Intelligent Interfaces for Content-Based Retrieval of Images and Designs from Digital Databases

Geert Caenen
ESAT-PSI
KULeuven
B-3001 Leuven, Belgium
geert.caenen@esat.kuleuven.ac.be

Eric J. Pauwels
PNA 4
CWI
Amsterdam SJ 1098, The Netherlands
eric.pauwels@cwi.nl

Abstract

We outline the architecture of a CBIR-interface that supports intuitively transparent user-feedback and database navigation. The feedback is expressed directly in terms of images (rather than features) by singling out both positive and negative examples. A parametric inference engine is introduced to translate this input into a relevance-probability model.

1. Introduction and Motivation

The explosive growth of digital multi-media repositories has highlighted the need for accurate and efficient *content-based image retrieval* (CBIR), which turns out to be a formidable problem. A large part of the challenge is due to the fact that there is no canonical way to capture the visual *content* that is encapsulated in an image. Indeed, the definition of content is intricately tied up with the underlying visual appreciation and goals of the user, and will consequently vary from occasion to occasion.

It therefore seems foolhardy to attempt full automation of content-based searches and the most successful CBIR search-engines are the ones that *keep the human in the loop* by regularly requesting his feedback. Hence the interest in intelligent interfaces that prompt the user for feedback and then try and capitalize on it to expedite the search-action. More precisely, the feedback is harnessed to estimate for each image in the database the *likelihood of its relevance to the user,* whereupon the most promising candidates are displayed for further inspection and feedback. This procedure is then iterated as often as necessary to locate the target image. The essence of the search-procedure can therefore be summarised as a *sample-feedback-update* loop [1]:

1. Using some internal model of the user's preferences, the search-engine draws a representative **sample** from the database and presents this selection to the user for inspection;
2. Upon inspection the user provides some **feedback**, thereby indicating which images are particularly relevant with respect to his goals;
3. The system uses this feedback to **update** its internal model of the user's preferences and returns to step 1.

Modes of relevance feedback The specific form in which feedback is solicited varies from interface to interface. In the earliest versions (eg. QBIC), the user could express perceptual preferences by adjusting sliders that govern the relative weights of pre-specified image-features. However, it quickly turned out that this was not very accommodating for several reasons. First of all, it is difficult to clearly visualise what the precise impact of different parameter-combinations is. Secondly, to keep the interface manageable the number of sliders must be kept to a minimum, which entails that the user can only interact with a small number (5-10 say) of hard-wired features. Given that the typical number of features easily exceeds 100, this is a severe restriction.

To circumvent these problems, the new generation CBIR-interfaces, of which PicHunter [1] is a prime example, have shifted their focus from *features* to *images*: The user is no longer prompted to specify individual feature values or

[1]The reader is invited to have a look at the accompanying movie-clip at *http://www.esat.kuleuven.ac.be/∼pauwels/parissxl.htm*

weights, but can directly express his (global) preference of one image relative to the rest. Clearly, compared to the original feature-centered interaction, this type of interaction is perceptually more transparent and appealing.

It is this model of *natural* interaction that we will adhere to in this paper. More precisely, at every stage of the search-procedure, the user is shown a collection of images and simply indicates which of them are particularly relevant or irrelevant to his search. The underlying *inference engine* will use this simple input to estimate a probabilistic model of the user's preferences and will bias the next sample accordingly. How to design such an inference engine is the main topic addressed in this paper.

Probabilistic Relevance Measure In the scenario outlined above, the main role of the interface is to obtain as good an estimate of relative relevance of images as possible. The most straightforward way to model the fuzzy state of knowledge about the user's preferences, is for the interface to assign to every image $I_j \quad (j = 1, \ldots, N)$ in the database a **relevance probability** $p(I_j)$ that reflects the *current* estimate of relevance. Hence, $p(I) \approx 1$ means that in its current estimate, the image I is estimated to be highly relevant, whereas $p(I) \approx 0$ expresses the opposite. As pointed out earlier, this probability will be time-dependent since it is refined as more examples and counter-examples are accrued during the search-procedure. If necessary, we will clarify this time-dependence by explicitly referring to the iteration-step t when mentioning the relevance probability $p^{(t)}(I)$.

