

# Speed up interactive image retrieval

Heng Tao Shen · Shouxu Jiang · Kian-Lee Tan ·  
Zi Huang · Xiaofang Zhou

Received: 17 September 2007 / Revised: 27 February 2008 / Accepted: 23 March 2008 / Published online: 23 April 2008  
© Springer-Verlag 2008

**Abstract** In multimedia retrieval, a query is typically interactively refined towards the “optimal” answers by exploiting user feedback. However, in existing work, in each iteration, the refined query is re-evaluated. This is not only inefficient but fails to exploit the answers that may be common between iterations. Furthermore, it may also take too many iterations to get the “optimal” answers. In this paper, we introduce a new approach called OptRFS (optimizing relevance feedback search by query prediction) for iterative relevance feedback search. OptRFS aims to take users to view the “optimal” results as fast as possible. It optimizes relevance feedback search by both shortening the searching time during each iteration and reducing the number of iterations. OptRFS predicts the potential candidates for the next iteration and maintains this small set for efficient sequential scan. By doing so, repeated candidate accesses (i.e., random accesses) can be saved, hence reducing the searching time for the next iteration. In addition, efficient scan on the overlap before the next search starts also tightens the search space

with smaller pruning radius. As a step forward, OptRFS also predicts the “optimal” query, which corresponds to “optimal” answers, based on the early executed iterations’ queries. By doing so, some intermediate iterations can be saved, hence reducing the total number of iterations. By taking the correlations among the early executed iterations into consideration, OptRFS investigates *linear regression*, *exponential smoothing* and *linear exponential smoothing* to predict the next refined query so as to decide the overlap of candidates between two consecutive iterations. Considering the special features of relevance feedback, OptRFS further introduces *adaptive linear exponential smoothing* to self-adjust the parameters for more accurate prediction. We implemented OptRFS and our experimental study on real life data sets show that it can reduce the total cost of relevance feedback search significantly. Some interesting features of relevance feedback search are also discovered and discussed.

**Keywords** Image retrieval · Relevance feedback · Query processing · Indexing

H. T. Shen (✉) · Z. Huang · X. Zhou  
School of Information Technology and Electrical Engineering,  
The University of Queensland, Brisbane, Australia  
e-mail: shenht@itee.uq.edu.au

Z. Huang  
e-mail: huang@itee.uq.edu.au

X. Zhou  
e-mail: zxf@itee.uq.edu.au

S. Jiang  
Department of Computer Science, Harbin Institute of Technology,  
Harbin, China  
e-mail: jsx@hit.edu.cn

K.-L. Tan  
Department of Computer Science,  
National University of Singapore, Singapore, Singapore  
e-mail: tankl@comp.nus.edu.sg

## 1 Introduction

With the popularity of digital imaging technologies, digital images are becoming an important part of our life. This means that tools to manipulate and manage a large collection of images will continue to be in great demand.

Query processing in content-based systems involves retrieving the top-K most relevant objects to the query (i.e., K nearest neighbors (KNN) search in database literature). Given a query, the system searches the database or the indexing structure if any, computes the similarity between the query and each candidate object based on the underlying metric. As image features are typically represented

as high dimensional feature vectors (varying from tens to hundreds), the search process is computationally expensive. Many indexing structures and query processing methods [8, 32, 21] have been proposed to speed up the retrieval in a high-dimensional space. However, the performances of most indexing structures degrade rapidly as the dimensionality increases. For exact KNN search, a linear scan on the whole database turns out to outperform them when the dimensionality reaches high because of the “dimensionality curse” [36]. Recently, approximate KNN search has been investigated to further improve the linear scan significantly [3, 20]. Clearly, designing an efficient content-based system is critical for it to be scalable.

However, existing content-based systems that exploit low-level features (such as color and texture) do not necessarily return semantically relevant (based on human perception) answers. One promising direction towards semantic retrieval is the adoption of relevance feedback mechanism which has been extensively studied recently [5, 10, 12, 14, 15, 18, 22–30, 34, 35, 37, 39, 40]. A relevance feedback process is interactive and iterative in nature. From the current results returned by the system, the user provides feedback to the system; based on these feedback, the system will refine the query to get better results which are closer to the user’s expectations. Feedback query is usually refined from either moving the query to a new position or modifying the similarity metrics (i.e., weights of feature vectors), or both, based on the selected objects. Recent proposal also allows multiple objects to be a refined query [19]. A comprehensive survey on relevance feedback in information retrieval can be found in [41].

While relevance feedback is well studied in information retrieval community, it mainly focuses on accuracy issue and neglects efficiency issue. As modern databases keep growing rapidly, retrieval efficiency becomes more and more critical. Recently, the distance between database and information retrieval research is actively eroding. Many works and discussions on bridging database and information retrieval research have appeared [11, 2, 9, 4, 31, 17, 19], driven by the requirements of many applications which require fast retrieval. Particularly, in this paper, we aim to improve the performance of relevance feedback by utilizing indexing and query processing techniques.

Typically, two factors affect the overall efficiency of relevance feedback:

**Searching time during each iteration** Since the feedback query moves away from the previous one with updated similarity metric, a complete KNN search has to be re-performed in next iteration. Generally, *random accesses* on candidates are the major concern in most research work.

**Number of iterations** Depending on the query, a number of iterations may be processed before the results converge to the user’s expectation. Given a query, the larger number of iterations occurs, the longer the searching takes.

We intend to speed up relevance feedback search by reducing both the random accesses during each iteration and the total number of iterations. In feedback loop, a new search is performed in each iteration due to the change of feedback query. However, the search spaces of two consecutive queries may overlap largely given that the query in current iteration is refined based on the “good” results from the last iteration. Furthermore, the refinement of feedback query may follow certain patterns. In this paper, we propose OptRFS, a new method to optimize the relevance feedback search, by discovering the overlap between two consecutive iterations and predicting the “optimal” query at an early stage.

OptRFS investigates three methods called *linear regression*, *exponential smoothing* and *linear exponential Smoothing* to predict the new query to be searched in next iteration. Taking the features of relevance feedback search into consideration, OptRFS further introduces *adaptive linear exponential smoothing* to achieve better prediction quality. By forecasting the search space of the new query, the overlap between two consecutive queries’ search spaces can then be estimated. By performing sequential scan on the overlap, expensive random accesses on those candidates lying in the overlap can be avoided in next iteration, hence the total number of random accesses can be reduced. For further improvement, OptRFS may also advance the prediction by multiple iterations to reach (or get closer to) the “optimal” query at a faster pace. By analyzing the changing trend of the queries executed in the early iterations, OptRFS is able to predict the “optimal” query accurately. By jumping to search the predicted “optimal” query directly, multiple intermediate iterations during the feedback loop can be avoided. Hence, OptRFS achieves to improve relevance feedback search by predicting the overlap and the “optimal” query. OptRFS can be easily and well integrated with existing feedback mechanisms and indexing structures. The strengths and weaknesses of forecasting methods in relevance feedback search are also analyzed. Experiments study proves both the effectiveness and efficiency of OptRFS, and also discovers some interesting features of relevance feedback search.

The rest of paper is organized as follows. Section 2 provides a review on preliminary work and related work. OptRFS, including its prediction models and KNN algorithms, is introduced Sect. 3. Experiments results are reported in Sect. 4. Finally we conclude the paper in Sect. 5.

## 2 Preliminary and related work

### 2.1 Relevance feedback mechanism

Low-level feature representations of a multimedia object often fail to capture its semantic meanings. Relevance feedback is a process to iteratively refine the user’s query to

improve the accuracy based on user's perception. After a number of iterations, the feedback loop is terminated with "optimal" results, corresponding to "optimal" feedback query [6]. It can be summarized as follows.

1. **Issue an initial query** The user issues an initial query  $Q(q, w)$  to start the search, where  $q$  is the query point, and  $w$  is the similarity metric used.
2. **Search the database** Search engine compares the query point  $q$  with objects in the database, or the candidate objects if any indexing structure is applied, with underlying  $w$  by using a distance function. Weighted Euclidean function is a popular measure to compute the relevance between the query and database objects. The  $K$  most relevant objects are then returned to the user.
3. **Iterative feedback loop** The user judges the relevances of the returned objects and identifies which are "good" and "bad" (a relevance score can also be evaluated by the user). Based on the relevance provided by the user, a new feedback query  $Q_t(q_t, w_t)$  for the  $t$ th iteration, where  $q_t$  and  $w_t$  are the new query point and similarity metric in  $t$ th iteration, is then computed and passed to the search engine for next round of search.
4. **Terminate the loop** The feedback loop is terminated when the user feels the returned results are satisfactory.

