IMAGE RETRIEVAL BASED ON FEATURE WEIGHTING AND RELEVANCE FEEDBACK

M. L. Kherfi, and D. Ziou

CoRIMedia, Université de Sherbrooke, Sherbrooke, Qc, Canada, J1K 2R1
email: {mohammed.lamine.kherfi, djemel.ziou}@usherbrooke.ca

ABSTRACT

In this paper, we present a relevance feedback model for CBIR, based on a feature weighting algorithm. The proposed model uses positive and negative items selected by the user to learn the importance of image features, then applies the obtained weights to define similarity measures corresponding to the user's perception. The basic principle of this work is to give more importance to features with a high likelihood and those which separate well between positive example (PE) classes and negative example (NE) classes. The proposed algorithm was validated separately and in image retrieval context, and the experiments show that it contributes in improving retrieval effectiveness.

1. INTRODUCTION

Relevance feedback (RF) is an automatic process introduced in the mid-1960s in text retrieval techniques to improve retrieval effectiveness. Original work on RF include [8] and [11]. In the context of content-based image retrieval (CBIR), researchers felt soon the need to integrate RF in order to overcome some difficulties. First, it is not always easy for the user to express his needs using an example-based query. Second, the retrieval system can fail in translating the user's needs into image features and similarity measures. Since that, iterative refinement via RF proves to be essential because it helps model the user subjectivity. Indeed, it has been noted that user judgement on the similarity between images is subjective and depends on many factors such as the importance he pays to each feature [5]. Hence, to be able to better meet the user's needs and correctly answer his queries, there is a need to integrate him in the process of RF in order to learn the feature weights, then apply the learnt information to define similarity measures that better correspond to his judgement. In this paper, we describe a RF model that interacts with the user and, on the basis on a feature weighting algorithm, learns the importance he pays to each feature in the process of retrieval. In Section 2, we give a review of the state of the art. In Section 3, we describe our feature weighting algorithm. Section 4 gives an evaluation of the proposed algorithm separately and another evaluation in image retrieval context.

2. STATE OF THE ART

Early CBIR systems that adopted RF were built upon the vector model in information retrieval theory [12]. They use one or both of query-point movement technique [8], and axis re-weighting technique [9]. In query-point movement technique, the Rocchio's formula [8] has been frequently used to move the ideal query point towards the positive example (PE) and away from the negative example (NE). In axis-reweighting technique, the main goal is to give more importance to features according to which example images are close to each other, and less importance to other features.

Some researchers consider RF as a classification problem where sample images provided by the user are used to train a classifier that is then used to classify the database into relevant images and irrelevant ones. . In [14], the authors propose a Bayesian model that supports image classes that assign a high membership probability to PE images and penalize classes that assign a high membership probability to NE images. In [7], a Bayesian model is used in which the distribution of relevant images is estimated and simultaneously the probability of retrieving irrelevant images is minimized. [12] presents a Bayesian classifier in which PE is used to estimate the Gaussian distribution that represents the class of sought images, while NE is used to modify the ranking of the retrieved candidates. Support vector machines (SVM) have been used in [13]. Considering RF as a classification problem is not without problem. First, because, in image retrieval, human subjectivity makes that the same image may belong to different classes. Second, because classification sometimes doesn't allow have a ranking of the retrieved images, while this may be necessary for some applications.

Other people consider RF as a learning problem in which samples fed back by the user are used to train a model that is then used for retrieval. Artificial neural networks have been used by Laaksonen et al. [6] who use Self-Organizing Map (SOM) to measure similarity between images. The PicHunter system [1] uses a Bayesian framework to predict what target image users want, given the action they undertook. The major problem

0-7803-8554-3/04/$20.00 ©2004 IEEE.

of learning techniques is the lack of data. Indeed, users usually provide a small number of images in retrieval process, while the algorithms need a large number of samples for training. Later on, some researchers considered RF as a distance optimization problem. In [3], Ishikawa et al. calculate the parameters that allow to find the ideal query, to weight the features, and to transform the feature space. This model has been improved in [10] by representing features separately, which allows to weight features and their components, and to alleviate the problem of lack of data. In [4], a further improvement is achieved by extending those models to support queries with positive and negative examples. In addition to the problem of lack of data, formulating RF as a distance optimization problem limits its capacities of modeling. In this paper, we consider RF as a probability optimization problem and this presents some advantages as we show in the next section.