Clearly, unless we have prior knowledge, initially every image in the database is considered equally likely, and as a consequence the initial estimate $p^{(0)}$ for the probability measure is uniform:

$$p^{(0)}(I_j) = 1/N \qquad \text{for every } j = 1, \ldots, N.$$

However, as gradually more information about the user preferences becomes available (through the use of relevance feedback), the probability measure will change: it will increase in some locations but decrease in others. In short, the probability measure will start concentrating on regions in the database that seem promising. As a consequence, images in these regions are more likely to be sampled for display (and hence feedback).

In mathematical parlance, if I_T is the target image (let us assume there is a unique one), then we would like that as the number of iteration-steps t increases, $p^{(t)}(I_T) \longrightarrow 1$, while $p^{(t)}(I_j) \longrightarrow 0$ (for all $j \neq T$). Rephrased in these abstract terms, a feedback-driven retrieval session therefore looks like this:

```
1. Assign an initial relevance-probability
   $p^{(0)}(I_i)$ to every image in the database;

2. At iteration step $t$ use the probability
   distribution $p^{(t)}$ to generate a sample
   $S$ from the database;

3. Allow the user to generate feedback
   on the sample $S$;

4. Use this feedback to estimate the
   updated relevance-probability $p^{(t+1)}$
   and go back to step 2.
```

Aim of paper Clearly the main problem in the iteration above is the 4th step: updating the relevance probability measure. The *aim* of this paper is to show how *logistic regression* promises to be a very simple and effective methodology that does exactly that. Compared to the more popular choice of Bayesian modelling, logistic regression enjoys two main advantages:

1. Being a *parametric* model, logistic regression is much more efficient in its use of the available data;

2. The *diagnostics* that form an integral part of regression models can be used to advantage when attempting automatic feature selection.

In the remainder of this paper, we will adhere to standard practice and assume that every image I_j can be represented by a K-dimensional (numerical) feature-vector $\mathbf{x}_j = (x_{j1}, x_{j2}, \ldots, x_{jK})$ so that a database that comprises N images can be represented as a $N \times K$ data-matrix, the j^{th} row representing the feature-vector of the j^{th} image I_j.

2. Interface lay-out

We will outline the lay-out of a simple generic interface that allows the user to set aside relevant images during the exploration of database, and to use this collection to steer the search into the required direction. In its simplest implementation this interface comprises two *display windows* and an *inference engine*. The former are used to display images for relevance feedback, while the latter (on which we elaborate in the next section) is invoked to translate the user's input into a probability measure. For the sake of clarity, we will discuss these components separately (for a schematic overview we refer the reader to the accompanying movie-clip).

At all times, the interface shows three display-windows, between which the user can move seamlessly:

- **Window 1: Sample display** This screen displays the by now standard matrix of images that are (initially randomly) sampled from the database and presented to the user for inspection. The user can select images that are deemed relevant whereupon they are copied to the *collection box* (see below). Images of no particular interest are simply ignored. Each time a "refresh button" is pressed, the sampling algorithm is activated and a new sample is generated for inspection. The sampling algorithm that is used to generate the new sample can be biased by the *inference engine* (cfr. section 3) to accommodate the preferences of the user.

- **Window 2: Collection box for relevant images** The second screen is used as a simple *collection box* in which there are two bins: one for the *positive feedback* (i.e. examples of similar or partially similar images), and one for the *negative feedback* evoked by strongly dissimilar images (also called *counter-examples*) that seem to run against our expectations. Whenever the user comes across an image he considers relevant in that respect, it is added to the collection box. This box can be inspected at all times, and images that — in the light of later additions — no longer seem particularly relevant, can be removed. The collection box should be thought of as reflecting the user's cumulative (qualitative) knowledge about the database. This information will be turned into more quantitative measures by the *inference engine* (see section 3).

3. Design of Inference Engine

3.1. Learning from relevance feedback: Probabilistic Framework

As explained above, the *collection box* is used to collect both *examples* and *counter-examples* of the sort of images the user deems relevant. The reason for collecting them is to be found in the fact that this information can be used to bias the next sample favorably so that the fraction of interesting retrieved images increases over time. To this end we have implemented an *inference engine* that uses these data to generate an estimate of where interesting images can be found.