Particularly, the feedback provided by users often affects the progress of convergence to the "optimal" feedback query. Different learning processes can be embedded into relevance feedback to help users provide better feedback so that faster convergence can be achieved [12, 13, 33, 39]. In this paper, for simplicity, users judge the (ir)relevance of the results without applying any learning process.

As mentioned, a multimedia object is usually represented by high-dimensional features vectors. The relevance between two object features is usually computed based on some distance functions. For easy illustration purpose, consider a database of  $N$  objects. We denote the query in  $t$ th iteration as  $Q_t$ , its point as  $q_t$ , its metric as  $w_t$ , its results as  $R_t$  and its candidate set as  $C_t$ . Let  $q_t[i]$  represent the  $i$ th coordinate value of  $q_t$ . Correspondingly, let  $w_t[i]$  represent the weight of  $i$ th dimension of  $q_t$ . The similarity between two objects  $Q$  and  $P$  in  $t$ th iteration is usually computed by the popular weighted Euclidean distance, i.e.,

$$d(Q_t, P_t) = d(q_t, p_t; w_t) = \sqrt{\sum_{i=1}^D w_t[i] * (q_t[i] - p_t[i])^2}$$

where  $D$  represents the dimensionality of feature space. A table of notations is listed in Table 1 for easy lookup.

**Table 1** A table of notations

Notation	Description
$N, D$	Number of objects and dimensionality of space
$Q_t, q_t, w_t$	A query, its point and metric in $t$ th iteration
$Q'_t, Q''_t$	Prediction, double prediction of $Q_t$
$Q_{opt}$	Optimal query of $Q$
$R_t, C_t$	Results and candidate set for $Q_t$
$d$	Distance function
$\sigma_t[i]$	Standard deviation along $i^{th}$ dimension
$\gamma_{t+1}$	$K$ th largest upper bound found so far
$\theta_{t+1}$	$K$ th largest upper bound of $C_t$ with $w_{t+1}$
$r_{t+1}^u$	upper bound on the distance of $K$ th NN
$r_{t+1}$	Tightest pruning radius
$\alpha, \beta$	Forecasting parameters
MSE	Mean square error

## 2.2 Feedback query refinement

Different relevance feedback strategies may be adopted in different systems. The most important issue is how to refine the feedback query. There are two basic methods. The first is to move the query to a new position. The second is to re-weight the importance of feature components. Both methods try to update the query towards the user's expectation.

**Query point movement** The query point's coordinate values are modified iteratively based on the feedback. The intuition of this method is to move the query point towards the good results and away from the bad ones. Ishikawa et al. [16] showed that the "optimal" query point is a weighted average of good results.

**Similarity metric modification** The weighted similarity metric is updated iteratively in feedback loop. The intuition of this method is to assign higher weights to the feature components that are more important in deciding the good results. Ishikawa et al. [16] proved that the "optimal" assignment of weight for  $i$ th dimension at  $(t + 1)$ th iteration is:  $w_{t+1}[i] = \frac{1}{\sigma_t[i]^2}$  where  $\sigma_t[i]$  is the standard deviation of coordinates values along the  $i$ th dimension from all the good results at  $t$ th iteration. The weight is further normalized as follows:  $w_{t+1}[i] = \frac{w_{t+1}[i]}{\sum_{i=1}^M w_{t+1}[i]}$  where  $M$  is the number of good results selected by the user. Rui and Huang [27] have extended the above technique by considering multiple features.

## 2.3 Existing proposals

While most previous works focus on the accuracy issue, very little has been done to solve the efficiency issue. There are two early attempts to speed up the relevance feedback by

either reducing searching time during each iteration [38] or reducing number of iterations [6].

### 2.3.1 Reduction of searching time during each iteration

Along the direction of reducing searching time during each iteration, Wu and Manjunath [38] exploit the correlations between two consecutive iterations when the underlying metric is changing. The method applies the known indexing structure—VA-file [36] with a tighter upper bound in its filtering process. The vector approximation file (VA-file) divides the data space into  $2b$  rectangular cells where  $b$  denotes a user specified number of bits (e.g. some number of bits per dimension, typically 4–6). The VA-file allocates a unique bit-string of length  $b$  for each cell, and approximates data points that fall into a cell by that bit-string. The VA-file itself is simply an array of these compact, geometric approximations. Nearest neighbor queries are performed by scanning the entire approximation file, and by excluding the vast majority of vectors from the search (filtering process) based only on these approximations.

The idea behind VA-file in [38] is to find a possible tighter upper bound on the  $k$ th nearest neighbors in the two-phase filtering process of VA-file. The two-phase filtering process can be standardized as follows: (a) In the first phase, the VA-file is sequentially scanned. The lower and upper bounds on the distance for each object's approximation (i.e., VA) is then computed. A buffer is used to remember  $K$ th largest upper bound found so far. Denote the  $K$ th largest upper bound found so far in  $(t + 1)$ th iteration as  $\gamma_{t+1}$ . If a VA's lower bound is greater than  $\gamma_{t+1}$ , the object can be safely filtered. Otherwise, it is considered as a candidate and its upper bound is used to update the buffer and  $\gamma_{t+1}$ . (b) In the second phase, the candidates obtained from first phase are increasingly ordered by their lower bounds. They are then visited by random accesses. Once an object's lower bound reaches the  $K$ th actual largest distance computed so far, the algorithm terminates and KNNs are returned.

Wu and Manjunath [38] improved the first phase of the filtering process in VA-file by using a tighter upper bound (denoted as  $r_{t+1}$ ) than  $\gamma_{t+1}$ .  $r_{t+1}$  enforces two more constraints.

The first is an upper bound on the distance of  $K$ th nearest neighbor in  $(t + 1)$ th iteration. Denote this upper bound as  $r_{t+1}^u$ . It is computed as:

$$r_{t+1}^u = \max\{d(q_{t+1}, R_t[i]; w_{t+1}), i = 1, \dots, k\}.$$

The second is the  $K$ th largest upper bound of  $C_t$  (i.e., the candidates from  $t$ th iteration) based on  $w_{t+1}$  (i.e., the similarity metric in  $(t + 1)$ th iteration). Denote this upper bound as  $\theta_{t+1}$ . As a result, for an object qualified to be a candidate, its VA's lower bound must be less than  $r_{t+1}$ , where

$r_{t+1}$  is computed as:

$$r_{t+1} = \min\{\gamma_{t+1}, r_{t+1}^u, \theta_{t+1}\}$$

Obviously,  $r_{t+1} \leq \gamma_{t+1}$ . Any object whose VA's lower bound is greater than or equal to  $r_{t+1}$  will be filtered away. By enforcing the tighter constraint  $r_{t+1}$ , fewer candidates can be included.

However, this method is limited by the following drawbacks. First and most importantly, repeated random accesses during two consecutive iterations occur. Second, the issue of the query point movement is not mentioned. Third, it does not remember the information before the last iteration. This may limit its capability to prune the search space significantly. We will compare it with our method with greater details in Sect. 3.4.

### 2.3.2 Reduction of number of iterations

Along the direction of reducing the number of iterations, Bartolini et al. [6] proposed a technique called Feedback-Bypass. By storing and maintaining the information on the queries gathered from past feedback loops, FeedbackBypass is possible to either 'bypass' the feedback loop completely for already-seen queries or to 'predict' the near-optimal parameters for the new queries. In both cases, as an overall effect, the number of feedback iterations can be reduced. However, FeedbackBypass requires a learning process before it can predict a near-optimal setting for the parameters.

## 3 The OptRFS

In this paper, our goal is to achieve efficient KNN search during the feedback loop by reducing the searching time during each iteration and reducing the number of iterations. To the best of our knowledge, considering both factors (i.e., searching time during each iteration and number of iterations) in relevance feedback search has never been addressed in the database research.

Our approach is to achieve faster retrieval in the subsequent iterations as the relevance feedback loop goes on and terminate the feedback loop as early as possible. Based on the information obtained from the early iterations, the number of random accesses can be further reduced in the subsequent iteration. Meanwhile, the number iterations can also be reduced by skipping some intermediate iterations. Hence **the key is to maintain the information among the iterations and explore their correlations.**

Our inspirations come from the following observations from relevance feedback search. First, from the methods in recomputing the query point and similarity metric as shown in Sect. 2.2, **search space of the refined query  $Q_{t+1}$  is highly likely to overlap with that of  $Q_t$ .** Second, relevance feedback

mechanisms assume that the feedback query is modified towards the “optimal” query further as more iterations are processed. Third, queries executed in the early iterations may suggest their changing trend.

