Chapter 8

Long-term learning

We have seen in the previous chapter that, in order to enable rapid convergence to the desired target image, good retrieval systems must know how to *integrate* information provided by the user over the entire retrieval session. In this chapter, we argue that this ability to learn from user interaction must also occur at longer time scales. In particular, retrieval systems should be able to develop internal representations of concepts that are likely to be of interest to their users.

Some of these representations may be hard-coded into the retrieval system from the start, i.e. it may contain modules specialized on the recognition of certain concepts that are required for semantic image understanding. Examples include detection of faces and people [148, 170, 45], face and gender recognition [112, 113], semantic scene classification [173, 182, 191, 186, 47], or even image classification according to form or function (e.g. graphics vs. photographs [6]). Since the design of such modules is usually difficult and, when feasible, requires large amounts of expert knowledge and training, they are likely to have economic justification only for visual concepts that are known to be of interest to most users.

This leaves out the majority of the concepts that are of interest to each individual user. For example, while it is unlikely that there will ever be sufficient economic pull to build a detector for the bluefinch shown in Figure 8.1, images of this bird may be among the top choices for a particular bird lover. Furthermore, users do not always require full-fledged

Figure 8.1: A bluefinch.

semantic recognition capabilities. E.g. while designing a generic dog recognizer is a daunting task, detecting the particular dog of an individual user may be a much simpler problem. In fact, the retrieval examples from the Columbia database show that it is relatively easy to find complicated objects in a relatively large database without requiring high-level knowledge of what an object is.

The really difficult task is *generalization*, e.g. the ability to classify the object of Figure 8.1 as a bird even though you may never have seen it before. While desirable, it is not clear that generalization will be required at all times and by all users. Instead, in most situations, users will probably be satisfied with the ability to train the retrieval system on the specific objects that are of interest to them. And, since users interested in particular visual concepts will tend to search for them quite often, there will be plenty of examples to learn from, by simply monitoring user actions. Hence, the retrieval system can build internal concept representations and become progressively more apt at recognizing them as time progresses. We refer to such mechanisms as *long-term learning* or *learning between retrieval sessions*, i.e. learning that does not have to occur on-line, or even in the presence of the user.

In addition to integrating information over time (learning), a good retrieval system should also be able to integrate information from diverse sources. Besides enabling more sophisticated queries (e.g. the ability to fuse face and speech recognition would allow queries for "the video clip where the president talks about the budget"), this ability to integrate information from diverse sources can significantly simplify the retrieval process. For example, any text annotations that may be available with the database can be used to constrain the visual search to the classes that are semantically relevant to the query.

In this chapter, we show that Bayesian retrieval can be easily extended to multiple content modalities and design a Bayesian long-term learning mechanism that complements the inference procedures discussed in Chapter 7. In particular, we show that the Bayesian approach scales well with the number of modalities to be integrated, has intuitive interpretation, and leads to extremely simple algorithms. Experimental evaluation on the Corel database demonstrates that it is possible to learn various types of visual concepts with surprising accuracy.

8.1 Probabilistic model

We start by extending the Bayesian retrieval model to multimodal content sources. First, we should notice that, conceptually, the problem does not change. The only difference is that, instead of a single feature space $\mathcal{X}$, we now have as many features spaces as the number of content modalities or *attributes* under consideration. Denoting the individual feature spaces by $\mathcal{X}_i$, we can combine them into a *meta feature space* $\mathcal{M}$ by simply concatenating the individual feature vectors. I.e. if $\mathbf{x}_i$ is a feature vector in $\mathcal{X}_i$, then

$$\mathbf{m} = \{\mathbf{x}_1, \dots, \mathbf{x}_m\},$$

where m is the total number of attributes, is a feature vector in $\mathcal{M}$. If we define a query $\mathbf{m}_i$ as a collection of N_i feature vectors $\mathbf{m}_i = \{\mathbf{m}_{i,j}\}_{j=1}^{N_i}$ and a retrieval session as a sequence of queries $\mathbf{m}_1^t = \{\mathbf{m}_i\}_{i=1}^t$, all the results that were previously derived for $\mathcal{X}$ are also valid for $\mathcal{M}$. The only difference is that we now talk about *content classes* instead of image classes.

While conceptually the two problems are identical, in practice there is a substantive difference: the dimension of $\mathcal{M}$ can be significantly higher than that of $\mathcal{X}_i$. While this makes the estimation of joint densities infeasible in $\mathcal{M}$, it is usually true that the different modalities are associated with processes that can be considered independent. For example, it is reasonable to assume that a set of features used to characterize speech is independent of the visual features extracted from images of the speaker. This leads to the following independence assumption.

Assumption 7 *Given the target content class, the different content modalities are independent*

$$P_{\mathbf{M}|Y}(\mathbf{m}|i) = \prod_{k=1}^{m} P_{\mathbf{M}_k|Y}(\mathbf{m}_k|i).$$

For retrieval, the user instantiates a subset of the m modalities. The particular process of instantiation depends on the nature of the content associated with each attribute. While text can be instantiated by the simple specification of a few keywords, pictorial attributes are usually instantiated by example.