In order to proceed we must set up a mathematical framework within which we can discuss the questions at hand. As pointed out before, each one of the images I_j in the database ($j = 1, 2, \ldots, N$) is represented by its K-dimensional feature-vector $\mathbf{x}_j = (x_{j1}, x_{j2}, \ldots, x_{jK})$. At every stage of the search-history, the user has inspected a (small) fraction of the database and has provided the system with feedback by making a number of positive and negative selections and putting them in the collection box as examples and counter-examples (see above). This can be formalized by saying that for the (say n) images in the collection-box $\mathcal{C}$ we have further information that is captured in the binary variable $y_i \quad (i = 1, \ldots, n)$ defined by

$$y_i = \begin{cases} 1 & \text{if image } \mathbf{x}_i \text{ is considered an example} \\ 0 & \text{if image } \mathbf{x}_i \text{ is considered a counter-example} \end{cases} \tag{1}$$

To get an operational mathematical model we assume that these feedback-variables are observations of a binary stochastic variable Y that assigns a *predicted relevance output* (0 or 1) to every possible feature-vector $\mathbf{x} \in I\!\!R^K$. Put differently, give its current state of knowledge based on the examples and counter-examples gathered in the collection box, $Y(\mathbf{x})$ is what the system predicts will be the user's relevance judgement (0 or 1) if the latter was asked to judge an image with feature-vector $\mathbf{x}$.

Since Y is a *binary* stochastic variable we can model it using a simple *Bernoulli* distribution (i.e. a *binomial distribution* with a single repetition) with success-probability p:

$$Y \sim \text{Ber}(p) \quad \text{i.e.} \quad \begin{cases} P(Y=1) = p \\ P(Y=0) = 1-p \end{cases} \tag{2}$$

The dependence of Y on the image-features $\mathbf{x}$ is captured by equating the binomial success-probability p to the relevance probability $p(\mathbf{x})$ introduced in section 1. As a consequence, if $p(\mathbf{x}) \approx 1$, then $Y(\mathbf{x}) = 1$ with a high probability and there is little uncertainty about the outcome. Similarly, if $p(\mathbf{x}) \approx 0$ then $Y(\mathbf{x}) = 0$ with a high probability. However, if $p(\mathbf{x}) \approx 0.5$ the predicted relevance outcome Y takes the values 0 and 1 with almost equal odds, indicating that for an image with feature-values $\mathbf{x}$ it is difficult to predict the user's reaction.

To finalize this mathematical setup we need to specify how the relevance-probability $p(\mathbf{x})$ depends on the feature-values $\mathbf{x}$. Without loss of generality we can assume that this dependence takes the form of a *logistic regression model*:

$$\log \frac{p(\mathbf{x})}{1-p(\mathbf{x})} = f(\mathbf{x}) \tag{3}$$

The left-hand side is called the logit-ratio of p and is introduced to transform the value of p, which is constrained to the interval $[0,\ 1]$, to a quantity that ranges over $I\!\!R$ and is therefore easier to model with an appropriate function $f(\mathbf{x})$. In standard uses of logistic regression the function f is taken to be linear, but we have opted for a quadratic version for reasons that will be explained in section 3.2.

Equation (3) completes our mathematical framework for modeling user's relevance feedback. Indeed, if one knows the explicit form of the function f, then given an arbitrary image with corresponding feature-vector $\mathbf{x}$, one can compute the relevance probability $p(\mathbf{x})$ by inverting eq.(3) to obtain:

$$p(\mathbf{x}) = \frac{\exp(f(\mathbf{x}))}{1+\exp(f(\mathbf{x}))} \tag{4}$$

Plugging this value into eq.(2) yields a complete specification of the stochastic variable $Y(\mathbf{x}) \sim Ber(p(\mathbf{x}))$ that predicts whether or not the user will consider the image relevant. Clearly, if we have a good model for user relevance (which will therefore perform well in proposing new relevant images), then we expect $Y_i \equiv Y(\mathbf{x}_i)$ to correspond closely to the actually observed relevance y_i for the images i in the collection box $\mathcal{C}$. Hence, within the constraints of this framework, finding a good model for relevance feedback boils down to specifying an appropriate function f. Recall that this function is user-dependent and changes during the search process as more information is accrued.

