

Content-Free Image Retrieval

May 2003

C. Lawrence Zitnick

Takeo Kanade

Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

We present a method for image retrieval that has no explicit knowledge about the appearance of the images within the database, i.e. it is content-free and yet it can retrieve relevant images. A collaborative filtering algorithm is used to make predictions about the current user's image preferences based on their current known preferences and other user's past preferences. The algorithm is based on a maximum entropy technique using a non-standard form of entropy called Rényi's quadratic entropy.

Our algorithm may be used in conjunction with other content or keyword based systems, or by itself if enough user data is available.

1. Introduction

Content-based image retrieval has received a large amount of attention over the last 10 years [4] [8]. As this technology matures and is adopted by more users, another source of information for image retrieval becomes available. When a user enters a keyword, draws a sample image or selects a query image a new list of images, typically thumbnails, is generated with the hope that the user will find them useful. As the user scans the generated list they will select some images while ignoring others. Since the user is looking for a certain type of image during a specific query, the selected images must be related in some manner. This relation could have several forms; a user could be interested in images that contain a specific object, relate to a certain topic, have a specific kind of texture or have any other conceivable relation. By analyzing the user selections, relations between images within the database can be discovered. After enough user data is accumulated, an algorithm for computing image relevance could be completely content-free. That is, it is conceivable that an image retrieval system could rely completely on user feedback without explicit knowledge about the actual appearance of the images.

Assume our system consists of n images $X = \{X_1, \dots, X_n\}$ and we consider a set of associated values

$x = \{x_1, \dots, x_n\}$. During a query the user selects images or keywords for which relevant images are to be found. The images the user selects will form the evidence set X_E . The images within the evidence set are labelled by the user as desired images $x_i = 1$ or undesired images $x_i = 0$. The unlabelled images form the hidden set $X_H = X - X_E$. It is our goal to compute for every image $X_i \in X_H$ the conditional probability $P(X_i = 1|x_E)$. That is, the probability for every hidden image that a user would find the image desirable given their labelling of the evidence images X_E .

A common method for solving problems of this type is called Collaborative Filtering. Collaborative filtering methods attempt to make predictions about a current user's unknown preferences X_H given their current known preferences X_E and preferences from past queries in the database. The database of past queries may consist of queries from the current user AND any other users that have used the system. We will represent the training database T of m past queries as $t_{j,i}$ for all $j \in m$ and $i \in n$. For a query j , $t_{j,i}$ will equal one if the user finds the image X_i desirable and zero otherwise.

1.1. Previous Work

Content-based image retrieval has been an active field of study for many years [4] [8]. Several systems use relevance feedback from users to refine their searches [2] [9] [10] [11]. Most of these systems only apply user feedback to the current query. Recently, some research has been done in long-term learning from user interactions [2].

Collaborative filtering algorithms [1] [6] have been developed for use in many applications, which include predicting user preferences for movies, web pages and television shows. A common approach to collaborative filtering called Nearest Neighbor is to look for other queries in the database that have similar preferences as that of the current query. By finding queries with similar preferences to the known items X_E , we can make predictions about the unknown items X_H . Such a method could compute

$P(X_i = 1|x_E)$ for all $X_i \in X_H$ as follows:

$$\bar{P}(X_i = 1|x_E) = \frac{1}{m} \sum_{j \in m} t_{j,i} \delta(j) \quad (1)$$

where

$$\delta(j) = \begin{cases} 1 & t_{j,k} = x_k \text{ for all } X_k \in X_E \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

Unfortunately, for many x_E no examples will exist in the training set. Nearest Neighbor approaches look for partial matches when there exists a statistically insignificant number of queries that exactly match.

We take a different approach to collaborative filtering in that we attempt to find relations, rather than particular instances, between items or images.

2. Maximum Entropy

It is our goal to compute $P(X_i = 1|x_E)$ for all $X_i \in X_H$. Since our training set $t_{j,i}$ is finite, approximations to the true conditional probabilities must be made. Otherwise, it is not possible to compute them. While we may not be able to reliably compute $P(X_i = 1|x_E)$ for all x_E , there typically exists a subset of all possible x_E for which enough training data does exist to make accurate predictions. More generally, there exists a set of feature functions F for which we can reliably compute $P(X_i = 1|f_k)$ for $f_k \in F$. For example a feature function might correspond to $x_2 = 1$ and $x_4 = 0$ with $X_2, X_4 \in X_E$:

$$f_k(x) = \begin{cases} 1 & x_2 = 1 \text{ and } x_4 = 0 \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

These feature functions f_k and their expected values C_k form a set of constraints on the probability distribution:

$$\bar{P}(X_i = 1|f_k) = \frac{\sum_{j \in m} t_{j,i} f_k(t_j)}{\sum_{j \in m} f_k(t_j)} = C_k \quad (4)$$

Typically, a small set of constraints still doesn't fully constrain the probability distribution; for binary valued variables 2^n constraints would be needed. Obviously, this is infeasible for real world image retrieval systems with thousands of images.