8.2 Incomplete queries

Of course, not all attributes need to be instantiated in all queries. Borrowing the terminology from the Bayesian network literature [127, 75, 76], we denote, for a given query, an index set $\mathbf{o}$ of *observed* attributes and an index set $\mathbf{h}$ of *hidden* attributes. E.g. if $m = 3$, attributes 1 and 2 are instantiated and attribute 3 is left unspecified then $\mathbf{o} = \{1, 2\}$ and $\mathbf{h} = \{3\}$. The likelihood of a query $\mathbf{q}$ is then given by

$$P_{\mathbf{M}|Y}(\mathbf{q}|i) = \sum_{\mathbf{q_h}} P_{\mathbf{M}|Y}(\mathbf{q_o}, \mathbf{q_h}|i), \tag{8.1}$$

where the summation is over all possible configurations of the hidden attributes[1]. Using Assumption 7 and the fact that $\sum_{\mathbf{x}} P_{\mathbf{X}|Y}(\mathbf{x}|i) = 1$,

$$\begin{aligned} P_{\mathbf{M}|Y}(\mathbf{q}|i) &= \sum_{\mathbf{q_h}} P_{\mathbf{M_o}|Y}(\mathbf{q_o}|i) P_{\mathbf{M_h}|Y}(\mathbf{q_h}|i) \\ &= P_{\mathbf{M_o}|Y}(\mathbf{q_o}|i) \sum_{\mathbf{q_h}} \prod_{k \in \mathbf{h}} P_{\mathbf{M}_k|Y}(\mathbf{q}_k|i) \\ &= P_{\mathbf{M_o}|Y}(\mathbf{q_o}|i) \prod_{k \in \mathbf{h}} \sum_{\mathbf{q}_k} P_{\mathbf{M}_k|Y}(\mathbf{q}_k|i) \\ &= P_{\mathbf{M_o}|Y}(\mathbf{q_o}|i), \end{aligned} \tag{8.2}$$

i.e. the likelihood of the query is simply the likelihood of the instantiated attributes. In addition to being intuitively correct, this result is also of considerable practical significance. It means that the complexity of retrieval grows with the number of attributes specified by

[1]The formulation is also valid in the case of continuous variables with summation replaced by integration.

the user and not with the number of attributes known to the system, which can therefore be made arbitrarily large.

8.3 Combining different content modalities

While the Bayesian framework can integrate all types of content modalities, in this thesis we restrict our attention to the integration of visual attributes and text annotations. In particular, we consider retrieval sessions $\mathbf{m}_1^t = \{\mathbf{t}_1^t, \mathbf{x}_1^t\}$, composed of text ($\mathbf{t}_1^t$) and visual attributes ($\mathbf{x}_1^t$). Combining (7.7) with Assumption 7, assuming only positive examples, and disregarding the decay factor[2], we obtain

$$g^*(\mathbf{m}_1^t) = \arg\max_i \left\{ \log P_{\mathbf{X}_t|Y}(\mathbf{x}_t|i) + \log P_{\mathbf{T}_t|Y}(\mathbf{t}_t|i) + \log P_{Y|\mathbf{M}_1^{t-1}}(i|\mathbf{m}_1^{t-1}) \right\}. \tag{8.3}$$

This equation has several equally intuitive interpretations. The standard one is to consider $\log P_{Y|\mathbf{M}_1^{t-1}}(i|\mathbf{m}_1^{t-1})$ as a prior belief, for iteration t, that gives more weight to those classes that have performed well in the past and $P_{\mathbf{X}_t|Y}(\mathbf{x}_t|i)$ and $P_{\mathbf{T}_t}(\mathbf{t}_t|i)$ as the likelihoods of the current observations. Two alternative interpretations are however possible.

The first, and vision centric, is that the optimal class is the one which would best satisfy the visual query alone but with a prior consisting of the combination of the second and third terms. By instantiating text attributes, the user establishes a *context* for the evaluation of visual similarity that changes the system's prior beliefs about which class is most likely to satisfy the visual query. Or, in other words, the text attributes provide a means to *constrain* the visual search. The second, and text centric, is to consider the second term the likelihood function, with the combination of the first and the third forming the prior. In this interpretation, the visual attributes constrain what would be predominantly a text-based search.

Independently of the interpretation, the significance of (8.3) is that it illustrates one of the most attractive properties of the Bayesian retrieval formulation: because all models

[2] In order to simplify the presentation, in this chapter we ignore negative examples and decay factors. However, all results are extensible to the generic case and the extension is straightforward: simply modify the denominator of (7.7) in the same way as the numerator is changed and add α and $(1-\alpha)$ where appropriate.

speak the same *language* (the language of probabilities) it is relatively easy to combine the outputs of specialized modules into a *global* inference. Because, due to Assumption 7, we are relying on a simple model for the dependencies between the attributes, the integration is fairly simple (simply adding log-likelihoods). If more complex models were available, it would still be possible even though probably through a more complicated expression. Notice that the only fundamental requirement for the integration to be possible is that the different attributes can be modeled probabilistically. In the previous chapters, we have seen how this can be done for pictures. We next concentrate on the issue of representing text.

8.4 Text representation

The standard representation for text retrieval is the *vector space model* [154, 50] where each document is represented as a collection of *indexing terms*. An indexing term can be a word, a group of words, or a word stem[3] among others. The space of indexing terms can be seen as the observation space $\mathcal{Z}$ for text, and associated with each index term and each document there is a binary feature indicating if the term appears, or not, in the document. That is, the document is represented by a binary vector $\mathbf{t} = \{t_1, \ldots, t_L\}$, where $t_j = 1$ ($t_j = 0$) if the index term i appears (does not appear) in the document, and L is the total number of indexing terms known to the retrieval system. A query is simply a collection of index terms and is therefore represented in the same way.