If we know which candidate in the  $i$ th iteration will be re-accessed in the  $(i + 1)$ th iteration, we could store them for efficient scan to avoid the expensive random accesses. Generally the number of candidates is very small compared with the data size. And the overlap is expected to be smaller. Maintaining such a small set of candidates for efficient scan can speed up the search. Furthermore, pre-scan on the overlap produces a potentially smaller pruning radius which tightens the search space for the refined query. This provides larger room of reducing the search space. On the other hand, after very first few iterations, if we know where the “optimal” query is, we can directly jump to the final iteration without moving the query slowly towards the optimal position. Apparently, integration of two powers has potential to speed up relevance feedback search by a larger scale. Next, let us look at how OptRFS discovers the overlap and the “optimal” query respectively.

### 3.1 Overlap prediction

To discover the overlap, the key is to predict  $Q_{t+1}$  and its search space. From the second observation that the query changes over iterations towards to its destination, this inspires OptRFS to explore using *linear regression model* and *exponential smoothing model* to predict the overlap.

OptRFS forecasts  $Q_{t+1}$  and  $r_{t+1}$  by using the query and search radius information from the first iteration to the  $t^{th}$  iteration. The prediction is made for every dimension of the  $Q_{t+1}$ , including the coordinate value and weight. Denote the prediction of  $Q_{t+1}$  and  $r_{t+1}$  as  $Q'_{t+1}$  and  $r'_{t+1}$  respectively. We first look at how the *linear regression* can be adapted in relevance feedback, followed by *exponential smoothing* [7].

#### 3.1.1 Linear regression

The feedback queries are assumed to move along a direction from initial query to its “optimal”. This satisfies the assumption of linear regression (LR) that the data change over time increasingly or decreasingly.

First, we look at how the  $i$ th dimensional value can be predicted. We denote the prediction of  $q_{t+1}$  to be  $q'_{t+1}$ . Hence the forecast of  $i$ th coordinate value by using linear regression is computed as follows:

$$q'_{t+1}[i] = \alpha + \beta * t$$

where  $t$  represents the  $t$ th iteration,  $\alpha$  and  $\beta$  are the parameters to be determined by regression and  $\beta$  indicates the amount changed over each iteration. To decide  $\alpha$  and  $\beta$ , we use the

least squares estimates as follows [7]:

$$\beta = \frac{\sum_{j=1}^t j * q_j[i] - t * \bar{j} * \bar{q}[i]}{\sum_{j=1}^t j^2 - t * \bar{j}^2}$$

$$\alpha = \bar{q}[i] - \beta * \bar{j}$$

where  $\bar{q}[i]$  and  $\bar{j}$  is the average of  $q[i]$  and  $j$  respectively, i.e.,

$$\bar{q}[i] = \frac{\sum_{j=1}^t q_j[i]}{t} \quad \text{and} \quad \bar{j} = \frac{\sum_{j=1}^t j}{t}$$

The estimates of  $\alpha$  and  $\beta$  give the least value of forecasting SSE (sum of square error), where

$$\text{SSE} = \sum_{j=1}^t (q_j[i] - q'_j[i])^2 = \sum_{j=1}^t (q_j[i] - \alpha - \beta * j)^2.$$

The model can be summarized in two steps. The values of  $\alpha$  and  $\beta$  are first estimated using the query information from early iterations. By using the resulting values of  $\alpha$  and  $\beta$ , the forecast of  $q_{t+1}$  can be computed. Based on the above regression model, the weight of each dimension can also be predicted. Finally  $Q'_{t+1}$ , the prediction of  $Q_{t+1}$ , is obtained.

So far we have seen how  $Q_{t+1}$  can be predicted by using linear regression. To decide the search space of  $Q'_{t+1}$ , its tightest pruning radius is also necessary to be known. Notice that most real multimedia datasets are not uniformly distributed. It is often that different regions in the space have different densities. In other words, given different queries in the space, their KNN search radii might vary greatly. Apparently, a query moving to a region with a smaller density tends to have a larger search radius, and vice versa. In feedback loop, as the query is refined from one position to another, its search radius may correspondingly change too. Depending on the change trend of space density, its radius may turn to be larger or smaller gradually. Intuitively, we also apply the same regression model to predict the search radius, and denote it as  $r'_{t+1}$ , given all the radii in early iterations. Denote the predicted overlap of search spaces between the  $t$ th and  $(t + 1)$ th iterations as  $Overlap'_{t,t+1}$ . Then  $Overlap'_{t,t+1}$  contains the candidates in  $C_t$  whose distances to  $Q'_{t+1}$  are not greater than  $r'_{t+1}$ .

Linear regression works with another assumption that the amount of change every time is fixed. However, in relevance feedback search, the changes over iterations may vary. Furthermore, the importance of past queries in linear regression is equal in predicting the next query. One interesting feature in relevance feedback is that the refined queries get closer and closer to the “optimal” over iterations, i.e.,  $Q_{t+1}$  is assumed to be closer to the “optimal” than  $Q_t$ . This indicates that the more recent query carries more information about the next. It is thus natural for us to investigate an alternative approach, *simple exponential smoothing* (or *exponential smoothing*)

which assumes that the most recent query is most important in determining the next prediction and whose quality is better when few predictions are required.

### 3.1.2 Exponential smoothing

Exponential smoothing (ES) assigns **unequal weight on different data, the largest weight on the most recent data**, and the least weight on the earliest data. It provides better predictions when the prediction is not greatly extended. When applied to relevance feedback, exponential smoothing gives greatest weight to the most recent query, and the least weight to the initial query, i.e., it is more “responsive” to changes occurring in the recent iterations. In exponential smoothing,  $q'_{t+1}[i]$  is computed as:

$$q'_{t+1}[i] = \alpha * q_t[i] + (1 - \alpha) * q'_t[i]$$

where  $\alpha$  is the smoothing parameter and  $0 < \alpha < 1$ .

Exponential smoothing is easy to use since only  $q_t[i]$  and its prediction  $q'_t[i]$  are required.

Expanding the above smoothing equation, we get

$$\begin{aligned} q'_{t+1}[i] &= \alpha * q_t[i] + (1 - \alpha) * q'_t[i] \\ &= \alpha * q_t[i] + (1 - \alpha) * [\alpha * q_{t-1}[i] \\ &\quad + (1 - \alpha) * q'_{t-1}[i]] \\ &= \dots \\ &= \sum_{j=0}^{t-1} \alpha * (1 - \alpha)^j * q_{t-j}[i] + (1 - \alpha)^t * q'_1[i] \end{aligned}$$

where  $q'_1[i]$  is the initial prediction. As we can see, the **exponential smoothing prediction is the weighted sum of all the past queries with the weight increasing as the iteration goes on**. There is no general way to get the  $q'_1[i]$ . A popular way is to use the average of all the known values, i.e.,

$$q'_1[i] = \frac{\sum_{j=1}^t q_j[i]}{t}$$

Exponential smoothing is intuitively more appealing. One major **drawback** of exponential smoothing is that **there is no intrinsic best value for  $\alpha$** . To determine  $\alpha$ , generally a set of values are tested, and the value which best fits the queries is selected. We use the set of [0.05, 0.1, ..., 0.9, 0.95] to choose  $\alpha$ . The value which gives the minimal SSE is then chosen. That is, given a feedback query, *using the  $\alpha$  value with minimal SSE is expected to achieve the most accurate prediction*. Clearly, a large  $\alpha$  adjusts more quickly to recent query changes. Notice that **exponential smoothing tends to lag behind a trend [7], which causes less accurate prediction**. **Exponential smoothing is appropriate when the underlying feedback queries behaves like a constant, i.e., when the mean of queries is moving very slowly.**

### 3.1.3 Linear exponential smoothing

In relevance feedback search, although the queries **tend to get closer to the “optimal” query, the trend does not necessarily remain constant**, i.e., the trend may vary slowly over time. This could be more convincing when more data are included. **To capture the time-varying/local trends of feedback queries**, one method is to use **linear (i.e., double) exponential smoothing (LES)**.

LES modifies exponential smoothing for following a linear trend, i.e., smooths the smoothed values obtained from double application of exponential smoothing. Denote the predictions of singly-smoothing and doubly-smoothing by exponential smoothing as

$$q'_{t+1}[i] = \alpha * q_t[i] + (1 - \alpha) * q'_t[i]$$

and

$$q''_{t+1}[i] = \alpha * q'_t[i] + (1 - \alpha) * q''_t[i]$$

respectively. Notice that the same  $\alpha$  is used. Then the final prediction of LES will be:

$$q_{t+1}^{\text{LES}}[i] = a(t) + b(t)$$

where

$$a(t) = 2q'(t) - q''(t)$$

and

$$b(t) = (\alpha/(1 - \alpha))(q'(t) - q''(t))$$

$a(t)$  is the estimated *value* and  $b(t)$  is the estimated *trend* at  $t$ th iteration. Clearly, **LES best fits the scenario when the queries have a local trend**. Our experiment results later will suggested that **local trends indeed exist for some relevance feedback search**. To select  $\alpha$ , we use the same strategy as that in exponential smoothing.