A common method for finding a probability distribution in a partially constrained system is to maximize the entropy while enforcing the constraints. The standard measure of entropy due to Shannon is:

$$H(x) = - \sum_x P(x) \log(P(x)) \quad (5)$$

It has been shown that the probability distribution P^* that maximizes $H(x)$ while enforcing the constraints takes the

following form:

$$P^*(x) = \prod_k \mu_k^{f_k(x)} \quad (6)$$

Computing the parameters μ_k can take exponential time. Approximations may be made, but the use of Shannon's entropy isn't currently feasible for real-time collaborative filtering on large scale systems.

A mathematician named Rényi developed a generalization [7] of Shannon's entropy. This generalization creates an entire family of entropy measures H_α of which one called Rényi's Quadratic Entropy (RQE) takes the following form:

$$H_2(x) = -\log\left(\sum_x P(x)^2\right) \quad (7)$$

Since we are concerned with conditional probabilities we maximize the conditional form of RQE:

$$H_2(X_i|X_E) = -\log\left(\sum_{x_i, x_E \in X_E} P(x_E) P(x_i|x_E)^2\right) \quad (8)$$

Let $\bar{P}(x)$ represent the observed probability from the training data, then we can approximate the conditional entropy by:

$$H_2(X_i|X_E) \approx -\log\left(\sum_{x_i, x_E \in X_E} \bar{P}(x_E) P(x_i|x_E)^2\right) \quad (9)$$

Since we are maximizing the entropy we may drop the log resulting in the final equation:

$$\sum_{x_i, x_E \in X_E} \bar{P}(x_E) P(x_i|x_E)^2 \quad (10)$$

Minimizing the above equation is equivalent to doing weighted least squares on the training set with the feature functions as axes. The resulting conditional probabilities P^* take the following form:

$$P^*(X_i = 1|x_E) = \sum_k \lambda_{i,k} f_k(x_E) \quad (11)$$

For some set of learned weights $\lambda_{i,k}$, which may be computed directly from the following set of equations:

$$\bar{P}(X_i = 1, f_j) = \sum_k \lambda_{i,k} \bar{P}(f_j, f_k) \text{ for all } j \quad (12)$$

The weights may also be learned using a function that iterates through the training data [13].

2.1. Recurrent Linear Network

The algorithm described above that maximizes RQE chooses the domain of the feature functions f_i from the evidence variables X_E . The weights used to compute the conditional probabilities are themselves computed from the feature functions. In real world problems the variables within

X_E will vary from query to query. Ideally, we would like to not have to recompute the weights $\lambda_{i,k}$ for every hidden variable every time the evidence variables X_E change. To accomplish this, we will describe an algorithm called the Recurrent Linear Network (RLN.)

Let the set F consist of all feature functions over the entire set X . A feature function f_i is a member of the "known" or evidence functions F_E if its value can be directly determined from X_E and a member of F_H otherwise. We will assume that there exists a unit feature function f_0 that always has a value of one. The feature functions f_1 through f_n will correspond to a single variable:

$$f_i(x) = x_i \text{ for all } i \in n \quad (13)$$

Additional feature functions may also be added that correspond to combinations of variables within X . Since f_0 always has a value of one it is always known. If $X_i \in X_E$ then $f_i \in F_E$. It may be possible for a feature function to be known even if all of its input variables are not known. For example, if:

$$f_k(x) = \begin{cases} 1 & x_2 = 1 \text{ and } x_4 = 0 \\ 0 & \text{otherwise} \end{cases} \quad (14)$$

then for $x_2 = 0$, $f_k(x)$ will equal zero regardless of the value of X_4 .