The simplest possible retrieval model is to, in response to a query, extract from the database all the documents whose feature vectors match that of the query. This, however, does not take into account the fact that some terms are more relevant to the characterization of a document than others. A better approach is therefore to associate a weight with each component of the feature vector. Even though such weights are not always justified probabilistically [51], usually they are constrained to the interval $[0, 1]$ and therefore have a probabilistic interpretation.

Truly probabilistic representations are, however, popular in the text retrieval literature.

[3] A word stem [154] is a descriptor for a collection of words with similar semantics, e.g. different verb tenses, variations on a word by the introduction of suffixes, etc.

Among these the most commonly used is the so-called *naive Bayes* model [87] which assumes that the L binary features are independent, and models each feature as a Bernoulli random variable

$$P_{T_j|Y}(t_j|i) = \begin{cases} 1 - p_{i,j}, & \text{if } t_j = 0 \\ p_{i,j}, & \text{if } t_j = 1. \end{cases} \tag{8.4}$$

The probabilities $p_{i,j}$ can be seen as the weight vector for documents from class i.

Despite its simplicity, and after decades of research on more sophisticated alternatives, the naive Bayes model has proven difficult to beat [87]. In terms of the discussion above, it is equivalent to considering each possible index term as a different content attribute and relying on Assumption 7. Hence, the naive Bayes model fits naturally into the overall Bayesian retrieval formulation developed in this thesis and, because it is simple, we adopt it[4]. Combining Assumption 7 with (8.2) and (8.4), and taking logs we obtain

$$\log P_{\mathbf{T}|Y}(\mathbf{t}|i) = \sum_j \delta_{t_j,1} \log p_{i,j} + \sum_j \delta_{t_j,0} \log(1 - p_{i,j}),$$

where δ is the Kronecker delta function (2.3). When there are only positive examples

$$\log P_{\mathbf{T}|Y}(\mathbf{t}|i) = \sum_j \delta_{t_j,1} \log p_{i,j}. \tag{8.5}$$

8.4.1 Parameter estimation

There are several ways to estimate the parameters $p_{i,j}$. When an actual text document is available (as is usually the case with text retrieval), the estimates can be derived from frequency counts, i.e. $p_{i,j}$ is a function of the number of times that term j appears in document (or document class) i. While there are image retrieval scenarios in which a text document is associated with each image (e.g. web pages), this does not always hold. Furthermore, it has not yet been established that, when a free text document is available, visual retrieval will add any improvements to text-based retrieval. For these reasons, we concentrate on the situations in which a detailed textual description of the image content is not available. In such cases, the straightforward solution to the annotation problem is to use manual labeling, relying on the fact that many databases already include some form of

[4]Notice that this does not mean that other more sophisticated text models could not be used in the Bayesian retrieval formulation.

coarse image classification. For example, an animal database may be labeled for cats, dogs, horses, and so forth. In this case, it suffices to associate the term "cats" with t_1, the term "dogs" with t_2, etc and make $p_{i,1} = 1$ for pictures with the cats label and $p_{i,1} = 0$ otherwise, $p_{i,2} = 1$ for pictures with the dogs label and $p_{i,2} = 0$ otherwise, and so forth. In response to a query instantiating the "cats" attribute, (8.5) will return 0 for the images containing cats and $-\infty$ for those that do not. In terms of (8.3) (and associated discussion in section 8.3), this is a *hard constraint*: the specification of the textual attributes eliminates from further consideration all the images that do not comply with them.

Hard constraints are usually not desirable, both because there may be annotation errors and because annotations are inherently subjective. For example, while the annotator may place leopards outside the cats class, a given user may use the term "cats" when searching for leopards. A better solution is to rely on *soft constraints* where the $p_{i,j}$ are not restricted to be binary. In this case, the "cats" label could be assigned to leopard images, even though the probability associated with the assignment would be small. In this context, $p_{i,j}$ should be thought of as the answer to the question "what is the likelihood that users will instantiate attribute t_j given that they are interested in images from class i?". In practice, it is usually 1) too time consuming to define all the $p_{i,j}$ manually, and 2) not always clear how to decide on the probability assignments. A better alternative is to rely on learning.

8.4.2 Long term learning

Unlike the learning algorithms discussed in section 7.2, here we are talking about *long-term learning* or *learning across retrieval sessions*. The basic idea is to let users attach a label to each of the regions that are provided as queries during the course of the normal interaction with the retrieval system. For example, if in order to find a picture of a snowy mountain a user selects a region of sky, the user has the option of labeling that region with the word "sky" establishing the "sky" attribute. Learning then consists of estimating the probabilities $p_{i,j}$ from the example regions. For this we rely on the following assumption.

