advanced Computer Studies*

[2] *Technical University of Berlin, Department of Computer Science*

Abstract

In this work, we consider an Image Retrieval System that merges different types of features and enables learning via user feedback to improve the results. We aim to integrate these multiple feature types describing the content of the images by proposing a generalized learning scheme based on a *concatenation approach* in which two (or more) feature vectors representing a certain image are concatenated forming a new single vector that represents both feature vectors.

Let the parameter α (and $1-\alpha$, respectively) represent the amount of the importance given to each of the two feature types. We find that a decision fusion approach requires two levels of learning, one for learning the weights of the various features within the feature type and one for learning the parameter α. We show that our data/information fusion approach, which uses our concatenated vector based on the parameter α in one level of learning, gives the same result as the two levels approach. By using a Multilevel Kernel Perceptron method, our concatenation method has the advantage that the weights to specify the relative importance of feature types can be learned implicitly for certain kernel types.

Keywords

Feature Integration, Adaptive Image Retrieval, Data/Information Fusion, Decision Fusion, Vector Space Model.

1 INTRODUCTION

In this work, we consider an image retrieval system that uses different types of features, merges image rankings resulting from feature types, and uses learning via user feedback to improve the results. Learning and result merging in this context have been a concern of both the Information Retrieval community (see [3] for a survey) and the multimedia database community (previously significant work done by Fagin [7], and Ciaccia et al. [2]).

In content-based image retrieval systems, there is a question of how to combine the information derived about an image based on different feature types (e.g., annotation, color, shape, etc.).

We aim to integrate these multiple attributes describing the content of the image by proposing a generalized learning scheme. Given a single feature type (e.g., color), methods are known to learn user preferences by determining the weights of the various features within the feature type [6]. However, we are additionally interested in dynamically learning the weights required to merge the (optimal) rankings obtained from individual feature types to form a final multi-feature ranking of the image collection.

A problem that arises when combining multiple features is that the user can not easily express his or her queries appropriately in terms of the relative importance of the given feature types (like color, shape). As an example, suppose that a user wants to find images with red circles in blue background. In this case, it is not clear how to assign importance to shape and color features, i.e which of these features is more important for the user.

Many Image Retrieval Systems allow user to introduce the weights (α) as measures of their importance to the different features. For example, QBIC, Virage, RetrievalWare, and the most recent version of Photobook, FourEyes, support arbitrary combinations of visual features by allowing users to adjust the weights associated with the features [13, 15]. Fagin and Wimmers [8] propose a model of combining the relevance of several features, which was implemented in GARLIC multimedia system.

We propose a learning scheme for the feature type integration problem that adopts the *concatenation approach* in which two feature vectors representing a certain image are concatenated forming a new single vector that represents both the color and the annotation vectors. Let the parameter α represent the amount of the importance given to each of the feature types. Now, the learning process can be performed under two settings of the parameter α: when α is known (given), and when α must be learned (not known).

For the first case, we employ learning by using the *Kernel Rocchio* approach [6] with two levels of relevance. Our method can be extended to have more than two levels of relevance, in which case the user can choose a better relevance match for each image. We prove that, once the merging weights α are known, the ranking does not change if one performs first the feature type merging by using our concatenation approach and then performs learning, or if one performs first separate learning for each feature type and then uses the known importance weights to merge the resulting rankings, when the Rocchio method with a scalar kernel is used. Moreover, the above result holds also for the polynomial kernel of degree two, but it does not apply for the radial basis kernel.

The *Kernel Rocchio* algorithm combines the simplicity of Rocchio method with the power of non-linear kernel functions to improve the relevance feedback process. Therefore, in this case, the concatenation approach is more efficient than the other learning methods such as ensemble methods.

A few Image Retrieval Systems like for example, MARS, try to organize various visual features into a meaningful retrieval architecture that can dynamically adapt to different applications and different users (see for more [11]). Such systems have some capability to handle situations where the parameter α is not known.

For this latter case, the best way is to obtain these importances (or weights) dynamically via relevance feedback given by user. We propose a learning scheme involving the concatenation approach along with the Multilevel Kernel Perceptron algorithm proposed by Herbrich et al. in [9]. By using this learning method for a scalar kernel and a polynomial kernel, we find that the

solution obtained by using our concatenation approach and the solution obtained by performing learning separately for both feature types are equivalent. More exactly, both solutions can achieve the same ranking of the image collection. In this way, the user is no longer required to specify a precise weight for each feature type at the query formulation stage. In other words, by using a Kernel Perceptron method our concatenation method has the advantage that the weights can be learned implicitly for certain kernel types.

The paper is organized as follows: The previous work and the learning methods used in our work are briefly described in Section 2. Our feature type integration approach based on learning, together with our proofs, is presented in Section 3. We conclude and suggest future research in Section 4.

2 PREVIOUS WORK

The effectiveness of a CBIR system depends on the choice of visual features and of the similarity metric that models the user perception of similarity. Since the latter is very difficult to model, the current tendency in the Image Retrieval community is to use both content-based image retrieval and text-based image retrieval to enhance the performance of the Image Retrieval System. Learning and result merging in this context have been a concern of both the Information Retrieval community (see [3] for a survey) and the multimedia database community (previously significant work done by Fagin [7], and Ciaccia et al. [2]).

In content-based image retrieval systems, there is a question of how to combine the information derived about an image based on different feature types (e.g., annotation, color, shape, etc.). In this work, we assume that an image is described by multiple feature types, both textual (annotation) and visual (color). In this context, searching for images involves the mixture of these different feature types.

The problem is that when using multiple feature types, it is generally not known a priori how these feature types should be integrated in order to determine a combined best decision about the category to which an image belongs. This integration is known in the literature under the name of *fusion* [5]. Given a single feature type (e.g., color), methods to learn user preferences by determining the weights of the various features within the feature type are known to exist [6]. However, we are additionally interested in dynamically learning the weights required to merge the (optimal) rankings obtained from individual feature types to form a final multi-feature ranking of the image collection.

Suppose we have N images in the collection, and k searching tools $A_1, \ldots, A_k$, with each searching tool specialized for a particular feature type describing the images from the collection [18]. In this framework, each searching tool $j, 1 \leq j \leq k$, ranks all the N images based on how close they are to the query according to the j-th feature type. This gives us k ranked lists of images $L_1, \ldots, L_k$. Each list has a length of N, and is sorted in descending order by its characteristic feature type. In our case, $k = 2$, because we are dealing with only two feature types, color and annotation. The color description used in our system is given by the color histogram obtained in the RGB color space; the annotation is given by a weight vector in which each element indicates the believed appropriateness of the respective word (annotation, keyword) to the image description.