3.2. Quadratic model for univariate covariates

Rephrased in the terminology introduced above the problem of the *inference engine* can be summarized as follows:

1. At every stage in the search process, the user provides the system with feedback that is summarized as a collection of examples ($y_i = 1$) and counter-examples ($y_i = 0$) in the collection box ($\mathcal{C}$);

2. Based on this information we need to estimate the relevance probability $p(\mathbf{x})$, or equivalently the function $f(\mathbf{x})$, that will maximise the agreement between the predicted (Y_i) and observed (y_i) values for the relevance of the collected images $\mathbf{x}_i \in \mathcal{C}$;

3. Once a good model has been obtained, one can use it to bias the next sample drawn from the database.

Although, in general, relevance might depend on the full feature vector $\mathbf{x} = (x_1, x_2, \ldots, x_K) \in I\!R^K$:

$$p = p(\mathbf{x}) = p(x_1, x_2, \ldots, x_K),$$

it is clear that reliably modeling the full probability density $p(\mathbf{x})$ on such scant information will in general prove to be an intractable problem, so we do the next best thing and model p as a function of each feature $x_k \quad (k = 1, \ldots, K)$ separately. Hence, in mathematical terms the problem boils down to this: the collection box contains a number of examples and counter-examples and for each single feature x_k (in what follows denoted x for short) we want to model the dependency of p on x.

The simplest case would be the one in which one can find a threshold value ($x^{(0)}$ say) that separates examples from counter-examples. Such information could then be fed back to the sampling procedure. However, in most cases the situation will be less clear-cut. The best one can hope for is that one can correlate the probability p with the x-values so that trends become visible and can be harnessed to improve the efficiency of the search.

The standard way to handle such a situation is to invoke a *logistic regression model*

$$\log \frac{p(x)}{1-p(x)} = f(x) \tag{5}$$

where the logit-ratio of p in the left-hand side is expressed as an appropriate function of the feature-value x. For the application we have in mind, we have opted for a *quadratic* logistic regression model:

$$\log \frac{p(x)}{1-p(x)} \quad = \quad \alpha x^2 + \beta x + \gamma; \tag{6}$$

The reason for the choice of a *quadratic* function might need some clarification. If relevance is (directly or inversely) proportional to the feature value, then a linear model will suffice (i.e. we can put $\alpha = 0$); however there obviously are situations where for instance, only medium feature-values are acceptable, while extreme values (both larger and smaller) are unacceptable. The quadratic model is the simplest model that can handle this sort of qualitative distinction. Introducing more sophisticated higher order models often does not pay as the feedback is only qualitative and rather coarse at that. Hence, quadratic models seem to strike an acceptable balance between flexibility and parsimony.

The parameters α, β and γ are determined by using *maximum likelihood estimation*, i.e. they optimize the probability of the actual configuration occurring. Loosely speaking, if we look up the x-value for each of the images in the collection box and then use eq.(6) to compute the probability p that they are in fact an example ($p > 1/2$) or counter-example ($p < 1/2$), then the parameters (α, β, γ) are chosen to optimize the *log-likelihood:*

$$\log L_n = \sum_{i=1}^{n} \left(y_i \log p_i + (1-y_i)\log(1-p_i)\right) \tag{7}$$

More details can be found in [2].

3.3. Combining univariate models

Up to now we have computed the probability for each individual feature. To extend this to a probability measure on the whole space we have to make further assumptions. The simplest is to assume independence of the different features so that the overall probability can be obtained by multiplying all the individual contributions (or summing the logarithms):

$$\log p(I) \equiv \log p(\mathbf{x}) = \sum_{k=1}^{K} \log p(x_k) \tag{8}$$

Although it is clear that the assumption of independence will not always be valid, it will be a worthwhile assumption in most cases. Furthermore, the structure of eq.(8)

immediately suggest an extension in which the contribution of the different features are weighted relative to their importance:

$$\log p(I) \equiv \log p(\mathbf{x}) = \sum_{k=1}^{K} w_k \log p(x_k) \qquad (9)$$

The weightfactors w_k can be determined automatically by considering the goodness-of-fit of each 1-dimensional regression model (see section 3.4).