Assumption 8 *When, during retrieval, a user instantiates a text attribute t_j, the user is looking for images that contain regions similar to those previously provided as examples for* *that attribute.*

The assumption simply means that we expect users to be consistent, providing a basis for the estimation of the $p_{i,j}$. It states that the instantiation of attribute t_j is equivalent to complementing the visual component of the query with the examples that were previously stored for attribute t_j. Mathematically, the query $\{\mathbf{x}, t_i = 1\}$, where $\mathbf{x}$ is a collection of visual feature vectors, is equivalent to the query $\{\mathbf{x}, \mathbf{e}_j\}$ where $\mathbf{e}_j$ is the *example set* containing the example regions for attribute t_j, $\mathbf{e}_j = \{\mathbf{e}_{j,1}, \ldots \mathbf{e}_{j,K}\}$. Since, from Assumption 7 and (8.5),

$$\begin{aligned}
\log P_{\mathbf{X},T_j|Y}(\mathbf{x}, 1|i) &= \log P_{\mathbf{X}|Y}(\mathbf{x}|i) + \log P_{T_j|Y}(1|i) \\
&= \log P_{\mathbf{X}|Y}(\mathbf{x}|i) + \log p_{i,j},
\end{aligned}$$

this means that

$$\log P_{\mathbf{X}|Y}(\mathbf{x}|i) + \log p_{i,j} = \log P_{\mathbf{X}|Y}(\mathbf{x}|i) + \log P_{\mathbf{X}|Y}(\mathbf{e}_{j,1}, \ldots \mathbf{e}_{j,K}|i)$$

or

$$p_{i,j} = P_{\mathbf{X}|Y}(\mathbf{e}_{j,1}, \ldots \mathbf{e}_{j,K}|i).$$

From Assumption 2,

$$\begin{aligned}
\log p_{i,j} &= \sum_k \log P_{\mathbf{X}|Y}(\mathbf{e}_{j,k}|i) \\
&= \sum_{k,l} \log P_{\mathbf{X}|Y}(\mathbf{e}_{j,k,l}|i), \qquad (8.6)
\end{aligned}$$

where $\mathbf{e}_{j,k,l}$ is the l^{th} feature vector from example region k for text attribute j.

This expression is all that has to be computed in the learning stage. Notice that, because only the running sum of $\log P_{\mathbf{X}|Y}(\mathbf{e}_{j,k}|i)$ must be saved from session to session, there is no need to keep the examples themselves. Instead, it suffices to store one number per image class/attribute pair. Notice also that, since all the $\log P_{\mathbf{X}|Y}(\mathbf{e}_{j,k}|i)$ terms have to be computed for the queries in which the examples $\mathbf{e}_{j,k}$ are defined, there is no computational cost associated with the learning procedure itself; i.e., long-term learning is highly efficient in terms of both computation and memory.

Grounding the annotation model directly in visual examples also guarantees that the beliefs of (8.6) are of the same order of magnitude as those of (5.7), making the application of (8.3) straightforward. If different representations were used for annotations and visual attributes, one would have to define weighting factors to compensate for the different scales of the corresponding beliefs. Determining such weights is usually not a simple task.

There is, however, one problem with the example-based learning solution. While the complete set of examples of a given concept may be very diverse, individual image class models may not be able to account for all this diversity. In the case of "sky" discussed above, while there may be examples of sunsets, sunrises, and skies shot on cloudy, rainy or sunny days in the "sky" example set, particular image classes will probably not encompass all this variation. For example, as illustrated in Figure 8.2, images in the class "pictures of New York at sunset" will only explain well a fraction of the sunset examples. It follows that, while this class should receive a high rank with respect to "skyness," there is no guarantee that this will happen since it assigns low probability to a significant number of sky examples.

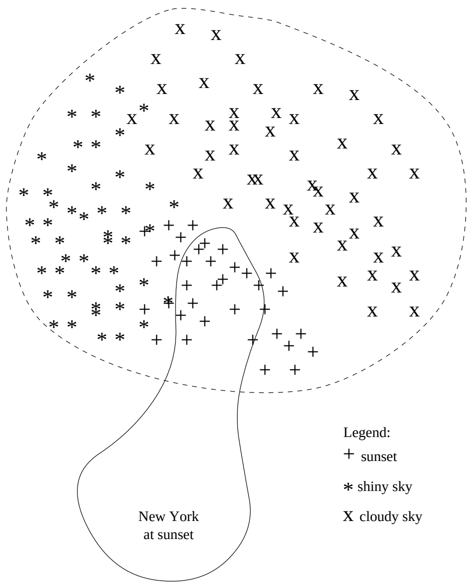

Figure 8.2: An image class may not encompass all the examples from a given attribute, even when the attribute is present. The solid line represents the density of the class "pictures of New York at sunset", "+" are examples of sunsets, "x" of shiny sky, "*" of cloudy skies, and the dashed line the overall density for "sky".

The fact is that most image classes will only overlap partially with broad concept classes like "sky". The problem can however be solved by requiring the image classes to explain well only a subset of the concept examples. One solution is to rank the examples according to their probability and apply (8.6) only to the top ones,

$$\log p_{i,j} = \sum_{r=1}^{R} \log P_{\mathbf{X}|Y}(\mathbf{e}_{j,k}^{(r)}|i), \tag{8.7}$$

where $\mathbf{e}_{j,k}^{(r)}$ is the example region of rank r and R a small number (10 in our implementation).

8.5 Experimental evaluation

The performance of a long-term learning algorithm will not be the same for all concepts that may need to be learned. In fact, the learnability of a concept is a function of two main properties: *visual diversity* and *distinctiveness* on the basis of local visual appearance. Diversity is responsible for misses, i.e. instances of the concept that cannot be detected because the learner has never seen anything like them. Distinctiveness is responsible for false positives, i.e. instances of other concepts that are confused with the desired one. Since the two properties are functions of the particular image representation, it is important to test the performance of the learner with concepts from various points in the diversity/distinctiveness space.