3. RELEVNCE FEEDBACK PRINCIPLE AND FORMULATION

Understanding the user's needs in image retrieval may be a very challenging task because of human subjectivity. However, if we can understand correctly these needs, then this could help appreciably meeting them. This is especially true in what concerns the identification of the importance the user assigns to each feature and the adoption of the similarity measures that correspond to his judgement. This idea has been demonstrated by many previous research [4][3]. In the current work, we consider RF as a feature weighting problem whose ultimate objective is to find the query parameters and essentially find the importance of each feature. Furthermore, we want our model to support NE in addition to PE queries. Indeed, it has been observed that NE is very useful for query refinement since it allows to determine undesired images and thus to discard them [4]. Rather than a distance optimization problem, we formulate RF probabilistically, and this presents a multitude of advantages and opens the door for more modeling possibilities that may achieve a better feature weighting. It allows for a better and fuller understanding of data by clustering it into classes, choosing the probability law that better models each class, modeling the missing data, and supporting queries with multiple classes of PE and/or NE.

Before we give our formulation of the problem, let us make the following assumptions. 1) We suppose that we have a query made up of I features. 2) For each feature i, our query is constituted of several data samples X_{ni}, each of which is associated with a relevance degree π_{ni}.3) Our data may belong to PE or to NE: we specifically allow for two kinds of queries: those with PE only and those with PE and NE. 4) In each feature's space, PE constitute a number of classes and NE constitute a number of classes.

3.2. The case of PE only

In the case our query contains PE only, we consider that a given feature is important if it is salient, and that a salient feature must have a little dispersion among the classes of sample data. In terms of probabilities, we want to give more importance to features for which the query data have a high likelihood, and less importance to other features. Our parameters, namely class parameters Θ_{ik} (means, covariances and *a priori* probabilities in the case of Gaussian distributions) and features' weights α_i must maximize the query likelihood given by Equation (1) subject to the constraint $\sum_{i=1}^{I}(1/\alpha_i)=1$.

$$L=\prod_{i=1}^{I}\left(\prod_{n=1}^{N_i}\left[\sum_{k=1}^{K_i}P_{ik}P(X_{ni}\mid\Theta_{ik})\right]^{\pi_{ni}}\right)^{\alpha_i} \quad (1)$$

where for the i^{th} feature, X_{ni} is its n^{th} data, Θ_{ik} is the density parameters of its k^{th} class, and α_i the weigh of this feature. The two products in Equation (1) mean that we maximize the likelihood for each data of each feature. The sum represents a mixture of probability densities over the query classes. We derived this model in the case of Gaussian distributions; however other probability laws remain possible. The obtained results are as follows:

$$\mu_{il}=\frac{\sum_{n=1}^{N_i}\pi_{ni}P(\Theta_{il}\mid X_{ni})X_{ni}}{\sum_{n=1}^{N_i}\pi_{ni}P(\Theta_{il}\mid X_{ni})}\ ,\ P_{il}=\frac{\sum_{n=1}^{N_i}\pi_{ni}P(\Theta_{il}\mid X_{ni})}{\sum_{n=1}^{N_i}\pi_{ni}}$$

$$\Sigma_{il}=\frac{\sum_{n=1}^{N_i}\pi_{ni}P(\Theta_{il}\mid X_{ni})(X_{ni}-\mu_{il})(X_{ni}-\mu_{il})^T}{\sum_{n=1}^{N_i}\pi_{ni}P(\Theta_{il}\mid X_{ni})}$$

$$\alpha_i=\sum_{j=1}^{I}\sqrt{\frac{f_j}{f_i}}\ \text{ s.t. }\ f_i=-\sum_{n=1}^{N_i}\pi_{ni}\log\sum_{k=1}^{K_i}P_{ik}P(X_{ni}\mid\Theta_{ik})$$