3.4. Using regression diagnostics for feature selections

Selecting which of the pre-computed features are most informative with regard to the user's preferences, is crucial to achieving efficiency in retrieval. In this section we will argue that one of the additional boons of the proposed approach is that regression comes with a set of diagnostic tools that allow the system to automatically quantify the *goodness-of-fit* of the proposed model and select the features accordingly.

As explained above, the inference engine uses the selected examples and counter-examples to construct for each feature x a regression model that predicts the relevance of that feature in the overall judgement of the user. Standard regression diagnostic tools can then be invoked to judge the fit and predictive power of each such model. This helps us to gauge the success of the feature in predicting relevance and can therefore be used to narrow down the feature-set. More precisely, if for a particular feature x_i, the prediction of the fitted model (6) fails to square up with the relevance feedback from the user, this indicates that that particular feature x_i does not feature prominently in the perceptual appreciation of the user. Hence, a uniform sampling regime for that feature is advisable, as there is no reason to narrow its sampling-range.

Conversely, if for a different feature x_j, logistic regression yields a well-fitting model we can conclude that the feature plays an important role in the user's appreciation of the image, and we are well advised to bias the sampling-procedure as to favour feature-values x_j that have high p-value. That way, the fraction of relevant features in each new sample (as displayed on the *sample display*) will gradually increase.

The simplest way to measure the model's performance is by looking at the value of the maximum log-likelihood (7) that is achieved for each feature. Let λ_k denote this value for the k-th feature x_k. It is easy to check that $\lambda_k \leq 0$ and proportional to the model-fit, so that $\lambda_k = 0$ corresponds to a perfectly linearly separable model (i.e. examples and counter-examples are separated by a threshold) while more negative values are indicative of a progressively poorer fit. Hence the weights w_k in eq.(9) can be taken to be $w_k = \exp(-|\lambda_k|)$.

4. Discussion and conclusion

In this paper we have presented the outline of a CBIR-interface that supports natural forms of interaction and relevance feedback. The hallmark of our approach is that the user interacts directly with *images* rather than with arcane mathematical *features*. Furthermore, the interaction proceeds in an intuitively transparent fashion: the user simply clicks on images to single out examples and counter-examples. The real work is done by an *inference engine* that harnesses this information to iteratively refine its estimate of the underlying *parametric probability model* which predicts each image's relevance.

We have built a prototype of this interface for experimental purposes and attached it to a database of about 1500 images taken from collections of design and geology images. These images were chosen because of the complexity of their visual content that effectively defies keyword characterisation. Due to lack of space, it's not possible to include an extensive discussion; suffice it to say that preliminary experiments confirm our expectations: *(i)* Because of its intuitively transparent lay-out the threshold for use is very low, and even non-experts have little difficulty operating the system; *(ii)* The proposed feedback-procedures are both more flexible and powerful than the ones encountered in the more standard interfaces.

References

[1] I.J. Cox, M.L. Miller, T.P. Minka, T. Papathomas and P. N Yianilos: The Bayesian Image Retrieval System, *PicHunter*: Theory, Implementation and Psychophysical Experiments. *IEEE Trans. Image Processing, Vol.9, No.1, January 2000.*

[2] D.W. Hosmer and S. Lemeshow: *Applied Logistic Regression.* Wiley Series in Probability and Mathematical Statistics, 1989.

[3] Greet Frederix, Geert Caenen and Eric J. Pauwels: *PARISS: Panoramic, Adaptive and Reconfigurable Interface for Similarity Search.* Proc. of ICIP 2000, International Conference on Image Processing. Vol.III, pp. 222-225. Vancouver, September 2000.

[4] S. Santini and R. Jain: *Beyond Query by Example.* Proceedings of ACM Multimedia'98, Bristol, UK.