We relied on the Corel database to evaluate long-term learning and identified five such concepts: a flag, tigers, sky, snow, and vegetation. The flag is representative of computer graphics objects, such as logos, that tend to be presented with small variation of visual appearance and therefore are at the bottom of the diversity scale. Tigers (like most animals) are next: while no two tigers are exactly alike, they exhibit significant uniformity in visual appearance. However, they are usually subject to much stronger imaging transformations than logos (e.g. partial occlusion, lighting, perspective). Snow and sky are representative of the next level in visual diversity. Even though relatively simple concepts, their visual appearance varies a lot with factors like imaging conditions (e.g. shiny vs. cloudy day) or the time of the day (e.g. sky at noon vs. sky at sunset). Finally, vegetation encompasses a large amount of diversity.

In terms of distinctiveness, logos rank at the top (at least for Corel where most images contain scenes from the real world), followed by tigers (few things look like a tiger), vegetation, sky and snow. Snow is clearly the less distinctive concept, on the basis of local visual appearance, since large patches of smooth white surfaces are common in many scenes (e.g. clouds, white walls or other objects like tables, paper, etc.). The distribution of the five concepts on the diversity/distinctiveness space is shown in Figure 8.3.

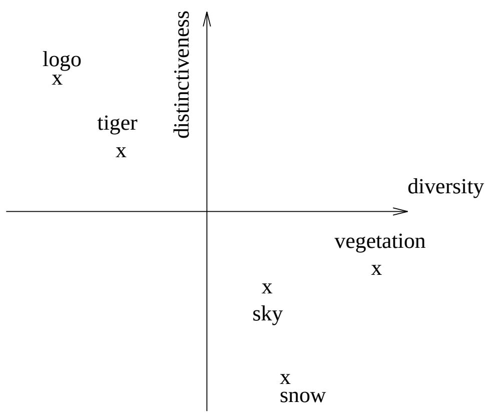

Figure 8.3: Distribution of the concepts in terms of diversity and distinctiveness.

In order to train the retrieval system, we annotated all the images in the database according to the presence or not of each of the five concepts. We then randomly selected a number of example images for each concept and manually segmented the regions where the concepts appeared. These regions were used as examples for learning. Concept probabilities were estimated for each image outside the training set using (8.7) and, for each concept, the images were ranked according to these probabilities. Figure 8.4 presents the resulting precision/recall curves for the five concepts. Retrieval accuracy seems to be directly related to concept distinctiveness: a single training example is sufficient for perfect recognition of the logo and with 20 examples the systems does very well on tigers, reasonably well on vegetation and sky, and poorly on snow. These are surprisingly good results, particularly if one takes into account the reduced number of training examples and the fact that the degradation in performance is natural for difficult concepts.

Performance can usually be improved by including more examples in the training set, as

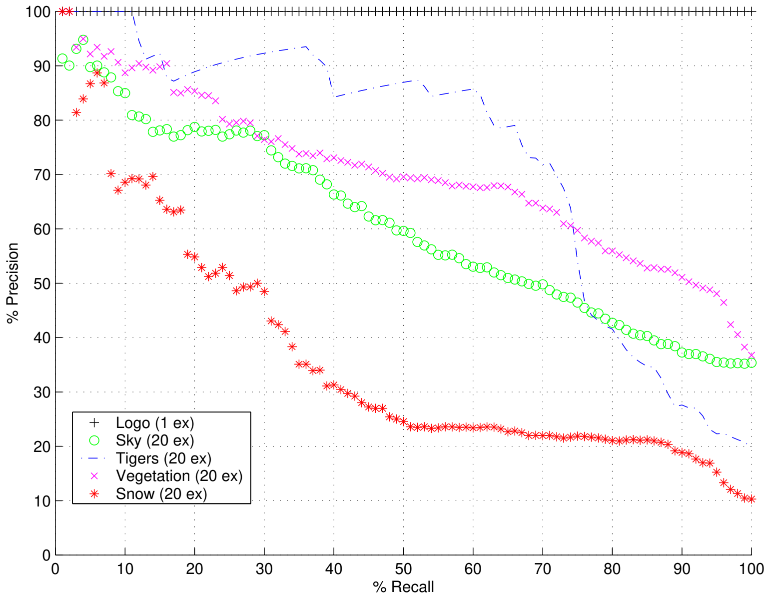

Figure 8.4: Performance of long-term learning. Precision/recall curves for the five concepts described in the text.

this reduces the concept diversity problem. This is illustrated in Figure 8.5, where we show the evolution of precision/recall as a function of the number of training examples for sky and tigers. In both cases, there is a clear improvement over the situation in which only one example is used. This result is particularly significant, since the one-example scenario is equivalent to the standard query-by-example paradigm. Under this paradigm, a user would try to retrieve images containing a given concept by providing the retrieval system with one example of that concept. As the figures clearly demonstrate, one example is usually not enough, and long-term learning does improve performance by a substantial amount. In the particular case of sky, it is clear that performance can be made substantially better than that of Figure 8.4 by taking more examples into account.

On the other hand, Figure 8.6 shows that more examples make a difference only when the performance is limited by a poor representation of the concept diversity, not distinctiveness. For snow, where the latter is the real bottleneck, providing more examples does not seem to make a difference.