The RLN will use the feature functions F to compute estimates of the conditional probabilities $P^*(X_i = 1|x_E)$ that are equivalent to the method described above. The outputs of the network are labelled y_i and correspond to $P^*(f_i(x) = 1|x_E)$. Thus the value of y_i is equal to $P^*(X_i = 1|x_E)$ for all $i \in n$, since $f_i(x) = x_i$ for all $i \in n$.

Similar to the algorithm described above, the following is true for all unknown or hidden feature functions $f_i \in F_H$:

$$P^*(f_i(x) = 1|x_E) = \sum_k \omega_{i,k} y_k \quad (15)$$

where

$$y_i = \begin{cases} P^*(f_i(x) = 1|x_E) & f_i \in F_H \\ f_i(x) & f_i \in F_E \end{cases} \quad (16)$$

The weights $\omega_{i,k}$ for the RLN may be computed from the following set of equations:

$$\bar{P}(f_i, f_j) = \sum_k \omega_{i,k} \bar{P}(f_k, f_j) \text{ for all } j \quad (17)$$

with the following restriction:

$$\omega_{i,i} = 0 \text{ for all } i \quad (18)$$

As stated above, the weights may also be learned using a function that iterates through the training data [13].

Thus we can compute the value of y_i for any set of evidence variables X_E given the same set of weights $\omega_{i,k}$. The only problem is the value y_i is dependent on other y_j s that may not be known. The y_i that correspond to evidence functions, i.e. $f_i \in F_E$, are set equal to the value of the function, however the y_i corresponding to unknown or hidden functions must be computed iteratively. A simple iterative method to solve for all y_i is as follows:

$$y_i^{j+1} = (1 - \sigma)y_i^j + \sigma \sum_k \omega_{i,k} y_k^j \text{ for all } f_i \in F_H \quad (19)$$

and

$$y_i^{j+1} = f_i(x) \text{ for all } f_i \in F_E \quad (20)$$

The parameter σ controls the rate of convergence. Conjugate gradient methods can be used for faster convergence.

2.2. Properties of the RLN

The recurrent linear network possesses a couple interesting properties:

First, the conditional probabilities computed by the RLN are equivalent to those computed by maximizing the conditional form of Rényi's quadratic entropy. As shown in [13], the errors as measured from the true probability distribution are nearly equivalent as those produced using Shannon's entropy. Using maximum entropy helps us avoid over-fitting the data if we limit our selection of feature functions to those which occur a statistically large number of times. As Jaynes [3] states,

Maximum entropy agrees with everything that is known, but carefully avoids anything that is unknown.

Second, many weights within the RLN will be equal to or close to zero. Theoretically, the amount of computation needed to compute the conditional probabilities is $O(IN^2)$ where I is the number of iterations and N is the number of feature functions. For most real world problems, the weights ω will be a sparse matrix resulting in much faster running times. Theoretically, the following can be shown:

$$\omega_{i,j} = 0 \text{ if } (f_i \perp\!\!\!\perp f_j | f_k) \text{ for some } f_k \quad (21)$$

That is, the weight $\omega_{i,j}$ will be equal to zero if there exists a feature function f_k such that f_i is conditionally independent of f_j given f_k .

2.3. Alternative Solution

An alternative solution to finding the values y_i exists if we can compute the pairwise conditional probability matrix $\mathbf{P}_{i,j} = P(f_i | f_j)$. If \mathbf{P}^E consists of all columns of $\mathbf{P}_{i,j}$ such that $f_j \in F_E$ then

$$y = \mathbf{P}^E z \quad (22)$$

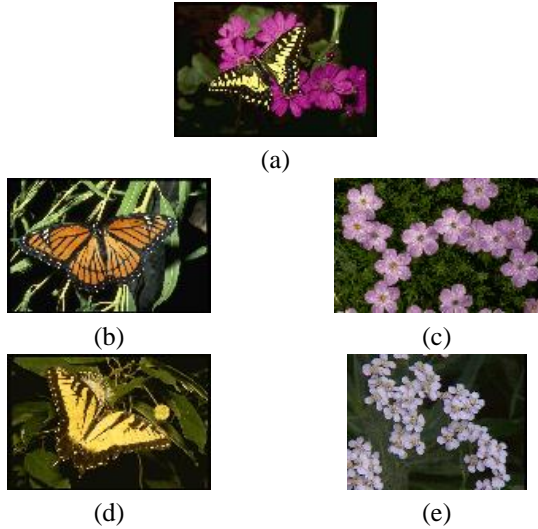


Figure 1: Five sample images from our database.