### 3.1.4 Adaptive linear exponential smoothing

Naturally, the underlying behavior of  $q_t$  series may change as relevance feedback search goes on. **If the behavior seems stable, queries in very early iterations can be used in forecasting so as to take advantage of averaging**. On the other hand, **if the behavior changes, recent queries are more desirable since very early queries are no longer valid for current behavior**. Meanwhile, one distinguishable feature of relevance feedback is that the **number of iterations may be very small** (usually less than 10). That is, the **number of past queries in hand for prediction is rather limited**. This simply suggests that a **noise** (i.e., a query with suddenly abnormal change) may affect the quality of prediction. For example, in ES or LES, the selection of  $\alpha$  is decided by SSE. A **noisy query may dominate the overall SSE**. As a result, a  $\alpha$  value far from best may be selected.

## ALES

1. Compute and record the prediction errors for each iteration
2. Identify the changing trend of errors
3. If the noisy queries exist
4. Smooth them and re-identify the trend of errors
5. If the errors tend to be smaller/constant
6. Reduce  $\alpha$  to be next smaller value from the set
7. Else if the errors tend to be larger
8. Increase  $\alpha$  to be next larger value from the set
9. Use new  $\alpha$  to predict next query

**Fig. 1** Adaptive linear exponential smoothing

Considering above factors, we introduce *adaptive* prediction for relevance feedback search. In this paper, we apply the adaptive strategy particularly on LES for its effectiveness as shown in experiments and name it as adaptive linear exponential smoothing (ALES). The following Fig. 1 outlines the algorithm of ALES.

ALES monitors the prediction error (i.e.,  $|q_t - q_t^{\text{LES}}|$  for  $t$ th iteration) and judges its changing trend (lines 1–2). From the changing trends, ALES identifies and smooth the noisy queries (lines 3–4). A query is identified as a noise if its prediction error exhibits a sharp up or down along the changing trend. In our experiments, we use the following heuristic. A query is identified as a noisy query if the following inequality holds:

$$|q_t - \frac{q_{t-1} + q_{t+1}}{2}| \geq \frac{q_{t-1} + q_{t+1}}{2}$$

That is, we compute a query's change volume by looking at its *left and right neighbors*. The above inequality also suggests that the noise identification is more sensitive when the query value is smaller. Bear in mind that the number of iterations in relevance feedback is small. *Instead of removing the noisy queries, we smooth them by automatically replacing them with their predictions*. The purpose of doing so is to have enough data for future prediction without sacrificing the quality. The intrinsic changing trend is then re-identified. *If the prediction errors tend to be smaller and smaller over iterations, far early queries well fit the current behavior and a smaller  $\alpha$  value is expected to reduce the error more quickly* (line 5–6). On the other hand, *if the prediction errors tend to be larger and larger over iterations, far early queries are out of date for the current behavior and a larger  $\alpha$  value is expected to emphasize on more recent queries* (line 7–8). Finally, *new prediction is made on the modified  $\alpha$  value* (line 9).

As we can see, ALES considers noises and modifies  $\alpha$  (its values has to be in the range of (0, 1)) to be adaptive on the changing prediction error. Furthermore, in relevance feedback search, ALES has the functionality of smoothing noisy queries too.

So far, we have looked at four prediction models for query point prediction. Similarly, for each prediction model, the weight for each dimension and the pruning radius in next

iteration are also predicted to compute the overlap. Notice that prediction methods need information of at least two queries before it can make the first prediction. Hence, above methods can only be applied after the second iteration. To predict  $Q_2$ , we use the same methods as described in Sect. 2.2 applied on whole  $R_1$  by assuming that all the results are “good”. If distance functions like weighted Euclidean function are used, generally speaking, the smaller the distance is, the greater the relevance is. To transfer the distance value of the  $i$ th nearest neighbor (except the query itself if it comes from database)  $R_1[i]$  into its corresponding weight, we use the following formula:

$$w(R_1[i]) = \frac{\sum_{j=1}^K d(Q_1, R_1[j])}{d(Q_1, R_1[i])}$$

It is then normalized as follows:

$$w(R_1[i]) = \frac{w(R_1[i])}{\sum_{j=1}^K w(R_1[j])}$$

The predicted query point  $q'_2$  is then computed:

$$q'_2 = \sum_{j=1}^K w(R_1[j]) * R_1[j]$$

The weight for each dimension is assigned as formulated in Sect. 2.2, where the standard deviation is computed based on all the results.

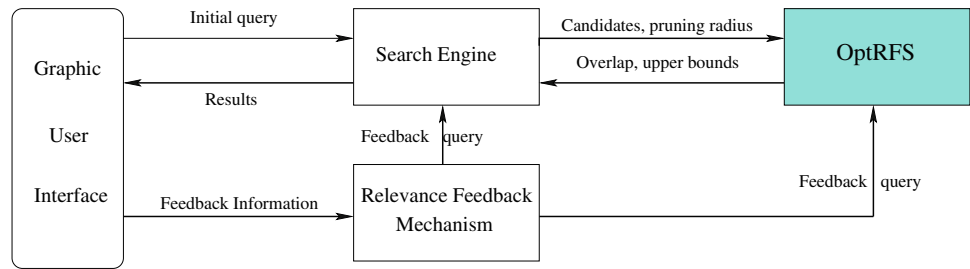
## 3.2 “Optimal” query prediction

In the last subsection, we have discussed how to predict the next refined feedback query  $Q_{t+1}$  and its search radius  $r_{t+1}$  based on  $Q_1$  to  $Q_t$  and  $r_1$  to  $r_t$ , so as to estimate the overlap between  $t$ th and  $(t+1)$ th iterations. Given an initial query  $Q_1$ , we assume its “optimal” query, denoted as  $Q_{\text{opt}}$ , is reached after  $\text{opt}$  iterations. After the first  $t$  iterations, it still needs  $(\text{opt}-t)$  iterations to terminate the feedback loop.

From the third observation that *queries executed in the early iterations may suggest their changing trend, it inspires OptRFS to discover  $Q_{\text{opt}}$  directly so that the intermediate  $(t+1)$ th to  $(\text{opt}-1)$ th iterations can be saved in feedback loop*. That is, given the information in the past  $t$  iterations, OptRFS tries to predict  $Q_{\text{opt}}$  instead of  $Q_{t+1}$ . However, this is not trivial. To do so, the key is to know the value of  $\text{opt}$ .

In our forecasting models, as more data become available, the prediction tends to be more accurate [7]. Hence in feedback loop, the prediction error in the next iteration is expected to be smaller than that of the current iteration. In the ideal situation, the prediction error is zero when the “optimal” query is reached. Motivated by this, we *derive  $\text{opt}$  by forecasting when the prediction error approaches zero*. Here we use the prediction error defined as  $|q_t - q'_t|$  for  $t$ th iteration. Given the prediction errors in the past  $t$  iterations and

**Fig. 2** A general relevance feedback architecture integrated with OptRFS



$|q_{\text{opt}} - q'_{\text{opt}}| = 0$ , the value of  $opt$  can be easily determined when any of four prediction models is applied. After knowing the value of  $opt$ ,  $Q_{\text{opt}}$  can be simply predicted based on  $Q_1$  to  $Q_t$  by treating the prediction of  $Q_{t+1}$  to  $Q_{\text{opt}-1}$  as real refined queries.

Notice that our prediction of  $opt$  is based on the assumption that the prediction error is zero when the feedback loop is terminated. However, this may not always be the case since the prediction models often carry prediction errors. Consequently, the “optimal” query might be over-predicted. Assume the prediction of “optimal” query is invoked after  $t$  iterations. Generally, the larger the  $t$  is, the more accurate the prediction is, and the less number of iterations is saved.

In short, by predicting the “optimal” query, OptRFS can quickly direct the feedback search to the final iteration so that the number of iterations can be reduced, with minor accuracy sacrifice. The detailed embedment of OptRFS in feedback search will be discussed in the next subsection.

### 3.3 System overhead

OptRFS investigates linear regression and exponential smoothing. Their performances will be reported shortly. The final output of OptRFS is  $Overlap'_{t,t+1}$ . As mentioned, OptRFS can be nicely integrated with existing feedback mechanisms and search methodologies. Figure 2 depicts a general system architecture enriched with OptRFS. As we can see, OptRFS receives the query information and candidate set, and returns the overlap to the search engine which deploys some KNN search algorithms for the corresponding indexing structures.