Figures 8.7 to 8.11 show the top 36 matches for the five concepts. These figures illustrate

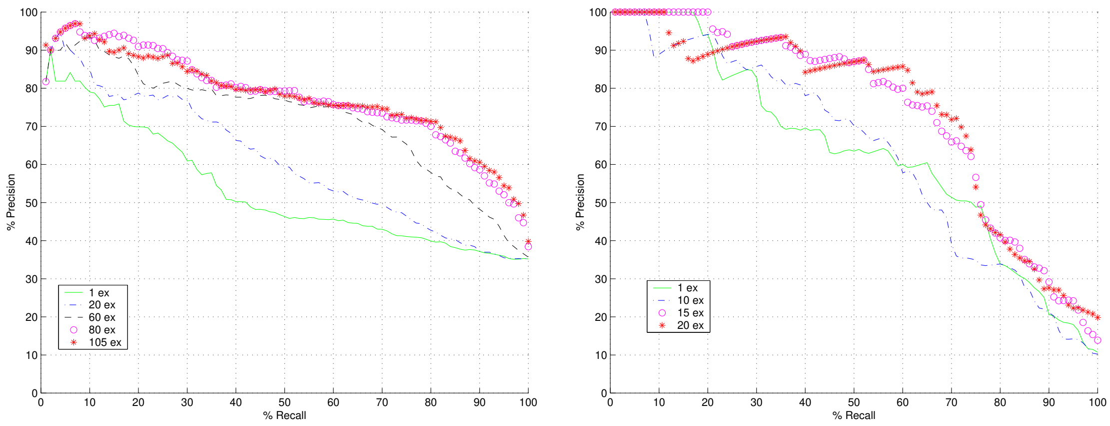

Figure 8.5: Evolution of precision/recall as a function of the training set size for sky (left), and tigers (right).

well how the long term learning mechanism is robust with respect to concept diversity, either in terms of different camera viewpoints, shading, occlusions, etc (e.g. tiger images) and variations in visual appearance of the concept itself (e.g. sky or vegetation images). In general, the errors are intuitive: for the logo retrieval is perfect; for sky and vegetation errors correspond to images that could have been labeled either way (e.g. images of a tiger or an owl with some trees on the background were labeled as not containing vegetation); and for snow errors tend to be images containing large smooth white surfaces (walls, flower, clouds, car hoods, etc.). The less intuitive errors happen for tigers, where the texture of some paintings is confused with a tiger, but are few.

The snow example actually illustrates one advantage of systems that learn by example: since users provide all the examples, they can develop an understanding of which concepts are easier to learn. In this case, because the training for snow consisted of smooth white image patches and all the errors contain such patches, it is not clear how the system could be trained to improve its ability to detect snow. Hence, a user could quickly realize that snow is a difficult concept for the system. This is indeed the case since distinguishing a patch of snow from a white wall requires high-level scene understanding abilities that the system does not possess.

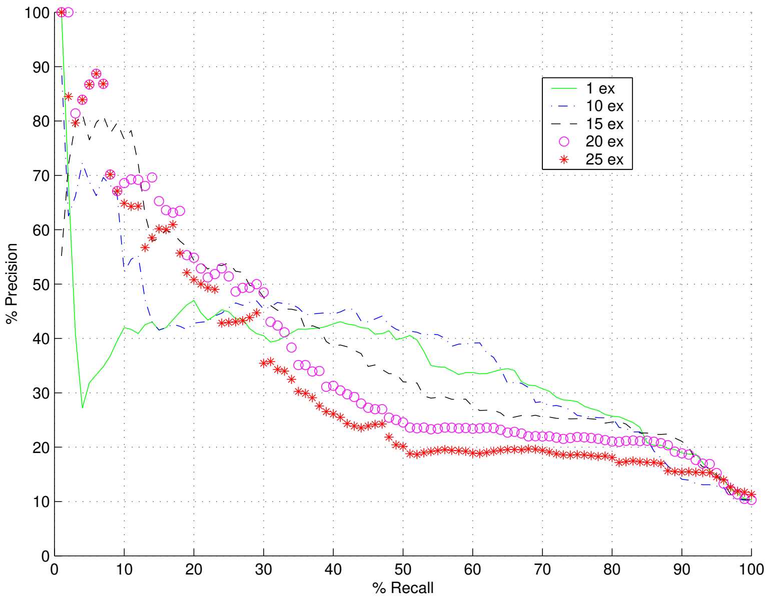

Figure 8.6: Evolution of precision/recall for snow.