Selection	Frequency
(a)	0.0180
(a) (b)	0.0720
(a) (b) (d)	0.6480
(a) (c)	0.0162
(a) (c) (e)	0.1458

Table 1: User selections and their frequencies for the five image example.

for the vector z computed from:

$$\mathbf{P}_i^E z = f_i(x) \text{ for all } f_i \in F_E \quad (23)$$

If the number of evidence feature functions F_E is small, this method can be computationally less expensive than computing the y_i 's from the weights ω . The running time is $O(E^3 + E * H)$ were E and H are the number of evidence and hidden functions respectively.

3. Simple Example

To demonstrate how the RLN works for image retrieval we've created the following simple example. Consider the five images in figure 1. Image (a) consists of a butterfly sitting on a flower. Images (b) and (d) are butterfly images without flowers and images (c) and (e) are flower images without butterflies. The system is used by selecting a subset of these images and labeling them as desired or undesired. Our goal is to predict the desirability of the remaining unlabelled images. For our training data we'll assume our users labelled the sets of images in Table 1 as desirable with the corresponding frequencies.

$\omega_{i,j}$	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$
$i = 0$	0.155	0	0	0	0
$i = 1$	0	0.800	0.475	0	0
$i = 2$	0.848	0	-0.475	0.900	0
$i = 3$	0.848	-8.000	0	0	0.900
$i = 4$	0	0.200	0	0	0
$i = 5$	0	0	0.525	0	0

Table 2: Weights $\omega_{i,j}$ computed using the frequencies from table 1.

Query #	(a)	(b)	(c)	(d)	(e)
1	1	0.80	0.18	0.72	0.16
2	1	1	0.00	0.90	0.00
3	1	0.00	1	0.00	0.90
4	1	0	0.90	0.00	0.81
5	1	0.29	0.64	0	0.58

Table 3: Example results for different queries using the RLN. Bold values correspond to evidence images.

We constructed an RLN that has 6 feature functions. A unit function f_0 and one for each image, i.e. f_1 corresponds to image (a), etc. The weights $\omega_{i,j}$ found using the frequencies in table 1 are shown in table 2.

As we stated earlier all weights which correspond to feature functions that are conditionally independent given another feature function are found to be equal to zero. For example, image (a) is independent of image (d) given image (b), thus $\omega_{1,4} = 0$ and $\omega_{4,1} = 0$.

Table 3 demonstrates the RLN's results on several possible user queries. The bold values are evidence images, i.e. the user has labelled the image as either desirable or undesirable. For example, imagine a user labels the butterfly images (a) and (b) as desirable, as in query 2. The RLN correctly predicts the probability of image (d) being desirable as 90% while the flower images (c) and (e) are found to be undesirable. In query 4, the butterfly image (b) is labelled undesirable and the flower images are correctly found to be more desirable than the butterfly images. For this simple example the RLN computes nearly exact probabilities for all possible queries.

4. Large Databases

To demonstrate the RLN on a large number of images we obtained a 9,900 image database from [5] [12]. Along with the images, we selected 80 keywords such as "Butterflies, Mountains, Autumn, Children, Planets, Bridge, Colorful Texture, etc." Several users were asked to select images they thought were similar along with any keywords they thought were relevant. In total, approximately 2,000 entries were made. In a real world system, used by thousands of users, a

much larger amount of input data would be available.

The keywords and images were treated identically with each assigned a feature function. No feature functions besides those corresponding to a single image or keyword were used, resulting in approximately 10,000 feature functions. Training of the system takes approximately 10 minutes. All running times are generated on a 1GHz PC. The computed weight matrix was sparse with only 1,147,680 non-zero weights out of a possible 99,800,100.

To decrease the computation time needed for each query we made the following optimization. The values of y_i^{j+1} were computed using only y_i^j that had values greater than a threshold. For our experiments we set the threshold at 0.01. This optimization has a minimal effect on the ordering of the final values y_i .

On average, between 5 to 20 queries can be made per second using the RLN. The queries typically had between 5 and 50 user labelled images or keywords. Since the keywords were treated identically as the images, keywords are also suggested to the user given their preferences for images and other keywords. For example if image (a) from figure 1 was labelled as desired, the RLN would suggest keywords "Butterflies, Flowers and Nature."