3.2. The case of PE and NE

In the case our query contains both PE and NE, we have to estimate three families of parameters: PE class parameters Θ_{ik}, NE class parameters Θ_{ih}, and features' weights α_i. PE class parameters are estimated from PE data exactly like in Section 3.1, and NE class parameters are estimated from NE data in a similar way. Now, let us explain how we estimate the features' weights α_i. We consider that a given feature is important if it distinguishes well between PE classes on a side and NE classes on the other side. In terms of probabilities, our weights must maximize the internal likelihood of each PE class and that of each NE class, and simultaneously maximize the discrimination between PE classes and NE classes. This can be achieved by calculating the weights that maximize the likelihood of Equation (6) where the numerator contains the internal likelihoods (that of PE data with respect to PE densities and that of NE data w.r.t. NE densities) and the denominator contains the cross likelihoods (that of PE data w.r.t. NE densities and that of NE data w.r.t. PE densities).

$$L = \prod_{i=1}^{I} \left(\frac{\left(\prod_{n=1}^{N_i} \left[\sum_{k=1}^{K_i} P_{ik} P(X_{ni} \mid \Theta_{ik}) \right]^{\pi_{ni}} \right) \left(\prod_{m=1}^{M_i} \left[\sum_{h=1}^{H_i} P_{ih} P(Y_{mi} \mid \Theta_{ih}) \right]^{\pi_{mi}} \right)}{\left(\prod_{m=1}^{M_i} \left[\sum_{k=1}^{K_i} P_{ik} P(Y_{mi} \mid \Theta_{ik}) \right]^{\pi_{mi}} \right) \left(\prod_{n=1}^{N_i} \left[\sum_{h=1}^{H_i} P_{ih} P(X_{ni} \mid \Theta_{ih}) \right]^{\pi_{ni}} \right)} \right)^{\alpha_i} \quad (2)$$

After derivation, we obtain the following result:

$$\alpha_i = \sum_{j=1}^{I} \sqrt{\frac{f_j'}{f_i'}} \text{ s.t. } f_i' = R_i - A_i \text{ and}$$

$$A_i = \sum_{n=1}^{N_i} \pi_{ni} \log \sum_{k=1}^{K_i} P_{ik} P(X_{ni} \mid \Theta_{ik}) + \sum_{m=1}^{M_i} \pi_{mi} \log \sum_{h=1}^{H_i} P_{ih} P(Y_{mi} \mid \Theta_{ih})$$

$$R_i = \sum_{m=1}^{M_i} \pi_{mi} \log \sum_{k=1}^{K_i} P_{ik} P(Y_{mi} \mid \Theta_{ik}) + \sum_{n=1}^{N_i} \pi_{ni} \log \sum_{h=1}^{H_i} P_{ih} P(X_{ni} \mid \Theta_{ih})$$

4. EXPERIMENTS AND EVALUATION

4.1. Decontextualized evaluation

First, we evaluate the proposed model independently of image retrieval. Let us first recall the objectives we laid down for our algorithm: we want to give more importance to concentrated features (those with a high likelihood) in the case of PE only, and more importance to discriminant features in the case on PE and NE. Hence, we first need to define a criterion C_i that measures the concentration (or its inverse: the dispersion) in the case of PE, and the discrimination (or its inverse: the confusion) in the case of PE and NE. After that, we have to evaluate the weight our algorithm assigns to each feature according to the value of these criteria, i.e., $\alpha_i = f(C_i)$. In the case of PE only, to measure the dispersion of each feature, we adopt the following criterion used in Principal Component Analysis (PCA). The global dispersion of a set of classes is measured by the sum of the traces of their covariance matrices weighted by their a priori probabilities, i.e., $C = \sum_{k=1}^{K} P_k \Sigma_k$ [2]. In the case of PE and NE, to measure the confusion of each feature, we use a criterion used in Discriminant Analysis (DA) that we modify to fit in our context. Our criterion $C = tr(S_w) / tr(S_b)$ where S_w is the within class dispersion and S_b is a matrix that measures the discrimination between PE classes and NE classes

In both cases, we have conducted several experiments by varying many parameters such as the number of features, their dimensions and the number of data and clusters. In each experiment, we randomly generate Gaussian data, calculate the corresponding criteria, calculate the features' weights, and finally evaluate the weights according to the criteria. We conducted experiments with different numbers of features and the conclusions were almost the same. Thus, due to the limitation in space, we limit ourselves to the curves in the case of 2 and 3 features.