This merging process is depicted in Figure 1 and is known as *decision fusion* [5]. Without

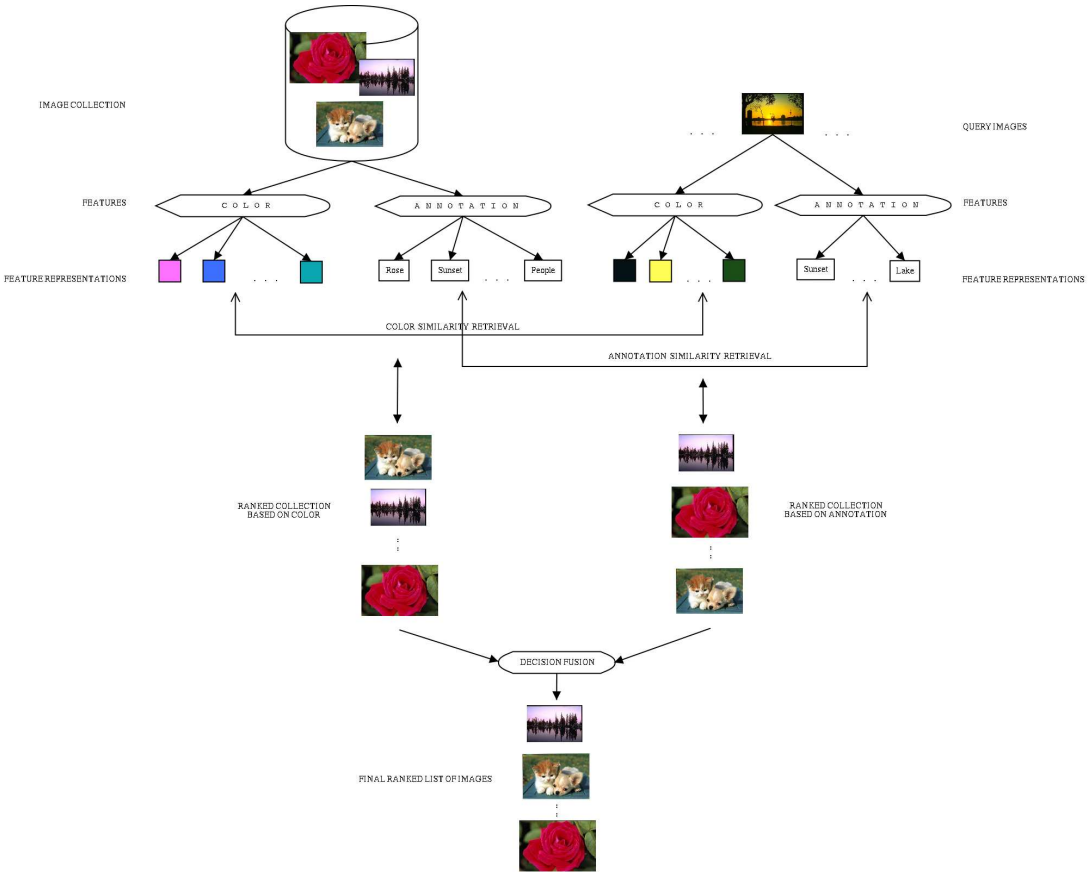

Figure 1. Decision Fusion in Image Retrieval System.

loss of generality, we can assume that the higher the score the better the rank of an image. More, we assume that each feature type applies to each image in the image collection. In order to make use of some alternative image representations, the query must be described by using the same representations (see Figure 1).

In decision fusion, the goal is to synthesize from these lists a single ordering of the image collection. For this, one can combine[1] (integrate, merge, fuse, aggregate) these lists into a single ranked list that shows an overall ranking of the images as answer to user query, with the aim of both alleviating user's work and to improve the retrieval effectiveness. For example, let us assume that a user searches for images that include the red color and the "ball" annotation, and the user cares twice as much about the color as about the annotation. In this case, the system should automatically assign twice more weight to color than to annotation.

Another fusion approach is the *data/ information fusion* [5], which deals with combining multiple representations of the same input. A probabilistic model to the feature combining problem is used in ImageRover [10] and IMKA [1], where each image is represented by its combined LSI (latent semantic indexing, for annotation) and image feature vector (for color).

[1] In this section, we provide the other terms used in the literature for the combination problem.

In our image retrieval context, this approach aims to combine different image representations (for example, color and annotation) into a general image representation. We follow this approach in Section 3.1, where we introduce a new image representation under the name of the *concatenated vector* in which two feature vectors representing a certain image are concatenated forming a new single vector that represents both the color and the annotation vectors.

A few Image Retrieval Systems like for example, MARS, try to organize various visual features into a meaningful retrieval architecture that can dynamically adapt to different applications and different users (see for more [11]). Such systems have some capability to handle situations where the weights α are not known.

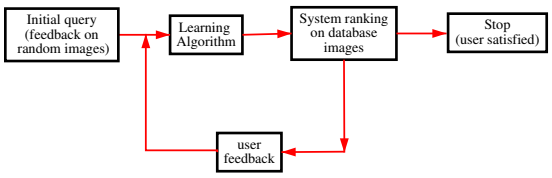

Figure 2. Relevance Feedback.

If the weights α are not known, the best way is to obtain these importances (or weights) dynamically via relevance feedback given by user. Relevance is a subjective attribute which characterizes a request-data pair given by user according to her or his needs [12]. The relevance feedback process starts with an initial query given by user. Based on this query a ranking of the data is made and presented to the user for feedback. By using this new feedback knowledge, the system makes another ranking which is further presented to the user for feedback. This feedback process continues until the user's needs are satisfied. The whole process is sketched in Figure 2.

In our Adaptive Image Retrieval System, the user only gives relevance information with respect to her or his image(s) query. Based on the feedback information received from the user as reflected in the given relevance on the retrieved images, the system dynamically adjusts the query weights to incorporate the high-level concepts and the user perceived subjectivity. We propose to employ learning by using the Kernel Rocchio approach if the weight α is known, and a Multilevel Kernel Perceptron if the weight is now known. Next we briefly describe these two learning methods in our image retrieval framework.

Kernel Rocchio. Let $\mathcal{X}$ = collection of images be our data set, $\mathcal{X} = \{\mathrm{P}_1, \mathrm{P}_2, \ldots, \mathrm{P}_N\}$, $\mathcal{R}$ = the set of the relevant images from the collection and $\mathcal{NR}$ = the set of the non-relevant images. Then, $\mathcal{X} = \mathcal{R} \cup \mathcal{NR}$ and $\mathcal{R} \cap \mathcal{NR} = \emptyset$.