During the  $t$ th iteration, relevance feedback mechanism generates  $Q_t$  and passes it to both search engine and OptRFS. OptRFS maintains the past queries. Upon completing the  $t$ th iteration, search engine returns  $R_t$  to the user. Meanwhile, it passes  $r_t$  and  $C_t$  to OptRFS. OptRFS then predicts  $Overlap'_{t,t+1}$  based on the past queries,  $r_t$  and  $C_t$ . After predicting  $Overlap'_{t,t+1}$ , OptRFS discards  $r_t$  and  $C_t$  by freeing their memory locations. That is,  $r_t$  and  $C_t$  are used on-the-fly. So before the initialization of the  $(t+1)$ th iteration, the system storage overhead for OptRFS includes the past queries and  $Overlap'_{t,t+1}$ . And  $Overlap'_{t,t+1}$  is sequentially organized. After the search engine completes a sequential scan on

1. if  $t=1$
2.      $C_1 \leftarrow \text{VA\_P1}(Q_1, r_1)$
3.      $R_1 \leftarrow \text{VA\_P2}(Q_1, C_1, R_1)$
4. else
5.      $Overlap'_{t-1,t}, r_t \leftarrow \text{OptRFS}(Q_{t-1}, r_{t-1}, R_{t-1}, C_{t-1})$
6.      $R_t \leftarrow \text{Scan}(Q_t, Overlap'_{t-1,t})$
7.      $C_t \leftarrow \text{VA\_P1}(Q_t, r_t)$
8.      $R_t \leftarrow \text{VA\_P2}(Q_t, C_t, Overlap'_{t-1,t}, R_t)$
9. return  $R_t$ .

**Fig. 3** OptRFS-based relevance feedback

$Overlap'_{t,t+1}$  during the  $(t+1)$ th iteration,  $Overlap'_{t,t+1}$  is then discarded. Notice that OptRFS also predicts the “optimal” query after the first few iterations by using the past prediction errors. If the prediction of “optimal” query is invoked after  $t$ th iteration, then the feedback loop is terminated after its  $(t+1)$ th iteration. That is,  $Overlap'_{t,t+1}$  is the predicted overlap between  $t$ th iteration and final iteration.

### 3.4 The KNN search in relevance feedback

In this subsection, we look at how OptRFS can be easily embedded into an existing KNN algorithm to improve its performance significantly. A universal KNN algorithm can be generalized into two steps (Fig. 3). In the first step, data space is filtered based on certain techniques, and a set of candidates are returned. In the second step, random accesses are performed on the candidates to compute their real distances, and the results are ranked and returned. OptRFS can be easily embedded as follows. Before the first step, a sequential scan is performed on the predicted overlap returned from OptRFS. Interestingly, after sequentially scanning the predicted overlap, an initial set of results can also be computed. And the tighter pruning radius is then passed to the first step. In the second step, random accesses are performed on the candidates excluding those in the predicted overlap, and the results are correspondingly updated.

While OptRFS can be easily deployed in existing KNN search methods, here we choose VA-file’s two-phase algorithm as our example for its effectiveness and simplicity [36]. This also facilitates comparison with the proposal [38] described in Sect. 2.3.1. In OptRFS,  $r'_{t+1}$  is further tightened by considering all the results from early iterations, i.e.,  $r''_{t+1}$  is

the  $K$ th largest distance between  $Q_{t+1}$  and  $R_1, \dots, R_t$  with  $w_{t+1}$ . This provides a possible smaller  $r_{t+1}$  than that in [38].

The following algorithm shows how OptRFS can be embedded into VA-file's KNN search and how OptRFS reduces the random accesses. We denote the standard first and second phase filtering in VA-file as VA\_P1 and VA\_P2. VA\_P1 receives two parameters  $Q_t$  and  $r_t$  in the phase I filtering and returns the candidate set  $C_t$ . VA\_P2 accepts  $Q_t$ ,  $C_t$ ,  $Overlap'_{t-1,t}$  (i.e., candidate set for random accesses),  $R_t$  (i.e., initial result set computed from the predicted overlap), and returns the final results set  $R_t$ .

For the first iteration, a standard VA-file KNN search is performed (lines 1–3). Starting from the second iteration, OptRFS plays the role. It predicts the overlap between two consecutive iterations. At the same time, it also returns  $r_t$  based on the past results (line 5). The predicted overlap is then sequentially scanned and an initial set of results are also computed (line 6). Function VA\_P1 performs phase I filtering by enforcing the  $r_t$  (line 7). Function VA\_P2 then performs phase II filtering by randomly accessing on the candidates excluding those that have been sequentially scanned, and the results are also correspondingly updated (line 8). Notice that the above algorithm does not indicate the function of predicting the “optimal” query. In OptRFS, users can specify after how many iterations this function can be invoked. Once the “optimal” query is predicted, OptRFS then estimates the overlap between current iteration and final iteration, and passes the predicted “optimal” query, denoted as  $Q'_{opt}$ , to functions Scan, VA\_P1, and VA\_P2 for the final process of feedback loop (lines 5–8 except  $Q_t$  is replaced by  $Q'_{opt}$ ).

Comparing with the approach in [38], OptRFS has several distinctive features. First and most interestingly, **repeated random accesses on the same candidates in two consecutive iterations are avoided**. Instead, an efficient scan on the overlap leads to a much faster response. Generally, the **overlap size is much smaller than the candidate size** [38]. Manipulating overlap instead of whole candidate set in last iteration saves both storage and scan overhead. Second, an **initial set of results are computed when the sequential scan is performed on the predicted overlap**. It is potential that the initial set contain some real results. This provides a chance for **phase II to stop earlier**. Let us consider the best case. When the initial set contains all real results and the lower bound of first candidate in phase II is already greater than the  $K$ th actual largest distance, phase II stops immediately without consuming single random access. Third, a **tighter  $r_t$  is computed**, which results in a smaller number of candidates. Fourth, OptRFS is able to **predict the “optimal” query so that the feedback loop can be terminated earlier**. It's noticed that **OptRFS is more effective when there are indeed some considerable overlap between two iterations**. While skipping many feedback iterations improves the performance of feedback search, it

may also affect the result quality. Naturally, two questions immediately follow up: how many percent of candidates in  $C_t$  overlap with candidates in  $C_{t-1}$ , and how many iterations are needed before an accurate “optimal” query prediction can be made? We will see the answers in the next section.

## 4 Experiments

We have implemented OptRFS. In this section, we report results of our experimental study on real image databases.

### 4.1 Experiments setup

All the experiments were performed on a Sun UltraSparc II 450 Mhz (2 CPU) with 1 Gb RAM. We use two real datasets in our experiments.

- **WWW image dataset**: We created one data set extracted from **62,400 WWW images** randomly crawled from over 40,000 web sites. It consists of **159-dimensional color histograms** extracted from these images. The size is about 40 Mb.
- **Corel image dataset**: The Corel image collection contains 68,040 images [1], from which a **32-dimensional HSV Color Histogram feature space** is extracted. The size is about 8.7 Mb.

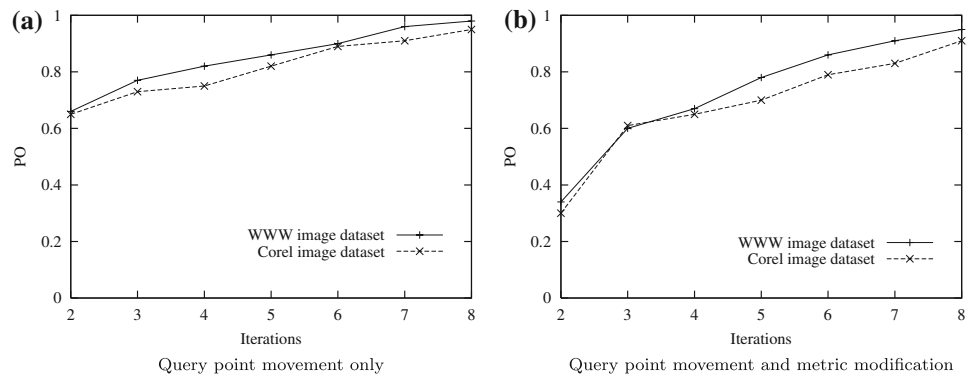
A VA-file is constructed for each image feature space. The implemented VA-file uses 5 bits for each dimension. All results reported are based on 50 queries for 20NN. The queries are randomly selected from the database. We adopted the weighted  $L_2$  norm for similarity measure. By default, every dimension has the same weight. To avoid the subjectivity involved in selecting “good” images to refine the query, by default we choose the top 5 most relevant images as “good” images, where the relevance is identified by the similarity. Meanwhile, user studies were also conducted to further prove our proposals. For each user study, a group of **five students** perform subjective judgement on the ground truth for each query. Each student reports average results on **ten queries**.