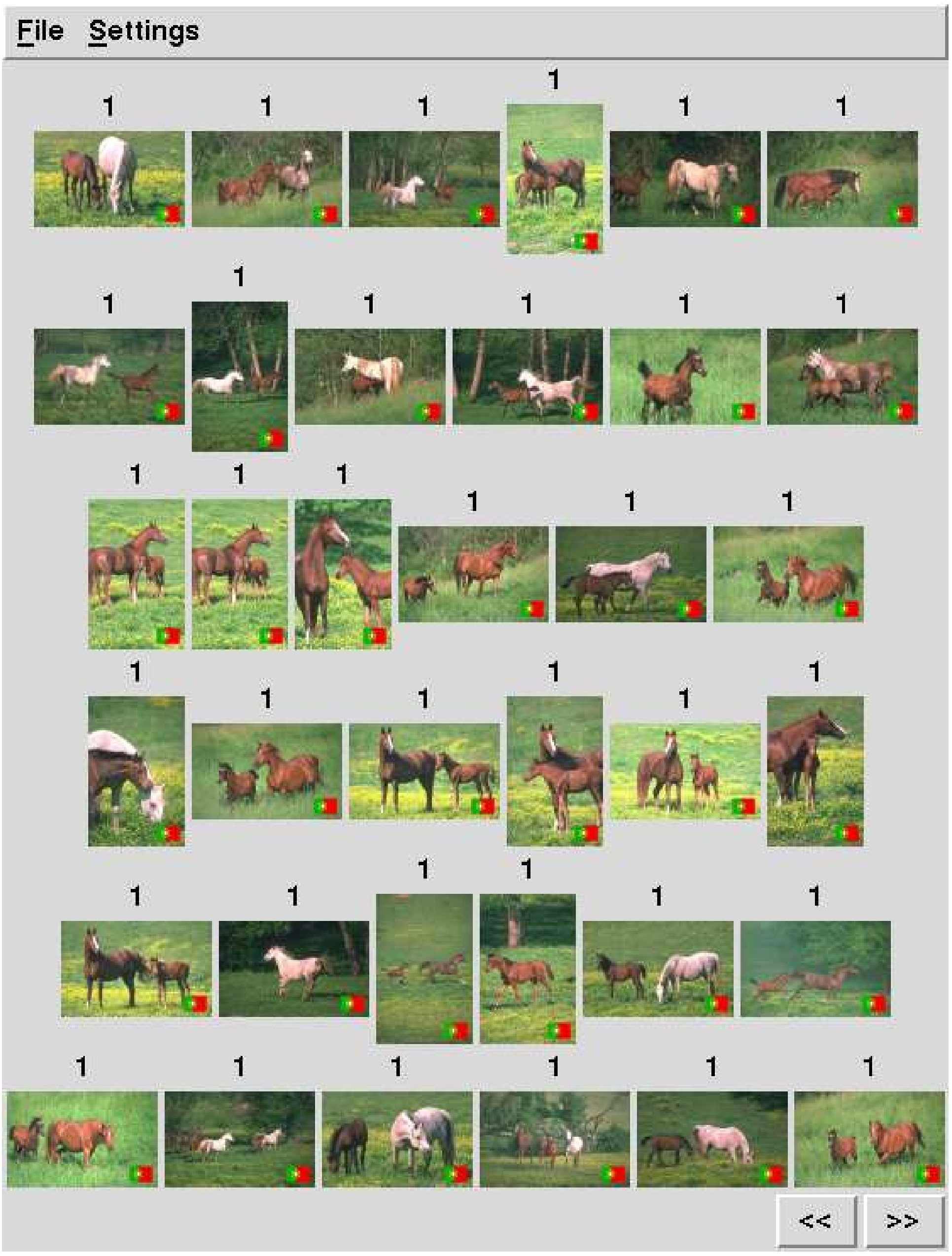

Figure 8.7: Top 36 matches for the flag concept. The number shown on top of each image indicates if the image was annotated as containing the concept (1) or not (0).

Figure 8.8: Top 36 matches for the tiger concept.

Figure 8.9: Top 36 matches for the sky concept.