In the case of PE only, we have 2 features in the first experiment. We set the data of the first feature so that its dispersion = 40 and vary that of the second so that its dispersion increases each time; then, we observe the weights that our algorithm attributes to each of them. In the higher part of Figure 1, we represent the curves of the two weights α_1 and α_2 according to the second feature's dispersion. The main observation we make is that when the dispersion of feature 2 increases, its weight decreases, and this corresponds exactly to our objectives. Furthermore, in spite of the fact that the first feature's dispersion is constant (C_1=40), its weight α_1 increases when the dispersion of the second feature increases. This is due to the constraint we put on feature weights. Finally, the two curves $\alpha_1 = f(C_2)$ and $\alpha_2 = f(C_2)$ meet in the neighborhoods of C_2=40 and $\alpha_1 = \alpha_2 = 2$. This point corresponds to the case where the two features have almost the same dispersion, and hence it is natural that our algorithm gives them the same weight. In the second experiment, we have 3 features where we set the data of the first feature and vary those of the second and the third. The lower part of Figure 1 shows the weights α_1, α_2 and α_3 as functions of C_2 and C_3. In short, α_1 increases whatever the dispersion of one or both of the other features increases; α_2 decreases when C_2 increases, and simultaneously increases when C_3 increases; and finally α_2 has a behavior symmetric to that of α_2.

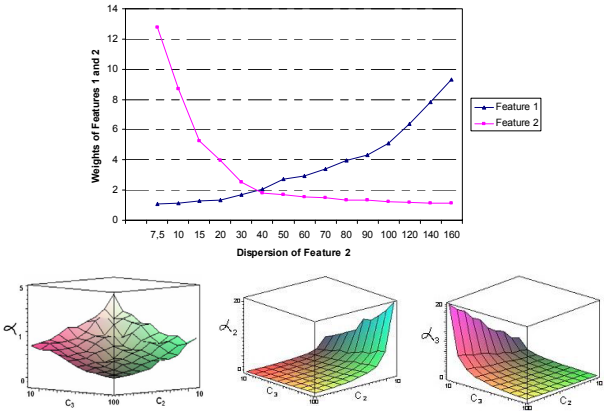

Figure 1. Feature weights in the case of PE.

In the case of PE and NE, we performed similar experiments. In the first one, we set the confusion of the first feature to 1.1 and vary the data of the second so that its confusion increases each time. In the second experiment, we have 3 features where the confusion of the second and the third are varying. The weight curves are given in Figure 2. The observations we make are similar to the case of PE (with confusion at the place of dispersion).

4.1. Evaluation of the overall engine

The described algorithm was implemented in our image retrieval engine "Atlas". Several visual features are used to describe images including color from the L*a*b* space, cooccurrence features (mean, variance, contrast,

energy, entropy, correlation, contrast, homogeneity, cluster shade

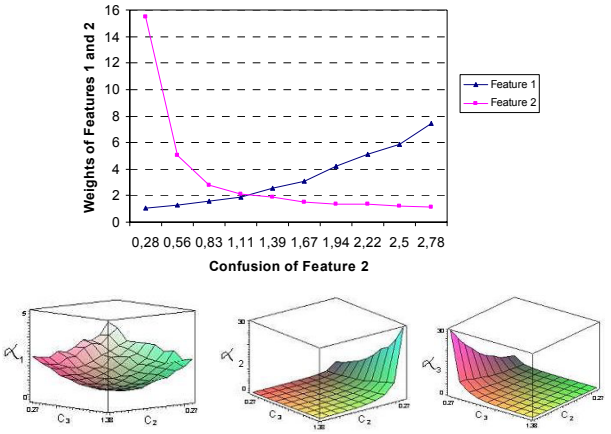

Figure 2. Feature weights in the case of PE and NE.

and cluster prominence), and orientation of edge pixels. In order to generate enough data for training, we oversample our images by first segmenting each one into a set of regions, then each region is further decomposed into subimages that constitute data for our algorithm. In order to evaluate the performance of our retrieval engine, we first constitute a collection of several thousands of images including free images from the CalPhotos collection, images from the Penn State University, as well as some thousands of free images collected from the Web. We want to perform a comparison between our engine and the one described in [4] which is based on distance optimization. To do so, we ask our users to use each of the engines and try to locate images belonging to different categories (such as horses, bridges, and buses), then to annotate each retrieved image with a relevance score among: relevant (+1), somewhat (0) and irrelevant (-1). The average scores are given in Table 1 where the proposed engine is noted "Prob" and that of [4] "Dist".