The *retrieval status value* RSV corresponding to a query Q is a function $RSV : \mathcal{X} \to \mathbb{R}$ given by $RSV\,(\mathrm{P}) = \mathrm{P}^t \cdot \mathrm{Q}$, where P^t stands for the transpose of the vector P. The greater the RSV of an image, the closer the image to the query Q, or, in other words, the retrieval status value is a measure of the similarity between the image and the query Q.

Let $\Phi : \mathcal{X} \to \mathbb{R}^n$ be a continuous mapping from the n- dimensional feature space $\mathcal{X}$ to the real space, called the feature map and which satisfies:

$$< \Phi(\mathrm{P}_i), \Phi(\mathrm{P}_k) >= K(\mathrm{P}_i, \mathrm{P}_k), \tag{1}$$

where $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a kernel function [4, 14]. The Kernel Rocchio method is given by:

$$RSV(\Phi(\mathrm{P}_k)) = \frac{1}{|\mathcal{R}|}\sum_{\mathrm{P}_i \in \mathcal{R}} K(\mathrm{P}_i, \mathrm{P}_k) - \frac{1}{|\mathcal{NR}|}\sum_{\mathrm{P}_j \in \mathcal{NR}} K(\mathrm{P}_i, \mathrm{P}_k), \tag{2}$$

The *Kernel Rocchio Method* combines the simplicity of Rocchio method with the power of non-linear kernel functions in order to obtain faster results with the same computational cost.

Multilevel Kernel Perceptron. The *multilevel kernel perceptron* for learning preferences on pairs of documents, proposed in [9], is a rank perceptron that extends the problem of learning preference relation (proposed by [17]) to more than two levels of relevance. Based on training data which consist of pairs of images and their preference relations, the algorithm learns a linear mapping from image descriptions to scalar utility values. We briefly show this algorithm in the followings:

Let $\mathcal{X}$ be the image collection, and $\mathcal{T}$ the training set consisting of pairs of images and their relations, $\mathcal{T} = \left\{\left(\left(\mathrm{z}_i, \mathrm{z}_i^{'}\right), t_i\right)\right\}_{i=1}^{l}$, where $\mathrm{z}_i \succcurlyeq \mathrm{z}_i^{'}$, and $t_i = \begin{cases} 1 & \mathrm{z}_i \succcurlyeq \mathrm{z}_i^{'} \\ -1 & otherwise \end{cases}$ [9]. The utility of a image to the user need can be defined as $U : \mathcal{X} \to \mathbb{R}$, where $U(\mathrm{z}) = \sum_{k=1}^{n} w_k z_k$ is given by a n- dimensional weight vector $w = (w_1, \ldots, w_n)$. Then, images can be ranked according to their utility values, as follows:

$$\mathrm{z} \succcurlyeq \mathrm{z}^{'} \Leftrightarrow U(\mathrm{z}) > U(\mathrm{z}^{'}). \tag{3}$$

We are particularly interested in the non-linear case that includes non-linear utility functions which capture the correlations between features [9]. This is done by mapping the images to a feature space and by using the kernel methods. The linear utility function in the feature space is:

$$U(\mathrm{z}) = \widetilde{w} \cdot \Phi(\mathrm{z}).$$

The coefficients $\delta_i^* \neq 0$ can be obtained in the same way as in the linear case by working in the feature space, i.e. by replacing each image z by its mapping $\Phi(\mathrm{z})$ and applying the solution to the linear case [9] in the feature space. As a result, the optimal weight will be given by

$$w^* = \sum_{i=1}^{l} \delta_i^* t_i \left(\Phi(\mathrm{z}_i) - \Phi(\mathrm{z}_i^{'})\right),$$

and the utility function by

$$U(\mathrm{z}) = \sum_{i=1}^{l} \delta_i^* t_i \left(\Phi(\mathrm{z}_i) - \Phi(\mathrm{z}_i^{'})\right) \cdot \Phi(\mathrm{z}) = \sum_{i=1}^{l} \delta_i^* t_i \cdot K(\mathrm{z}_i - \mathrm{z}_i^{'}, \mathrm{z}). \tag{4}$$

Next, we present our feature type integration approach based on learning together with our proof.

3 FEATURE TYPES INTEGRATION BASED ON THE CONCATENATION APPROACH

3.1 Vector Concatenation Approach

For simplicity, we assume that our image collection is described by only two feature types, respectively, color and annotation. Let x be the color vector (color histogram in RGB color space, for example) representing the color feature of an image P, and y be the annotation vector representing the annotation description of the same image P. Then, by concatenating the two representations of the image P we obtain a new vector z which gives the knowledge about the image:

$$\mathrm{z} = (\alpha \mathrm{x}, (1-\alpha)\,\mathrm{y})\,, \tag{5}$$

where $\alpha > 0$. More precisely, if $\mathrm{x} = (x_1, \ldots, x_c)$ and $\mathrm{y} = (y_1, \ldots, y_a)$, then

$$\mathrm{z} = (\alpha x_1, \ldots, \alpha x_c, (1-\alpha)\,y_1, \ldots, (1-\alpha)\,y_a)\,.$$

The parameter α represents the weight or the amount of the importance the user give to each of the feature types .

Let $K_c(\mathrm{x}, \mathrm{x}')$ be the kernel for the color feature, $K_a(\mathrm{y}, \mathrm{y}')$ be the kernel for the annotation feature, and $K(\mathrm{z}, \mathrm{z}')$ be the kernel for the concatenated vector of an image P. We want to see now how these kernels are related depending on the kernel type they implement.

Scalar Kernel: $K(\mathrm{z}, \mathrm{z}') = \left\langle \mathrm{z}, \mathrm{z}' \right\rangle$

$$\begin{aligned} K(\mathrm{z}, \mathrm{z}') &= \left\langle (\alpha \mathrm{x}, (1-\alpha)\,\mathrm{y})\,, \left(\alpha \mathrm{x}', (1-\alpha)\,\mathrm{y}'\right) \right\rangle \\ &= \left\langle \alpha \mathrm{x}, \alpha \mathrm{x}' \right\rangle + \left\langle (1-\alpha)\,\mathrm{y}, (1-\alpha)\,\mathrm{y}' \right\rangle \\ &= \alpha^2 \left\langle \mathrm{x}, \mathrm{x}' \right\rangle + (1-\alpha)^2 \left\langle \mathrm{y}, \mathrm{y}' \right\rangle \\ &= \alpha^2 K_c(\mathrm{x}, \mathrm{x}') + (1-\alpha)^2 K_a(\mathrm{y}, \mathrm{y}'). \end{aligned} \tag{6}$$