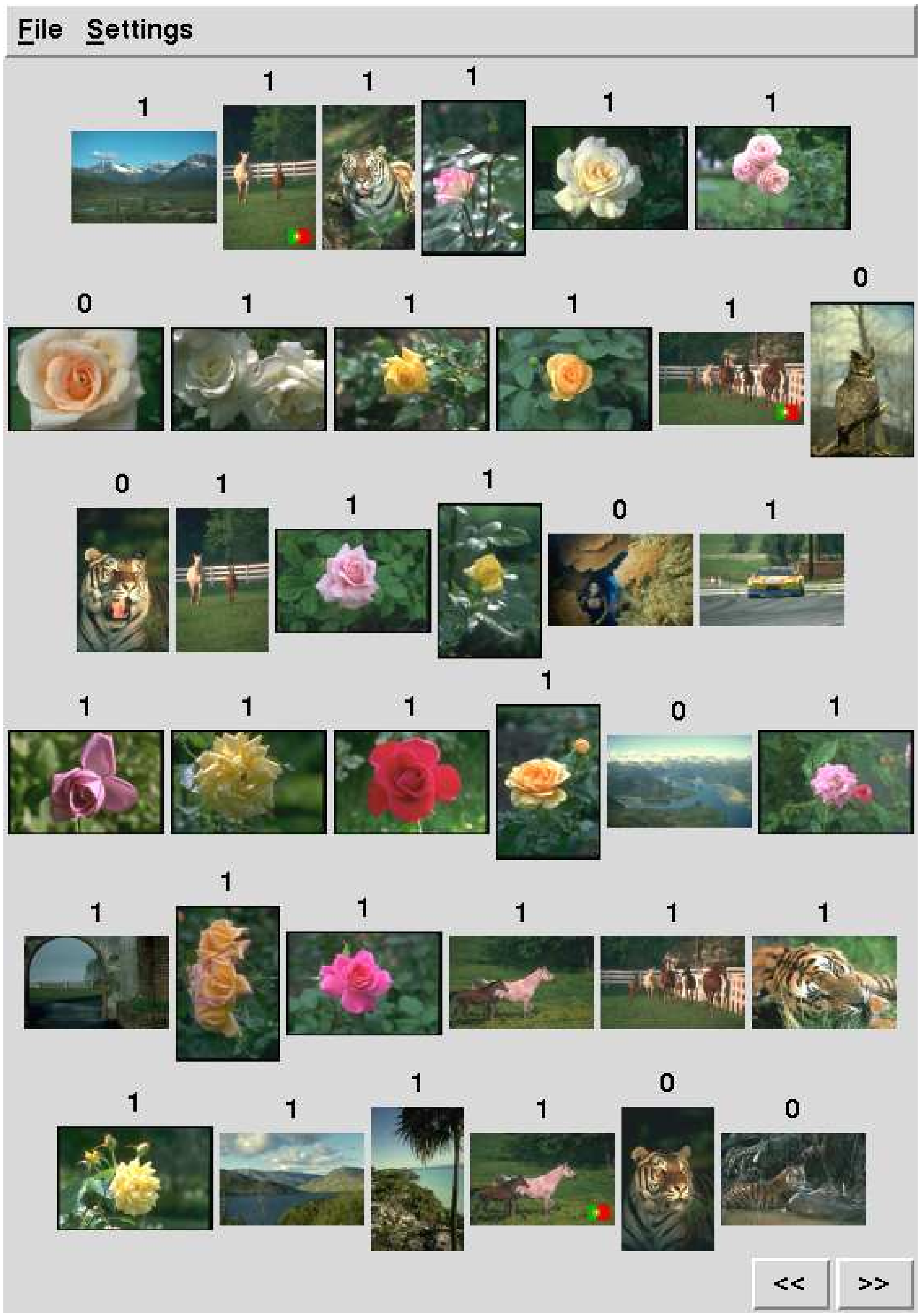

Figure 8.10: Top 36 matches for the vegetation concept.

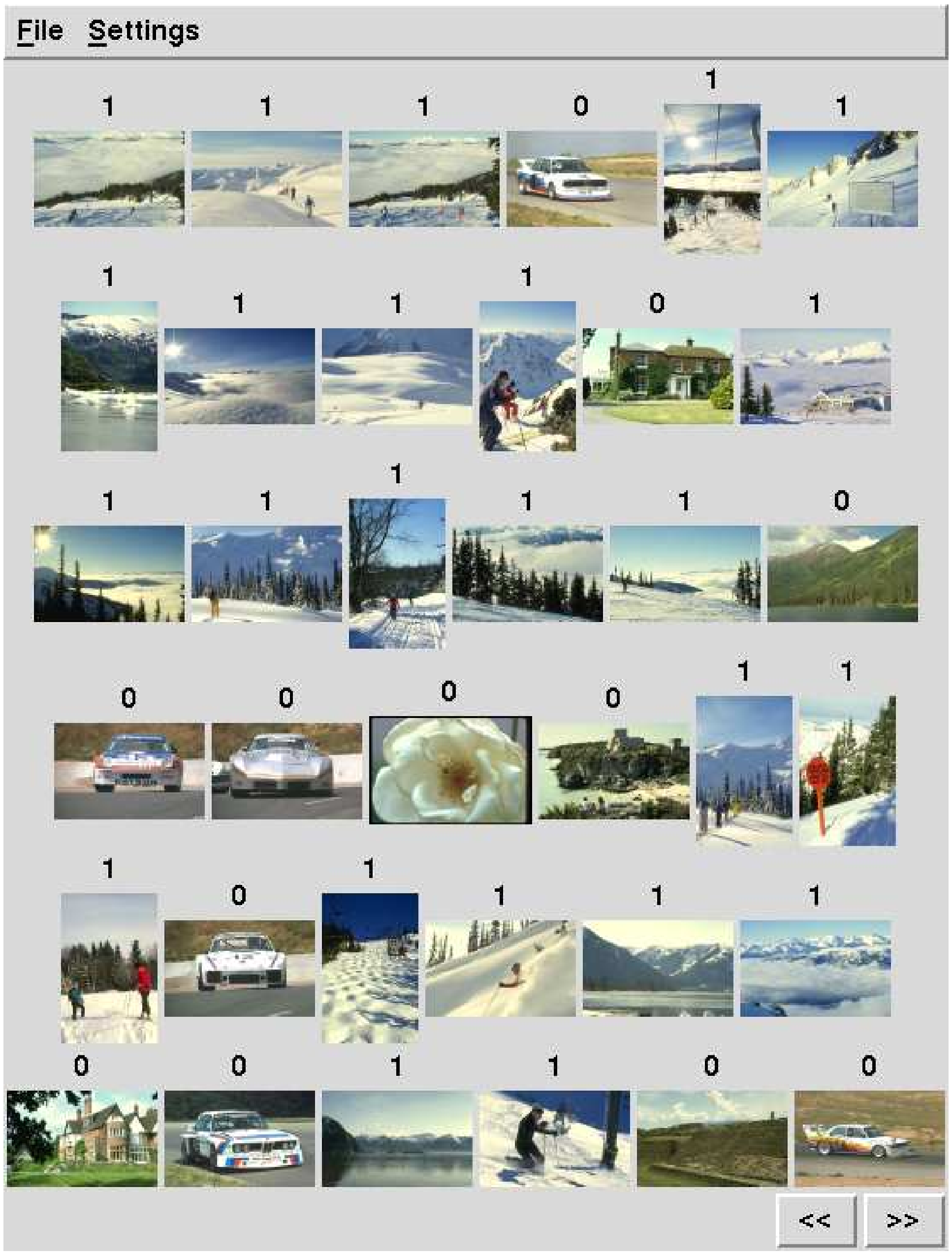

Figure 8.11: Top 36 matches for the snow concept.