Table 1. Average percentage of relevant images retrieved.

Engine	**Prob**			**Dist**		
Category	+1	0	-1	+1	0	-1
Birds	61%	39%	00%	47%	32%	21%
Bridges	72%	18%	10%	50%	35%	15%
Buses	95%	05%	00%	71%	25%	04%
Dinosaurs	100%	00%	00%	96%	04%	00%
Flowers	79%	15%	06%	68%	10%	22%
Horses	77%	13%	12%	59%	25%	16%
Leaves	80%	17%	03%	84%	13%	03%
Lighthouses	55%	25%	20%	42%	20%	38%
Sunsets	75%	16%	09%	73%	16%	11%
waterfalls	64%	26%	10%	43%	37%	20%
Average	**76%**	**17%**	**07%**	**63%**	**22%**	**15%**

We see from Table 1 that the proposed method outperforms in general. This can be seen on the average number of noise (irrelevant) images which passes from 15% in Dist to 7% in Prob, while that of relevant images grows from 63% to 76%. Finally, we notice that the improvement is more noticeable for some categories, especially those with multi-class images (e.g., Horses).

5. CONCLUSION

In this paper, we presented a feature weighting algorithm that we applied to RF in CBIR. The problem is formulated as a probabilistic optimization problem with PE and NE, and was evaluated separately and in IR context.

Acknowledgement. The completion of this research was made possible thanks to NSERC and Bell Canada's support through its Bell University Laboratories R & D program.

11. REFERENCES

[1] I. J. Cox, et al. The Bayesian Image Retrieval System, PicHunter: Theory, Implementation, and Psychophysical Experiments. IEEE Trans. On IP, 9(1), 2000.

[2] K. Fukunaga. Statistical Pattern Recognition. Academic Press, San Diego, CA, Second Edition, 1990.

[3] Y. Ishikawa, et al. Mindreader: Querying Databases Through Multiple Examples. 24th Intl. Conf. on VLDB, New York, 1998.

[4] M.L. Kherfi, et al. Combining Positive and Negative Examples in Relevance Feedback for Content-Based Image Retrieval. Journal of Vis. Comm. and Image Rep., 14(4), 2003.

[5] M.L. Kherfi, et al. Image Retrieval from the World Wide Web: Issues, Techniques, and Systems. ACM Computing Surveys, 2004.

[6] J. Laaksonen, et al. PicSOM: Self-Organizing Maps for Content-Based Image Retrieval. Intl. Joint Conf. on N.N., Washington DC, 1999.

[7] C. Meilhac and C. Nastar. Relevance Feedback and Category Search in Image Databases. IEEE Intl. Conf. on Multimedia Comp. and Systems, Florence, Italy, 1999.

[8] J.J.Jr. Rocchio. Relevance Feedback in Information Retrieval. In The Smart System-Experiments in Automatic Document Processing, Prentice Hall, 1971.

[9] Y. Rui, T.S. Huang, and S. Mehrotra. Content-Based Image Retrieval with Relevance Feedback in MARS. In IEEE ICIP, Santa Barbara, California, 1997.

[10] Y. Rui and T.S. Huang. Optimizing Learning in Image Retrieval. In IEEE CVPR, Hilton Head, NC, 2000.

[11] G. Salton. Relevance Feedback and the Optimization of Retrieval Effectiveness. In The Smart System-Experiments in Automatic Document Processing, Prentice Hall, 1971.

[12] Z. Su, et al. Relevance Feedback in Content-Based Image Retrieval: Bayesian Framework, Feature Subspaces, and Progressive Learning. IEEE Trans. On IP, 12(8), 2003.

[13] S. Tong, and E. Chang. Support Vector Machine Active Leaning for Image Retrieval. ACM Multimedia Conf., Ottawa, Canada, 2001.

[14] N. Vasconcelos and A. Lippman. Learning from User Feedback in Image Retrieval Systems. In Neural Information Processing Systems, Denver, Colorado, 1999.