Polynomial Kernel of degree $d = 2$: $K(\mathrm{z}, \mathrm{z}') = \left\langle \mathrm{z}, \mathrm{z}' \right\rangle^2$

$$\begin{aligned} K(\mathrm{z}, \mathrm{z}') &= \left\langle (\alpha \mathrm{x}, (1-\alpha)\,\mathrm{y})\,, \left(\alpha \mathrm{x}', (1-\alpha)\,\mathrm{y}'\right) \right\rangle^2 \\ &= \left[\left\langle \alpha \mathrm{x}, \alpha \mathrm{x}' \right\rangle + \left\langle (1-\alpha)\,\mathrm{y}, (1-\alpha)\,\mathrm{y}' \right\rangle \right]^2 \\ &= \left[\alpha^2 \left\langle \mathrm{x}, \mathrm{x}' \right\rangle + (1-\alpha)^2 \left\langle \mathrm{y}, \mathrm{y}' \right\rangle \right]^2 \\ &= \alpha^4 \left\langle \mathrm{x}, \mathrm{x}' \right\rangle^2 + (1-\alpha)^4 \left\langle \mathrm{y}, \mathrm{y}' \right\rangle^2 + 2\alpha^2 (1-\alpha)^2 \left\langle \mathrm{x}, \mathrm{x}' \right\rangle \left\langle \mathrm{y}, \mathrm{y}' \right\rangle \\ &= \alpha^4 K_c(\mathrm{x}, \mathrm{x}') + (1-\alpha)^4 K_a(\mathrm{y}, \mathrm{y}') + 2\alpha^2 (1-\alpha)^2 \sqrt{K_c(\mathrm{x}, \mathrm{x}') K_a(\mathrm{y}, \mathrm{y}')}. \end{aligned} \tag{7}$$

because $x_i > 0, \forall i = 1, \ldots, c$ and $y_j > 0, \forall j = 1, \ldots, a$ from the feature extraction process.

Radial Basis Kernel: $K(\mathrm{z},\mathrm{z}^{'}) = \exp\left(-\frac{\left\|\mathrm{z}-\mathrm{z}^{'}\right\|^2}{2}\right)$

$$\begin{aligned}
\left\|\mathrm{z}-\mathrm{z}^{'}\right\|^2 &= \left\langle \mathrm{z}-\mathrm{z}^{'}, \mathrm{z}-\mathrm{z}^{'} \right\rangle \\
&= \left\langle (\alpha\mathrm{x}, (1-\alpha)\,\mathrm{y}) - \left(\alpha\mathrm{x}^{'}, (1-\alpha)\,\mathrm{y}^{'}\right), (\alpha\mathrm{x}, (1-\alpha)\,\mathrm{y}) - \left(\alpha\mathrm{x}^{'}, (1-\alpha)\,\mathrm{y}^{'}\right) \right\rangle \\
&= \left\langle \left(\alpha\left(\mathrm{x}-\mathrm{x}^{'}\right), (1-\alpha)\left(\mathrm{y}-\mathrm{y}^{'}\right)\right), \left(\alpha\left(\mathrm{x}-\mathrm{x}^{'}\right), (1-\alpha)\left(\mathrm{y}-\mathrm{y}^{'}\right)\right) \right\rangle \\
&= \alpha^2 \left\langle \mathrm{x}-\mathrm{x}^{'}, \mathrm{x}-\mathrm{x}^{'} \right\rangle + (1-\alpha)^2 \left\langle \mathrm{y}-\mathrm{y}^{'}, \mathrm{y}-\mathrm{y}^{'} \right\rangle \\
&= \alpha^2 \left\|\mathrm{x}-\mathrm{x}^{'}\right\|^2 + (1-\alpha)^2 \left\|\mathrm{y}-\mathrm{y}^{'}\right\|^2
\end{aligned}$$

$$\begin{aligned}
K(\mathrm{z},\mathrm{z}^{'}) &= \exp\left(-\frac{\left\|\mathrm{z}-\mathrm{z}^{'}\right\|^2}{2}\right) = \exp\left(-\frac{\alpha^2\left\|\mathrm{x}-\mathrm{x}^{'}\right\|^2 + (1-\alpha)^2\left\|\mathrm{y}-\mathrm{y}^{'}\right\|^2}{2}\right) \qquad (8) \\
&= \exp\left(-\frac{\alpha^2\left\|\mathrm{x}-\mathrm{x}^{'}\right\|^2}{2}\right) \cdot \exp\left(-\frac{(1-\alpha)^2\left\|\mathrm{y}-\mathrm{y}^{'}\right\|^2}{2}\right) \\
&= K_c(\mathrm{x},\mathrm{x}^{'})^{-\frac{\alpha^2}{2}} \cdot K_a(\mathrm{y},\mathrm{y}^{'})^{-\frac{(1-\alpha)^2}{2}}
\end{aligned}$$

Next section describes our learning scheme based on the concatenation approach.

3.2 Multiple Feature Type Integration with Learning

3.2.1 Data Fusion by Using the Concatenation Approach

To enhance the performance of the Image Retrieval System we use a learning approach based on relevance feedback together with the fusion models. More, we try to learn dynamically the importance of the different features from users at each iteration. Our goal is to enhance the system performance, as well as to provide a better understanding of an important approach which is the feature fusion problem applied in the context of Adaptive Image Retrieval.

A problem that arises when combining multiple features is that the user can not easily express his or her queries appropriately in terms of the relative importance of the given feature types (like color, shape). As an example, suppose that a user wants to find images with red circles in blue background. In this case, it is not clear how to assign importance to shape and color features, i.e which of these features is more important for the user.

In a decision fusion approach, to get useful results, a function that dynamically incorporates the features' weights, have to be applied to produce a single score for an image from the separate scores for text and color, and to return the images ordered by this new score to the user. Given a single feature type (e.g., color), methods are known to learn user preferences by determining the weights of the various features within the feature type [6]. However, we are additionally interested in dynamically learning the weights required to merge the (optimal) rankings obtained from individual feature types to form a final multi-feature ranking of the image collection.