To study the effectiveness of our proposal, we consider the following measures:

- (a) **Percentage of overlap—PO**: It is a precondition for OptRFS to know in relevance feedback search how many percent of candidates in the  $(t + 1)$ th iteration have been accessed in the  $t$ th iteration. Intuitively, the larger the PO is, the more random accesses can be saved. Formally, the PO is defined as:

$$PO = \frac{C_t \cap C_{t+1}}{C_{t+1}}.$$

**Fig. 4** Percentage of overlap between two consecutive iterations



- (b) *Precision of overlap prediction—POP*: POP indicates how accurate OptRFS can predict the overlap, and it is formally defined as:

$$\text{POP} = \frac{\text{Overlap}'_{t,t+1} \cap \text{Overlap}_{t,t+1}}{\text{Overlap}'_{t,t+1} \cup \text{Overlap}_{t,t+1}}$$

where  $\text{Overlap}_{t,t+1} = C_t \cap C_{t+1}$ , which is the actual overlap. Obviously, the large the POP, the more effective the OptRFS.

- (c) *Precision of “optimal” query prediction—PQP*: PQP indicates how accurate OptRFS can predict the “optimal” query, and it is formally defined as:

$$\text{PQP} = \frac{R_{\text{opt}} \cap R'_{\text{opt}}}{R_{\text{opt}}}$$

where  $R_{\text{opt}}$  and  $R'_{\text{opt}}$  are the results returned by the “optimal” query and predicted “optimal” query.

- (d) *Ratio of random access saved—RAS*: Since data objects reside in the secondary storage, one of our goals is to know the random accesses saved by OptRFS in the next iterations of relevance feedback by overlap prediction. Note that we do not consider the effects of main memory, including its cache sizes, replacement policies, pre-fetch policies, etc. In  $t$ th iteration, denote the number of randomly accessed candidates of standard VA-file search and OptRFS as  $C_t^{\text{VA}}$  and  $C_t^{\text{OptRFS}}$  respectively. Assume that a random access is ten times more expensive than a sequential scan. For  $t$ th iteration, RAS is then defined as:

$$\text{RAS} = \frac{C_t^{\text{VA}}}{C_t^{\text{OptRFS}} + \frac{\text{Overlap}'_{t-1,t}}{10}}$$

Notice that the scan on the  $\text{Overlap}'_{t-1,t}$  is also included as part of random access cost in OptRFS.

- (e) *Ratio of iterations saved—RIS*: The other goal is to know the number of iterations saved by OptRFS in

feedback loop by “optimal” query prediction. Denote the number of iterations with and without “optimal” query prediction as  $\text{opt}'$  and  $\text{opt}$  respectively, RIS is defined as:

$$\text{RIS} = \frac{\text{opt} - \text{opt}'}{\text{opt}}$$

Obviously, the total number of random accesses in feedback loop can also be reduced due to the reduction of iterations.

- (f) *A comparison study*: Finally, we would like to see how significantly OptRFS can improve feedback search, by considering both overlap prediction and “optimal” query prediction as a whole. It is always more convincing to show the superiority of one method over others by comparing their performances. Here we compare OptRFS with the latest proposal in [38] (denote as tighter VA or TVA for short). To measure the relative improvement, we also use the ratio of random access saved (RAS) for the whole relevance loop, which is defined as:

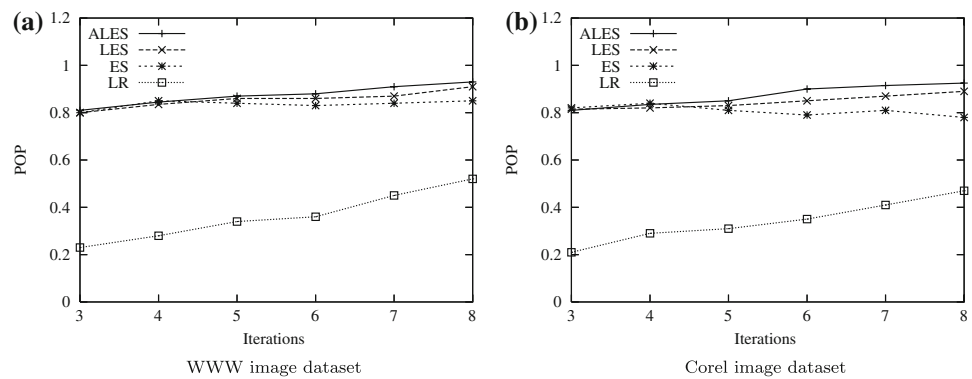
$$\text{RAS} = \frac{C^{\text{TVA}}}{C^{\text{OptRFS}} + \frac{\text{Overlap}'}{10}}$$

where  $C^{\text{TVA}}$  and  $C^{\text{OptRFS}}$  are the total number of randomly accessed candidates from all iterations for TVA and OptRFS respectively, and  $\text{Overlap}'$  is the sum of predicted overlaps from all iterations for sequential scan.

## 4.2 Percentage of overlap

We first confirm the existence of space overlap between two consecutive iterations. Figure 4 shows the percentage of overlap (PO) between the  $i$ th and  $(i - 1)$ th iterations for both WWW image dataset and Corel image dataset, where  $i$  ranges from 2 to 8. Figure 4a shows the results when only the query point movement is considered, while Fig. 4b takes both query point movement and metric modification into account.

**Fig. 5** Precision of overlap prediction



Unsurprisingly, such overlaps exist and the volume of overlap is considerably large. Both figures show that the overlap becomes more significant as more iterations are accomplished. And metric modification causes less overlap, comparing Fig. 4b to 4a. As the number of iteration increases, the PO grows and reaches above 90% in the 8th iteration. Both datasets show similar trends. This unveils one feature of relevance feedback search. The converging process towards the “optimal” setting may slow down as more iterations have been completed. At the beginning of the loop, the query may be far away from the “optimal” one and the amount of query modification could be large. As more good results are returned, the query refinement is expected to be less significant.

This experiment indicates that a large percentage of candidates in current iteration had been accessed before. Since one candidate consumes one random access to retrieve the actual data, a large number of repeated random accesses occur. Saving those random accesses will further speed up the retrieval.

Notice that there is a sharp jump in Fig. 4b when iteration changes from 2 to 3. This indicates that the default *equal weight* setting is too far away from the “optimal” setting. To reduce the prediction error caused by the default weight setting, we do not include the  $w_1$ , the weight setting for the first iteration, in our prediction models. That is, our prediction starts from the third iteration in the following experiments.

#### 4.3 Precision of overlap prediction

In this experiment, we show how accurate the prediction models in OptRFS can achieve for overlap prediction. Figure 5 depicts the precisions of prediction for four methods. For both datasets, we have the following observations. First, ALES achieves linearly increasing and best performance of more than 0.9 after 6th iteration. The next best performer is LES followed by ES who achieves stable performance of about 0.8. This confirms that some noisy queries and local trends exist in the relevance feedback search. Second, in the first few iterations, there is no clear difference between three models. The main reason we believe is due to the extremely

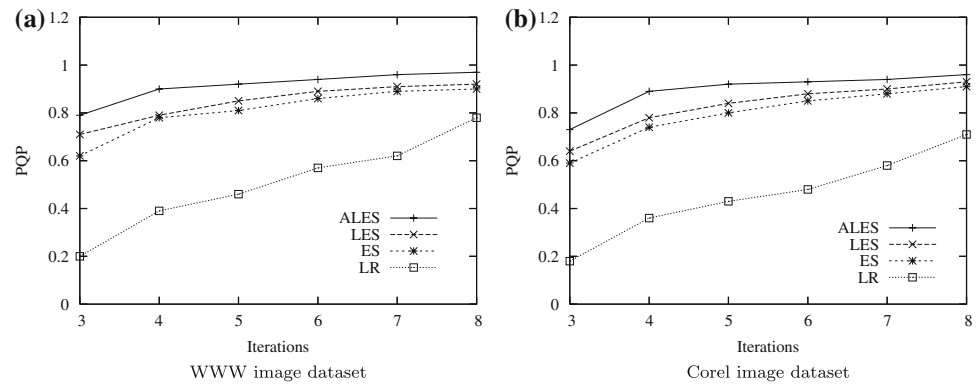
small number of available queries for prediction. As more iterations are executed, the improvements turn to be obvious (last few iterations). This is mainly because that after several iterations, recent queries start converging steadily and forming a local trend towards to “optimal” query. Adaptivity to the prediction errors and noises further distinguish ALES from LES. Notice that when precision is high, little improvement will cause significant improvement on the random access saved. Third and surprisingly, linear regression performs badly although it becomes more effective as iterations go on. This may be caused by the followings. First, from last experiment, it is noticed that the change of query over iterations is not constant. Less is changed in later iterations. However, linear regression assumes the change to be fixed. Second, the number of iterations in relevance feedback is relatively small. Linear regression gives large prediction error when only few data are available. This experiment shows that ALES best matches the features of relevance feedback search. Its performance is linearly increasing, starting from a high level.