By reformulating the RLN in terms of matrix notation it is possible to compute the values of y_i directly from the pairwise probability matrix P . Given our training data there are 1,169,527 non-zero pairwise probabilities taking about 30 seconds to compute. 40 to 500 queries can be made per second. This is about one order of magnitude faster than computing y_i from the weights $\omega_{i,j}$.

Figures 2 and 3 show some sample queries using the system. In query 1, three images are in X_E with values equal to one. The images show sunsets next to the ocean with two of them containing people. The RLN returns images of ocean sunsets. In contrast, query 2 contains images with people and two of them are ocean scenes. The RLN in this case returns images with people in sunsets.

Query 3 contains a single image of an astronaut with the space shuttle. Varying images of astronauts and space shuttles are returned. In query 4 the value of an astronaut image is set to zero, i.e. undesirable. The RLN then returns space shuttle images.

It is important to remember that the results of the RLN are based heavily on the quality of the input training data. These results are merely meant to show the potential of the system. The RLN has shown significantly better results in other collaborative filtering domains [13].

5. Discussion

We have described a collaborative filtering algorithm using maximum entropy to perform image retrieval. A measure of entropy called Rényi's quadratic entropy is used to create

a computationally efficient algorithm, which is capable of computing 5 to 500 queries per second.

A common problem for collaborative filtering algorithms such as ours is the "cold start" problem [6]. That is, our algorithm needs a large amount of training data from users before it can start producing good results. Unfortunately, users won't use the system until it produces good results. We see two methods for solving this problem. First, content or keyword based image retrieval systems could find sets of similar images that are used as seeds to initially train the system. Second, results could be initially found using the content or keyword image retrieval system and as more user data becomes available the collaborative filtering algorithm could be used. In this way, we believe a hybrid system combining the strengths of several systems will prove to be most beneficial.

Acknowledgments

References

- [1] J. Breese, D. Heckerman and C. Kadie, "Empirical Analysis of Predictive Algorithms for Collaborative Filtering," *Proc. of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, July, 1998.
- [2] X. He, W. Ma, O. King, M. Li and H. Zhang, "Learning and Inferring a Semantic Space from Users Relevance Feedback for Image Retrieval," MSR-TR-2002-62, June, 2002.
- [3] E. Jaynes, "Notes On Present Status And Future Prospects," in *Maximum Entropy and Bayesian Methods*, W. T. Grandy, Jr. and L. H. Schick (eds.), Kluwer, 1991.
- [4] M. S. Lew (Ed.), *Principles of Visual Information Retrieval*. Springer, 2001.
- [5] J. Li, J. Wang, G. Wiederhold, "IRM: Integrated region matching for image retrieval," in *Proc. ACM Int. Conf. on Multimedia*, pp. 147-156, Oct., 2000.
- [6] P. Melville, R. Mooney and R. Nagarajan, "Content-Boosted Collaborative Filtering for Improved Recommendations," in *Proc. Conf. on Artificial Intelligence (AAAI-2002)*, July, 2002.
- [7] A. Rényi, "Some Fundamental Questions of Information Theory," *Selected Papers of Alfred Rényi*, Vol. 2, pp. 526-552, 1976.
- [8] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta and R. Jain, "Content-Based Image Retrieval at the End of the Early Years," *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(12), pp. 1349-1380, 2000.
- [9] K. Tieu and P. Viola, "Boosting Image Retrieval," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 228-235, 2000.

- [10] S. Tong and E. Chang, "Support Vector Machine Active Learning for Image Retrieval," in *Proc. ACM Int. Conf. on Multimedia*, pp.107-118, Oct. 2001.
- [11] Y. Wu, Q. Tian and T. S. Huang, "Discriminant-EM Algorithm with Application to Image Retrieval," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2000.
- [12] J. Wang, J. Li and G. Wiederhold, "SIMPLicity: Semantics-sensitive Integrated Matching for Picture Libraries," *IEEE Trans. Pattern Analysis and Machine Intelligence*, 23(9), pp. 947-963, 2001.
- [13] My Paper