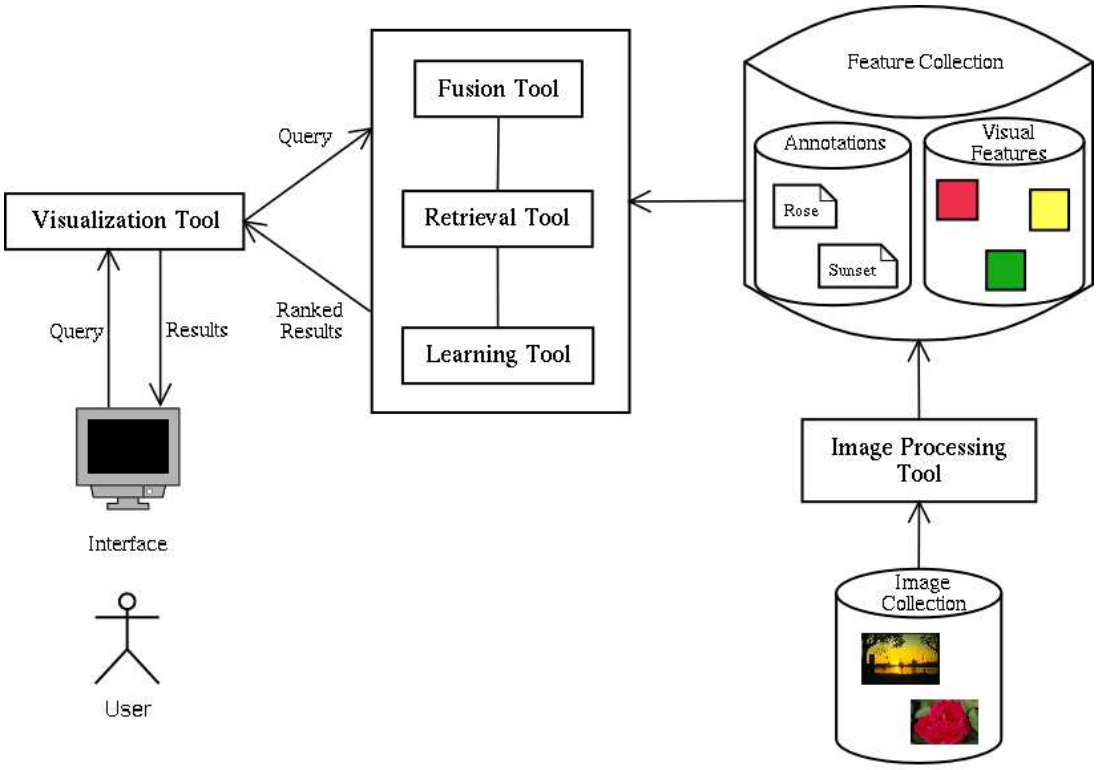

Figure 3. System architecture.

In this section we propose a generalized learning scheme that aims to integrate the different feature types by using our concatenation vector described in the previous section. Known in the literature under the name of data/ information fusion [5], the feature type integration process assumes that the different representations of an image are fused into a concatenated vector, which becomes the new representation of the image. In our learning scheme, the learning process is applied on these concatenated vectors. The process is depicted in Figure 3. In this way, the user is no longer required to specify a precise weight for each feature type at the query formulation stage. Our concatenation method has the advantage that the weights can be learned implicitly. Next, we study the two types of fusion when applied in the context of an Adaptive Image Retrieval System.

3.2.2 Learning Scheme for Feature Type Integration

We propose a learning scheme for the feature type integration problem that adopts the *concatenation approach* (Equation 5) in which two feature vectors representing a certain image are concatenated forming a new single vector that represents both the color and the annotation vectors. Let the parameter α represent the amount of the importance given to each of the feature types. Now, the learning process can be performed under two settings of the parameter α: when α is known (given), and when α must be learned (not known).

For the first case, we employ learning by using the *Kernel Rocchio* approach [6] with two levels of relevance. Our method can be extended to have more than two levels of relevance, in which case the user can choose a better relevance match for each image.

THEOREM 1. *If the merging weights α are known, the solution obtained by learning our concatenated vectors, and the solution obtained by performing first learning separately for both feature types followed by merging the resulting rankings, are equivalent, more exactly, that both solutions give the same ranking of the image collection, when the Rocchio method with a scalar kernel is used. Moreover, the above result holds also for the polynomial kernel of degree two, but it does not apply for the radial basis kernel.*

PROOF. a) Scalar Kernel. By using Equation 6 with the Kernel Rocchio formula, Equation 2, we obtain

$$\begin{aligned}
RSV(\mathrm{z}) &= \frac{1}{|\mathcal{R}|}\sum_{\mathrm{z}'\in\mathcal{R}} K(\mathrm{z},\mathrm{z}') - \frac{1}{|\mathcal{NR}|}\sum_{\mathrm{z}''\in\mathcal{NR}} K(\mathrm{z},\mathrm{z}'') \\
&= \frac{1}{|\mathcal{R}|}\sum_{\mathrm{z}'\in\mathcal{R}} \left[\alpha^2 K_c(\mathrm{x},\mathrm{x}') + (1-\alpha)^2 K_a(\mathrm{y},\mathrm{y}')\right] \\
&\quad -\frac{1}{|\mathcal{NR}|}\sum_{\mathrm{z}''\in\mathcal{NR}} \left[\alpha^2 K_c(\mathrm{x},\mathrm{x}'') + (1-\alpha)^2 K_a(\mathrm{y},\mathrm{y}'')\right] \\
&= \alpha^2\left[\frac{1}{|\mathcal{R}|}\sum_{\mathrm{x}'\in\mathcal{R}} K_c(\mathrm{x},\mathrm{x}') - \frac{1}{|\mathcal{NR}|}\sum_{\mathrm{x}''\in\mathcal{NR}} K_c(\mathrm{x},\mathrm{x}'')\right] \\
&\quad +(1-\alpha)^2\left[\frac{1}{|\mathcal{R}|}\sum_{\mathrm{y}'\in\mathcal{R}} K_a(\mathrm{y},\mathrm{y}') - \frac{1}{|\mathcal{NR}|}\sum_{\mathrm{y}''\in\mathcal{NR}} K_a(\mathrm{y},\mathrm{y}'')\right] \\
&= \alpha^2\cdot RSV_c(\mathrm{x}) + (1-\alpha)^2\cdot RSV_a(\mathrm{y}),
\end{aligned}$$

where $RSV_c(\mathrm{x})$, $RSV_a(\mathrm{y})$ and $RSV(\mathrm{z})$ represents the retrieval status values for the color vector, the annotation vector, and the concatenated vector of an image P from our collection.