#### 4.4 Precision of “optimal” query prediction

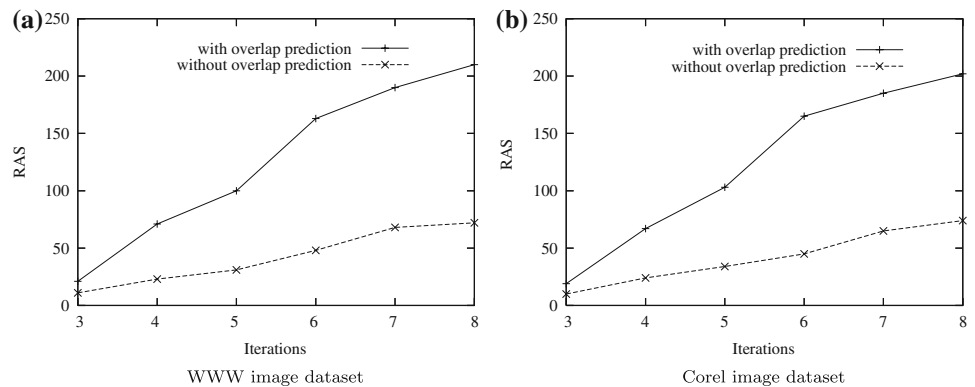
Next we show how accurate the prediction models in OptRFS can achieve for “optimal” query prediction. We conducted this experiment by a user study. The user’s perceived color relevance is used to construct the feedback query. At each iteration, the user manually judges a number of relevant results as “good” images based on his/her perception. Those “good” images are then used to update the feedback query as described in Sect. 2.2. The loop terminates until the user gets “optimal” results based on his/her perception.

Figure 6 depicts the precisions of prediction for four methods. Notice that here the x-axis indicates after how many iterations the “optimal” query prediction is invoked. As we can see, the precisions increase continuously for all methods as more iterations are executed before the prediction. This is expected since more data potentially better suggest the query changing trend. ALES scores the highest precision consistently, followed by LES, ES and LR. For both datasets,

**Fig. 6** Precision of “optimal” query prediction with human perception



**Fig. 7** Ratio of random access saved by overlap prediction



ALES predicts “optimal” query with more than 90% accuracy when the prediction is made after 4th iteration. Hence, after four iterations have been executed, OptRFS is ready to perform “optimal” query prediction. This experiment further confirms the greatest prediction power of ALES in feedback search.

Due to the best performance of ALES in both overlap prediction and “optimal query” prediction, consequently we only use ALES as the prediction model for OptRFS in the following experiments.

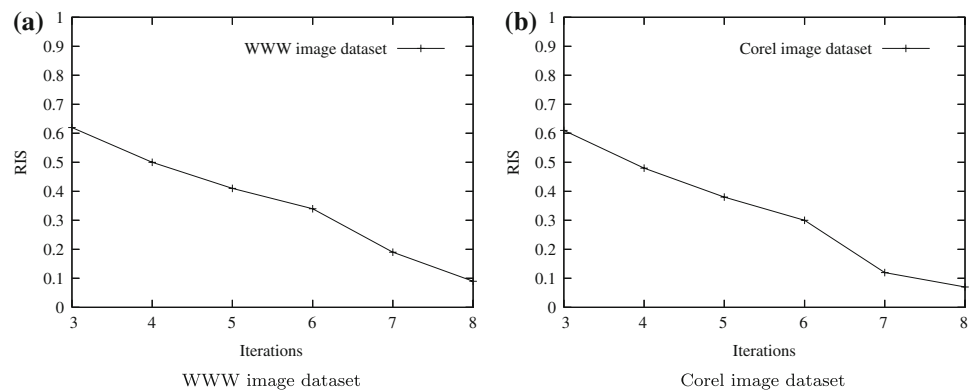
#### 4.5 Ratio of random access saved

In this experiment, we see how effective our OptRFS performs in saving random access by overlap prediction. Recall that OptRFS reduces the number of random accesses by two factors. The first is the much tighter pruning radius generated by considering all previous iterations’ results. The second is the overlap prediction to avoid duplicate random accesses. Here we tested OptRFS with and without overlap prediction function by comparing with the number of random accesses of the standard VA-file search algorithm. Figure 7a shows the results for WWW image dataset. As we can see, as iterations go on, **OptRFS without overlap prediction improves standard VA-file algorithm by more than an order of magnitude,**

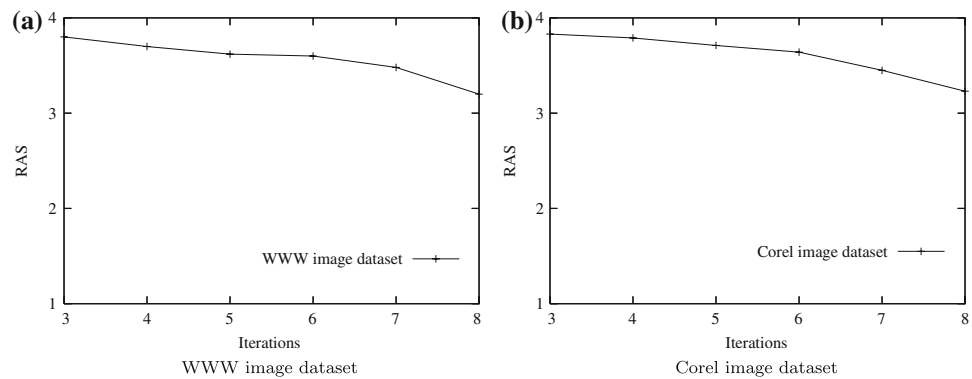
while **OptRFS with overlap prediction further outperforms standard VA-file algorithm by more than two orders of magnitude.** The results on OptRFS without overlap prediction suggest that the number of candidates drops dramatically in next iterations of relevance feedback. And the results on OptRFS with overlap prediction reconfirms that a large portion of candidates in current iteration have been accessed in the pervious iteration, and scan on the predicted overlap further reduces the number of random accesses greatly.

Figure 7b shows similar results for Corel image dataset. However, comparing Fig. 7a and b, the results of WWW and Corel image datasets are surprisingly similar. It might be caused by the following reasons. First, relevance feedback process always converges after a few iterations, regardless of queries, datasets and their feature types. Although different queries may reach convergence in different numbers of iterations, most queries in either dataset finish converging in similar pace. Second, both datasets are color features which exhibit similar skewness. Most images have significant values on very few dimensions only. It is expected that VA-file performs similarly on both datasets. Third, our ALES method is self-adaptive to data distributions by automatically adjusting  $\alpha$  values. That is, ALES considers local distribution/density to self-adjust its predictions. This eliminates the effect of different distributions to certain degrees. On the other hand,

**Fig. 8** Ratio of iterations saved by “optimal” query prediction with human perception



**Fig. 9** A comparison on total improvement with human perception



similar results on two different color datasets also indicate the consistency of our method on effective prediction.

#### 4.6 Ratio of iterations saved

Next, we see how effective our OptRFS performs in saving the number of iterations by “optimal” query prediction. We conducted this experiment by a user study where the user manually judges when relevance loop is terminated (i.e., when the “optimal” results are found) based on his/her perception. Figure 8a shows the results for WWW image dataset. Obviously, RIS drops as the “optimal” query prediction takes place after more iterations have been executed. When the prediction is invoked after three iterations, the number of iterations can be saved by more than 60%. Look back at Fig. 6 which suggests to invoke “optimal” query prediction after four iterations, OptRFS can still reduce the number of iterations by half. Figure 8b shows similar results for Corel image dataset. This experiment confirms the effectiveness of OptRFS by using “optimal” query prediction.

#### 4.7 A Comparison study

Finally, we test OptRFS as a whole by considering both overlap prediction and “optimal” query prediction. Here we

compare OptRFS with the latest proposal TVA in [38] by using the ratio of random access saved (RAS) as defined in Sect. 4.1, where the total numbers of random accesses from all iterations during feedback search for TVA and OptRFS are compared. This experiment was conducted by a user study, where the user determines the relevance of results and the termination of feedback search. Notice that for OptRFS, the loop is automatically terminated once the predicted “optimal” query is executed.