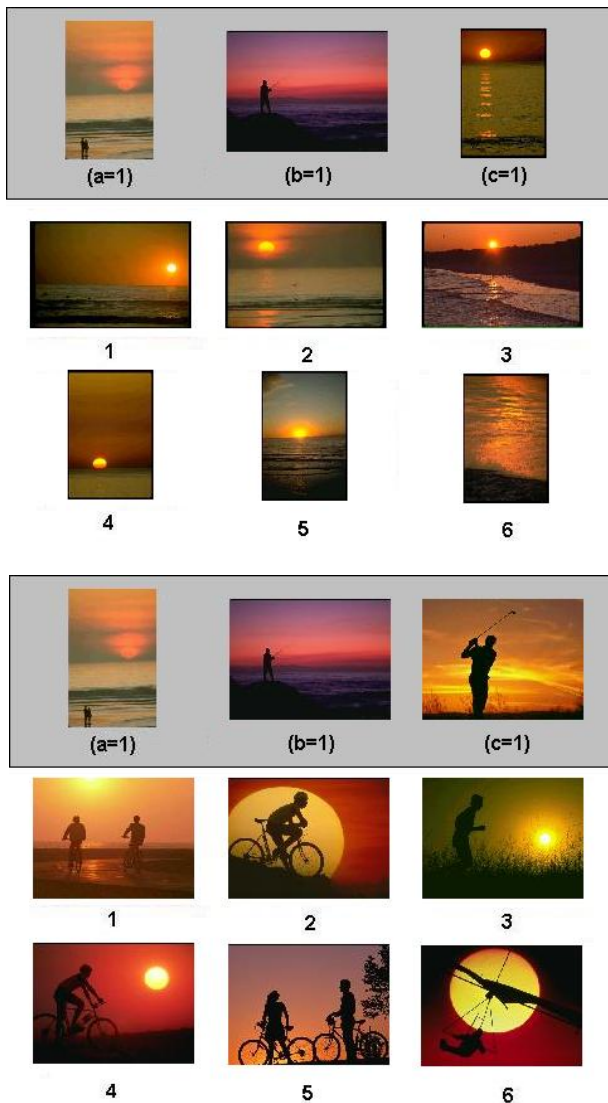


Figure 2: Query 1 and Query 2: Top row contains evidence images followed by the six most relevant images as predicted by the RLN.

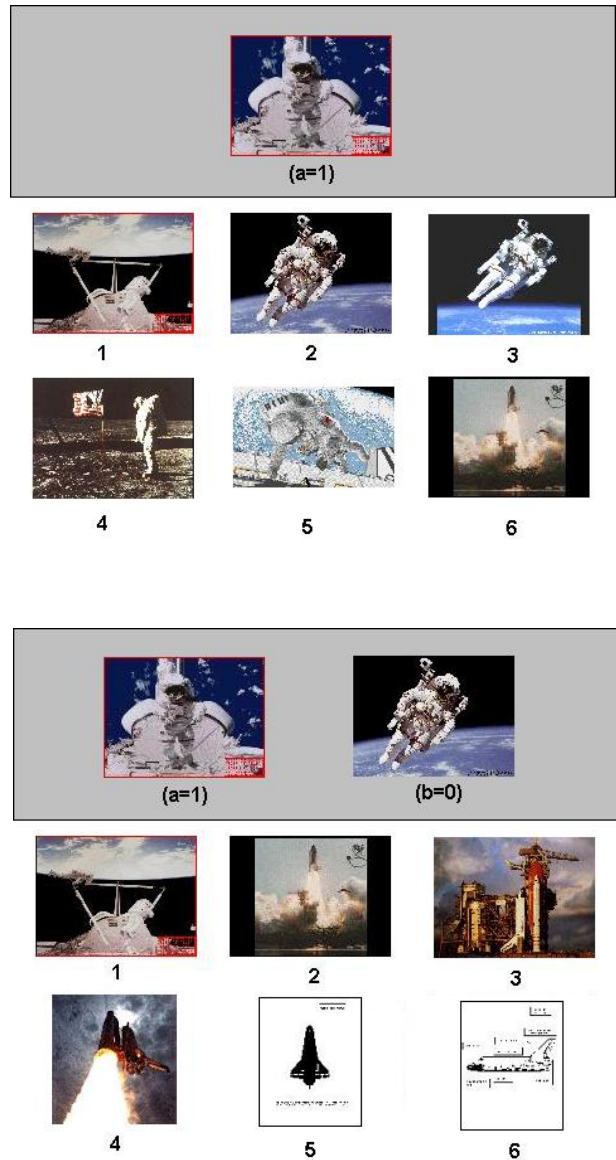


Figure 3: Query 3 and Query 4: Top row contains evidence images followed by the six most relevant images as predicted by the RLN.