b) Polynomial Kernel of degree $d = 2$. By using Equation 7 with the Kernel Rocchio formula, Equation 2, we obtain:

$$\begin{aligned}
RSV(\mathrm{z}) &= \frac{1}{|\mathcal{R}|}\sum_{\mathrm{z}'\in\mathcal{R}} K(\mathrm{z},\mathrm{z}') - \frac{1}{|\mathcal{NR}|}\sum_{\mathrm{z}''\in\mathcal{NR}} K(\mathrm{z},\mathrm{z}'') \\
&= \frac{1}{|\mathcal{R}|}\sum_{\mathrm{z}'\in\mathcal{R}} \left[\alpha^4 K_c(\mathrm{x},\mathrm{x}') + (1-\alpha)^4 K_a(\mathrm{y},\mathrm{y}') + 2\alpha^2(1-\alpha)^2\sqrt{K_c(\mathrm{x},\mathrm{x}')K_a(\mathrm{y},\mathrm{y}')}\right] \\
&\quad -\frac{1}{|\mathcal{NR}|}\sum_{\mathrm{z}''\in\mathcal{NR}} \left[\alpha^4 K_c(\mathrm{x},\mathrm{x}'') + (1-\alpha)^4 K_a(\mathrm{y},\mathrm{y}'') + 2\alpha^2(1-\alpha)^2\sqrt{K_c(\mathrm{x},\mathrm{x}'')K_a(\mathrm{y},\mathrm{y}'')}\right] \\
&= \alpha^4\left[\frac{1}{|\mathcal{R}|}\sum_{\mathrm{x}'\in\mathcal{R}} K_c(\mathrm{x},\mathrm{x}') - \frac{1}{|\mathcal{NR}|}\sum_{\mathrm{x}''\in\mathcal{NR}} K_c(\mathrm{x},\mathrm{x}'')\right]
\end{aligned}$$

$$
\begin{aligned}
&+(1-\alpha)^4\left[\frac{1}{|\mathcal{R}|}\sum_{\mathrm{y}'\in\mathcal{R}}K_a(\mathrm{y},\mathrm{y}')-\frac{1}{|\mathcal{NR}|}\sum_{\mathrm{y}''\in\mathcal{NR}}K_a(\mathrm{y},\mathrm{y}'')\right]\\
&+2\alpha^2(1-\alpha)^2\left[\frac{1}{|\mathcal{R}|}\sum_{\mathrm{z}'\in\mathcal{R}}\sqrt{K_c(\mathrm{x},\mathrm{x}')K_a(\mathrm{y},\mathrm{y}')}-\frac{1}{|\mathcal{NR}|}\sum_{\mathrm{z}''\in\mathcal{NR}}\sqrt{K_c(\mathrm{x},\mathrm{x}'')K_a(\mathrm{y},\mathrm{y}'')}\right]\\
=\;&\alpha^4\cdot RSV_c(\mathrm{x})+(1-\alpha)^4\cdot RSV_a(\mathrm{y})\\
&+2\alpha^2(1-\alpha)^2\left[\frac{1}{|\mathcal{R}|}\sum_{\mathrm{z}'\in\mathcal{R}}\sqrt{K_c(\mathrm{x},\mathrm{x}')K_a(\mathrm{y},\mathrm{y}')}-\frac{1}{|\mathcal{NR}|}\sum_{\mathrm{z}''\in\mathcal{NR}}\sqrt{K_c(\mathrm{x},\mathrm{x}'')K_a(\mathrm{y},\mathrm{y}'')}\right]
\end{aligned}
$$

The above form is a strictly monotonic increasing function, and therefore, it preserves the rankings.

c) Radial Basis Kernel.

$$K(\mathrm{z},\mathrm{z}')=K_c(\mathrm{x},\mathrm{x}')^{-\frac{\alpha^2}{2}}\cdot K_a(\mathrm{y},\mathrm{y}')^{-\frac{(1-\alpha)^2}{2}}$$

$$\ln K(\mathrm{z},\mathrm{z}') \;=\; -\frac{\alpha^2}{2}\ln K_c(\mathrm{x},\mathrm{x}')-\frac{(1-\alpha)^2}{2}\ln K_a(\mathrm{y},\mathrm{y}')$$

$$
\begin{aligned}
RSV(\mathrm{z}) &= \frac{1}{|\mathcal{R}|}\sum_{\mathrm{z}'\in\mathcal{R}}K(\mathrm{z},\mathrm{z}')-\frac{1}{|\mathcal{NR}|}\sum_{\mathrm{z}''\in\mathcal{NR}}K(\mathrm{z},\mathrm{z}'')\\
&= \frac{1}{|\mathcal{R}|}\sum_{\mathrm{z}'\in\mathcal{R}}K_c(\mathrm{x},\mathrm{x}')^{-\frac{\alpha^2}{2}}\cdot K_a(\mathrm{y},\mathrm{y}')^{-\frac{(1-\alpha)^2}{2}}-\frac{1}{|\mathcal{NR}|}\sum_{\mathrm{z}''\in\mathcal{NR}}K_c(\mathrm{x},\mathrm{x}'')^{-\frac{\alpha^2}{2}}\cdot K_a(\mathrm{y},\mathrm{y}'')^{-\frac{(1-\alpha)^2}{2}}.
\end{aligned}
$$

In this case, we need maybe some extra condition for the feature vectors in order to obtain a monotonic function which preserves the rankings.

□

The theorem shows that, once the merging weights α are known, the ranking does not change if one performs first the feature type merging by using our concatenation approach and then performs learning, or if one performs first separate learning for each feature type and then uses the known importance weights to merge the resulting rankings, when the Rocchio method with a scalar kernel is used. Moreover, the above result holds also for the polynomial kernel of degree two, but it does not apply for the radial basis kernel.

The *Kernel Rocchio* algorithm combines the simplicity of Rocchio method with the power of non-linear kernel functions to improve the relevance feedback process. Therefore, in this case, the concatenation approach is more efficient than the other learning methods such as ensemble methods [16].

If the weight α is not known, the best way is to obtain it dynamically via relevance feedback given by user. We propose a learning scheme involving the concatenation approach along with the Multilevel Kernel Perceptron algorithm proposed by Herbrich et al. in [9].

THEOREM 2. *If the merging weights α are not known, then by using the Multilevel Kernel Perceptron learning method for a scalar and a polynomial kernels, we find that the solution obtained by using our concatenation approach and the solution obtained by performing learning separately for both feature types are equivalent, more exactly, that both solutions give the same ranking of the image collection.*

PROOF. The utility value of an image for the non-linear case is given [9] by

$$U(\mathrm{z}) = \sum_{i=1}^{l} \delta_i^* t_i K\left(\mathrm{z}_i^{'} - \mathrm{z}_i^{"}, \mathrm{z}\right).$$

a) Scalar Kernel. By re-writing the weights needed for learning and by using Equation 6, we obtain:

$$\begin{aligned} U(\mathrm{z}) &= \sum_{i=1}^{l} \delta_i^* t_i K\left(\mathrm{z}_i^{'} - \mathrm{z}_i^{"}, \mathrm{z}\right) \\ &= \sum_{i=1}^{l} \delta_i^* t_i \left[\alpha^2 K_c\left(\mathrm{x}_i^{'} - \mathrm{x}_i^{"}, \mathrm{x}\right) + (1-\alpha)^2 K_a\left(\mathrm{y}_i^{'} - \mathrm{y}_i^{"}, \mathrm{y}\right)\right] \\ &= \sum_{i=1}^{l} \left(\delta_i^* \cdot \alpha^2\right) t_i K_c\left(\mathrm{x}_i^{'} - \mathrm{x}_i^{"}, \mathrm{x}\right) + \sum_{i=1}^{l} \left(\delta_i^* \cdot (1-\alpha)^2\right) t_i K_a\left(\mathrm{y}_i^{'} - \mathrm{y}_i^{"}, \mathrm{y}\right). \end{aligned}$$