Figure 9 shows the improvement achieved by OptRFS, where the x-axis indicates after how many iterations the “optimal” query prediction is triggered. OptRFS is clearly superior over the scheme in [38], by being able to achieve a significant further reduction in the number of random accesses by times (i.e., the total response time for feedback search can be improved by times). As more iterations are performed before the “optimal” query prediction, the improvement degrades slightly. This is expected since less number of iterations are saved in OptRFS. However, compared with Fig. 8, this deterioration is much slower. The reason behind is clear. At early iterations, both overlap prediction and “optimal” query prediction achieve significant improvement. As more iterations are executed, the improvement achieved by overlap prediction increases, which offsets the deterioration on ratio of iterations saved. When the “optimal” query is

predicted after later iterations, overlap prediction will dominate the overall improvement. In short, OptRFS improves the existing method TVA greatly. It performs better when larger overlaps occur and early “optimal” query prediction is made. In machines whose random access is even more expensive, the improvement is expected to be even higher.

## 5 Conclusion

In this paper, we proposed a new method called OptRFS to reduce the total cost in relevance feedback search. The key is to predict the overlap between two consecutive iterations and the “optimal” query after early iterations. OptRFS studied four prediction models, linear regression, exponential smoothing, linear exponential smoothing and adaptive linear exponential smoothing to forecast the future query (including “optimal” query) and its space to discover the overlap. Our experiments confirm the effectiveness and the improvement over an existing proposal achieved by OptRFS. And the new proposal—adaptive linear exponential smoothing is the most effective alternative for relevance feedback search. It will also be interesting to extend the idea of OptRFS to query processing methods in other areas which consume extensive random accesses.

## References

1. <http://kdd.ics.uci.edu/databases/corelfeatures>
2. Amer-Yahia, S., Case, P., Roelleke, T., Shanmugasundaram, J., Weikum, G.: Report on the db/ir panel at sigmod 2005. *SIGMOD Record* **34**(4), 71–74 (2005)
3. Andoni, A., Indyk, P.: Near-optimal hashing algorithms for near neighbor problem in high dimensions. In: *FOCS*, pp. 459–468 (2006)
4. Baeza-Yates, R., Consens, M.: The continued saga of DB-IR integration. In: *VLDB Tutorial* (2004)
5. Bang, H., Chen, T.: Feature space warping: an approach to relevance feedback. In: *ICIP*, pp. I: 968–971 (2002)
6. Bartolini, I., Ciaccia, P., Waas, F.: Feedbackbypass: a new approach to interactive similarity query processing. In: *VLDB*, pp. 201–210 (2001)
7. Benninga, S.: *Financial modeling*. In: MIT Press (2000)
8. Böhm, C., Berchtold, S., Keim, D.: Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. *ACM Comput. Surv.* **33**(3), 322–373 (2001)
9. Chaudhuri, S., Ramakrishnan, R., Weikum, G.: Integrating DB and IR technologies: what is the sound of one hand clapping? In: *CIDR*, pp. 1–12 (2005)
10. Cheng, J., Wang, K.: Multi-view sampling for relevance feedback in image retrieval. In: *ICPR*, pp. II: 881–884 (2006)
11. Croft, W.B., Schek, H.J.: Introduction to the special issue on database and information retrieval integration. *VLDB J.* **17**(1), 1–3 (2008)
12. El Naqa, I., Yang, Y., Galatsanos, N., Wernick, M.: Relevance feedback based on incremental learning for mammogram retrieval. In: *ICIP*, pp. I: 729–732 (2003)
13. Ferecatu, M., Boujemaa, N.: Interactive remote sensing image retrieval using active relevance feedback. *IEEE Trans. Geosci. Remote Sensing* **45**(4), 818–826 (2007)
14. Franco, A., Lumini, A., Maio, D.: A new approach for relevance feedback through positive and negative samples. In: *ICPR*, pp. IV: 905–908 (2004)
15. Hua, K., Yu, N., Liu, D.: A multiple neighborhood approach to relevance feedback in content-based image retrieval. In: *ICDE* (2006)
16. Ishikawa, Y., Subramanya, R., Faloutsos, C.: Mindreader: querying databases through multiple examples. In: *VLDB*, pp. 218–227 (1998)
17. K. A. Hua, N. Y., Liu, D. Z.: Query decomposition: a multiple neighborhood approach to relevance feedback in content-based image retrieval. In: *ICDE*, p. 84 (2006)
18. Kherfi, M., Ziou, D.: Relevance feedback for cbir: a new approach based on probabilistic feature weighting with positive and negative examples. *IP* **15**(4), 1017–1030 (2006)
19. Kim, D.H., Chung, C.W.: Qcluster: relevance feedback using adaptive clustering for content based image retrieval. In: *SIGMOD*, pp. 599–610 (2003)
20. Lejsek, H., Asmundsson, F.H., Jonsson, B.T., Amsaleg, L.: Scalability of local image descriptors: a comparative study. In: *ACM Multimedia*, pp. 589–598 (2006)
21. Lu, H., Ooi, B.C., Shen, H.T., Xue, X.: Hierarchical indexing structure for efficient similarity search in video retrieval. *IEEE Trans. Knowl. Data Eng.* **18**(11), 1544–1559 (2006)
22. Lu, Y., Hu, C., Zhu, X., Zhang, H., Yang, Q.: A unified framework for semantics and feature based relevance feedback in image retrieval systems. In: *ACM Multimedia*, pp. 31–37 (2000)
23. Lu, Y., Zhang, H., Liu, W., Hu, C.: Joint semantics and feature based image retrieval using relevance feedback. *IEEE Trans. Multimedia* **5**(3), 339–347 (2003)
24. Muneesawang, P., Guan, L.: An interactive approach for cbir using a network of radial basis functions. *IEEE Trans. Multimedia* **6**(5), 703–716 (2004)
25. Oh, S., Chung, M., Sull, S.: Relevance feedback reinforced with semantics accumulation. In: *CIVR*, pp. 448–454 (2004)
26. Rao, Y., Mundur, P., Yesha, Y.: Fuzzy svm ensembles for relevance feedback in image retrieval. In: *CIVR*, pp. 350–359 (2006)
27. Rui, Y., Huang, T.: Optimizing learning in image retrieval. In: *ICCV*, pp. 236–243 (2000)
28. Rui, Y., Huang, T.S., Mehrotra, S.: Content-based image retrieval with relevance feedback in mars. In: *ICIP*, pp. 815–818 (1997)
29. Rui, Y., Huang, T.S., Ortega, M., Mehrotra, S.: Relevance feedback: a power tool in interactive content-based image retrieval. *IEEE Trans. Circuits Systems Video Technol.* **8**(5), 644–655 (1998)
30. Saha, S., Das, A., Chanda, B.: Image retrieval based on indexing and relevance feedback. *PRL* **28**(3), 357–366 (2007)
31. Shao, J., Huang, Z., Shen, H.T., Zhou, X., Li, Y.: Dynamic batch nearest neighbour search in video retrieval. In: *ICDE*, pp. 1395–1399 (2007)
32. Shen, H.T., Ooi, B.C., Zhou, X., Huang, Z.: Towards effective indexing for very large video sequence database. In: *SIGMOD*, pp. 730–741 (2005)
33. Tong, S., Chang, E.: Support vector machine active learning for image retrieval. In: *ACM Multimedia*, pp. 107–118 (2001)
34. de Ves, E., Domingo, J., Ayala, G., Zuccarello, P.: A novel bayesian framework for relevance feedback in image content-based retrieval systems. *PR* **39**(9), 1622–1632 (2006)
35. Wang, L., Gao, Y., Chan, K., Xue, P., Yau, W.: Retrieval with knowledge-driven kernel design: an approach to improving svm-based cbir with relevance feedback. In: *ICCV*, pp. II: 1355–1362 (2005)
36. Weber, R., Schek, H., Blott, S.: A quantitative analysis and performance study for similarity search methods in high dimensional spaces. In: *VLDB*, pp. 194–205 (1998)

37. Wu, L., Faloutsos, C., Sycara, K., Payne, T.R.: Falcon: feedback adaptive loop for content-based retrieval. In: VLDB, pp. 297–306 (2000)
38. Wu, P., Manjunath, B.: Adaptive nearest neighbor search for relevance feedback in large image datasets. In: ACM Multimedia, pp. 87–98 (2001)
39. Yin, P., Bhanu, B., Chang, K., Dong, A.: Integrating relevance feedback techniques for image retrieval using reinforcement learning. PAMI **27**(10), 1536–1551 (2005)
40. Zhang, H., Chen, Z., Li, M., Su, Z.: Relevance feedback and learning in content-based image search. World Wide Web **6**(2), 131–155 (2003)
41. Zhou, X.S., Huang, T.S.: Relevance feedback in image retrieval: a comprehensive review. Multimedia Systems **8**(6), 536–544 (2003)