On the other hand

$$\begin{aligned} U_c(\mathrm{x}) &= \sum_{i=1}^{l} \gamma_i^* t_i K_c\left(\mathrm{x}_i^{'} - \mathrm{x}_i^{"}, \mathrm{x}\right), \\ U_a(\mathrm{y}) &= \sum_{i=1}^{l} \rho_i^* t_i K_a\left(\mathrm{y}_i^{'} - \mathrm{y}_i^{"}, \mathrm{y}\right). \end{aligned}$$

If we write $\gamma_i^* = \delta_i^* \cdot \alpha^2$ and $\rho_i^* = \delta_i^* \cdot (1-\alpha)^2$, then $U(\mathrm{z}) = U_c(\mathrm{x}) + U_a(\mathrm{y})$.

b) Polynomial Kernel of degree $d = 2$.

$$\begin{aligned} U(\mathrm{z}) &= \sum_{i=1}^{l} \delta_i^* t_i K\left(\mathrm{z}_i^{'} - \mathrm{z}_i^{"}, \mathrm{z}\right) \\ &= \sum_{i=1}^{l} \delta_i^* t_i [\alpha^4 K_c\left(\mathrm{x}_i^{'} - \mathrm{x}_i^{"}, \mathrm{x}\right) + (1-\alpha)^4 K_a\left(\mathrm{y}_i^{'} - \mathrm{y}_i^{"}, \mathrm{y}\right) + \\ &\quad +2\alpha^2 (1-\alpha)^2 \sqrt{K_c(\mathrm{x}_i^{'} - \mathrm{x}_i^{"}, \mathrm{x}) K_a(\mathrm{y}_i^{'} - \mathrm{y}_i^{"}, \mathrm{y})}] \\ &= \sum_{i=1}^{l} \left(\delta_i^* \cdot \alpha^4\right) t_i K_c\left(\mathrm{x}_i^{'} - \mathrm{x}_i^{"}, \mathrm{x}\right) + \sum_{i=1}^{l} \left(\delta_i^* \cdot (1-\alpha)^4\right) t_i K_a\left(\mathrm{y}_i^{'} - \mathrm{y}_i^{"}, \mathrm{y}\right) \\ &\quad + \sum_{i=1}^{l} \left[\left(\delta_i^* \cdot 2\alpha^2 (1-\alpha)^2\right) t_i \cdot \sqrt{K_c(\mathrm{x}_i^{'} - \mathrm{x}_i^{"}, \mathrm{x}) K_a(\mathrm{y}_i^{'} - \mathrm{y}_i^{"}, \mathrm{y})}\right]. \end{aligned}$$

On the other hand

$$\begin{aligned} U_c(\mathrm{x}) &= \sum_{i=1}^{l} \gamma_i^* t_i K_c\left(\mathrm{x}_i^{'} - \mathrm{x}_i^{"}, \mathrm{x}\right), \\ U_a(\mathrm{y}) &= \sum_{i=1}^{l} \rho_i^* t_i K_a\left(\mathrm{y}_i^{'} - \mathrm{y}_i^{"}, \mathrm{y}\right). \end{aligned}$$

If we write $\gamma_i^* = \delta_i^* \cdot \alpha^4$ and $\rho_i^* = \delta_i^* \cdot (1-\alpha)^4$, then $\gamma_i^* \rho_i^* = \delta_i^{*2} \cdot \alpha^4 (1-\alpha)^4 > 0$. Then, $\sqrt{\gamma_i^* \rho_i^*} = \delta_i^* \cdot \alpha^2 (1-\alpha)^2$ which implies $U(\mathrm{z}) = U_c(\mathrm{x}) + U_a(\mathrm{y}) + \sum_{i=1}^{l} t_i \sqrt{2\gamma_i^* K_c(\mathrm{x}_i^{'} - \mathrm{x}_i^{"}, \mathrm{x}) \cdot 2\rho_i^* K_a(\mathrm{y}_i^{'} - \mathrm{y}_i^{"}, \mathrm{y})}$. □

In this way, the user is no longer required to specify a precise weight for each feature type at the query formulation stage. By using a Kernel Perceptron method our concatenation method has the advantage that the weights can be learned implicitly for certain kernel types.

Observation. If we apply the method with a scalar kernel and we use Equation 6, we obtain

$$\begin{aligned} U(\mathrm{z}) &= \sum_{i=1}^{l} \delta_i^* t_i K\left(\mathrm{z}_i^{'} - \mathrm{z}_i^{"}, \mathrm{z}\right) \\ &= \sum_{i=1}^{l} \delta_i^* t_i \left\langle \mathrm{z}_i^{'} - \mathrm{z}_i^{"}, \mathrm{z} \right\rangle \\ &= \sum_{i=1}^{l} \delta_i^* t_i \left\langle \left(\alpha\left(\mathrm{x}_i^{'} - \mathrm{x}_i^{"}\right), (1-\alpha)\left(\mathrm{y}_i^{'} - \mathrm{y}_i^{"}\right)\right), (\alpha\mathrm{x}, (1-\alpha)\,\mathrm{y}) \right\rangle \\ &= \sum_{i=1}^{l} \delta_i^* t_i \left[\left\langle \alpha\left(\mathrm{x}_i^{'} - \mathrm{x}_i^{"}\right), \alpha\mathrm{x} \right\rangle + \left\langle (1-\alpha)\left(\mathrm{y}_i^{'} - \mathrm{y}_i^{"}\right), (1-\alpha)\,\mathrm{y} \right\rangle\right] \\ &= \alpha^2 \sum_{i=1}^{l} \delta_i^* t_i \left\langle \mathrm{x}_i^{'} - \mathrm{x}_i^{"}, \mathrm{x} \right\rangle + (1-\alpha)^2 \sum_{i=1}^{l} \delta_i^* t_i \left\langle \mathrm{y}_i^{'} - \mathrm{y}_i^{"}, \mathrm{y} \right\rangle \\ &= \alpha^2 U_c(\mathrm{x}) + (1-\alpha)^2 U_a(\mathrm{y}). \end{aligned}$$

Notice that this formula is similar with the result obtained when Kernel Rocchio method is applied to the same scalar kernel. Next section concludes the paper.

4 CONCLUSIONS

In this work, we consider an Image Retrieval System that merges and learns different types of features via user feedback to improve the quality of the results. We aim to integrate these multiple feature types describing the content of the images by proposing a generalized learning scheme, called the *concatenation approach,* in which two feature vectors representing a certain image are concatenated forming a new single vector that represents both the color and the annotation vectors. Given a single feature type (e.g., color), methods are known to learn user preferences by determining the weights of the various features within the feature type [6]. However, we are additionally interested in dynamically learning the weights required to merge the (optimal)

rankings obtained from individual feature types to form a final multi-feature ranking of the image collection.

Let the parameter α (and $1-\alpha$, respectively) represent the amount of the importance given to each of the two feature types. Now, the learning process can be performed under two settings of the parameter α: when α is known (given), and when α must be learned (not known).

For the first case, we employ learning by using the *Kernel Rocchio* approach [6] with two levels of relevance. We prove that, once the merging weights α are known, the ranking does not change if one performs first the feature type merging by using our concatenation approach and then performs learning, or if one performs first separate learning for each feature type and then uses the known importance weights to merge the resulting rankings, when the Rocchio method with a scalar kernel is used. Moreover, the above result holds also for the polynomial kernel of degree two, but it does not apply for the radial basis kernel.

For the latter case, the best way is to obtain these importances (or weights) dynamically via relevance feedback given by user. We propose a learning scheme involving the concatenation approach along with the Multilevel Kernel Perceptron algorithm proposed by Herbrich et al. in [9]. By using this learning method for a scalar kernel and a polynomial kernel, we find that the solution obtained by using our concatenation approach and the solution obtained by performing learning separately for both feature types are equivalent, more exactly, that both solutions give the same ranking of the image collection. In this way, the user is no longer required to specify a precise weight for each feature type at the query formulation stage, i.e. the weights can be learned implicitly for certain kernel types.

In conclusion, we find that a decision fusion approach requires two levels of learning, one for learning the weights of the various features within the feature type and one for learning the parameter α, but it gives the same result as a data/information fusion approach, which uses our concatenated vector based on the parameter α in one level of learning. By using a Multilevel Kernel Perceptron method our concatenation method has the advantage that the weights can be learned implicitly for certain kernel types.

In future work, we may study the problem when other types of kernels are used (e.g. radial basis). Besides, the implications of more than two feature types could be interesting.

References

[1] Ana B. Benitez and Shih-Fu Chang. Perceptual Knowledge Construction from Annotated Image Collections. In *Proceedings of the International Conference on Multimedia & Expo, ICME*, Lausanne, Switzerland, August 26-29 2002.

[2] Paolo Ciaccia, Marco Patella, and Pavel Zezula. Processing Complex Similarity Queries with Distance-based Access Methods. In *Proceedings of the 6th International Conference on Extending Database Technology, EDBT*, volume 1377 of *Lecture Notes in Computer Science*, pages 9–23, Valencia, Spain, March 1998. Springer-Verlag.

[3] W. Bruce Croft, editor. *Advances in Information Retrieval*, chapter Combining Approaches to Information Retrieval, pages 1–36. Kluwer Academic Publishers, 2000.

[4] Felipe Cucker and Steve Smale. On the mathematical foundations of learning. *Bulletin of the American Mathematical Society*, 39(1):1–49, 2001.

[5] Belur V. Dasarathy. Information/Decision Fusion - Principles and Paradigms. In Nageswara Rao, Vladimir Protopopescu, Jacob Barhen, and Guna Seetharaman, editors, *Proceedings of the Workshop on Foundations of Information/Decision Fusion with Applications to Engineering Problems*, pages 46–60, Washington, D.C., August 7-9 1996.

[6] Anca Doloc-Mihu, Vijay V. Raghavan, and Peter Bollmann-Sdorra. Color Retrieval in Vector Space Model. In *Proceedings of the 26th International ACM SIGIR Workshop on Mathematical/Formal Methods in Information Retrieval MF/IR*, Toronto, Canada, August 2003.

[7] Ronald Fagin, Ravi Kumar, and D. Sivakumar. Efficient Similarity Search and Classification via Rank Aggregation. In *Proceedings of the 2003 ACM International Conference on Management of Data, SIGMOD*, pages 301–312, San Diego, CA, June 9-12 2003. ACM Press.

[8] Ronald Fagin and Edward Wimmers. Incorporating User Preferences in Multimedia Queries. In *Proceedings of the 6th International Conference on Database Theory*, volume 1186 of *Lecture Notes in Computer Science*, pages 247–261. Springer-Verlag, January 1997.

[9] Ralf Herbrich, Thore Graepel, Peter Bollmann-Sdorra, and Klauss Obermayer. Learning Preference Relations for Information Retrieval. In *Proceedings of the AAAI Workshop Text Categorization and Machine Learning*, pages 80–84, Madison, USA, 1998.

[10] Michael S. Lew, editor. *Principles of Visual Information Retrieval.* Springer-Verlag, London, January, 2001.

[11] K. Porkaev, S. Mehrotra, and Michael Ortega. Query Reformulation for Content Based Multimedia Retrieval in MARS. In *Proceedings of IEEE International Conference on Multimedia Computing and Systems, ICMCS*, volume 2, pages 747–751, Florence, Italy, June 07-11 1999.

[12] J. J. Rocchio. *Relevance feedback in Information Retrieval*, chapter 14, pages 313–323. Prentice-Hall, Inc., 1971.

[13] Y. Rui, T. S. Huang, and S. F. Chang. Image retrieval: Current techniques, promising directions and open issues. *Journal of Visual Communication and Image Representation*, 10(1):39–62, 1999.

[14] Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola, editors. *Advances in Kernel Methods.* The MIT Press, Cambridge, Massachusetts, 1999.

[15] Remco C. Veltkamp and Mirela Tanase. Content-based image retrieval systems: A survey. Technical report, Ultrecht University, http://www.aa-lab.cs.uu.nl/cbirsurvey/cbir-survey/, 2000.

[16] David H. Wolpert. Stacked Generalization. *Neural Networks*, 5(2):241–259, 1992.

[17] S. K. M. Wong, Y. Yao, and P. Bollmann. Linear structure in information retrieval. In *Proceedings of the 11th SIGIR Conference*, pages 219–232, Grenoble, 1988.

[18] Ronald R. Yager and Vladik Kreinovich. On How to Merge Sorted Lists Coming from Different Web Search Tools. *Soft Computing - A Fusion of Fundations, Methodologies and Applications*, 3(2):83–88, September 1999.