FALCON: Feedback Adaptive Loop for Content-Based Retrieval

Leejay Wu Christos Faloutsos Katia Sycara Terry R. Payne

Carnegie Mellon University

Abstract

Several methods currently exist that can perform relatively simple queries driven by relevance feedback on large multimedia databases. However, all these methods work only for vector spaces; that is, they require that objects be represented as vectors within feature spaces. Moreover, their implied query regions are typically convex. This research paper explains our solution.

We propose a novel method that is designed to handle disjunctive queries within metric spaces. The user provides weights for positive examples; our system "learns" the implied concept and returns similar objects. Our method differs from existing relevance-feedback methods that base themselves upon Euclidean or Mahalanobis metrics, as it facilitates learning even disjunctive, concave models within vector spaces, as well as arbitrary metric spaces. In addition, our method is completely example-driven, and imposes no requirements upon the user for other aspects such as feature selection.

Our main contributions are two-fold. Not only do we present a novel way to estimate the dissimilarity of an object to a set of desirable objects, but we support it with an algorithm that shows how to exploit metric indexing structures that support range queries to accelerate the search without incurring false dismissals. Our empirical results demonstrate that our method converges rapidly to excellent precision/recall, while outperforming sequential scanning by up to 200%.

Proceedings of the 26th VLDB Conference, Cairo, Egypt, 2000.

1 Introduction

As the size and diversity of multimedia databases increase, so does the potential complexity of queries. Query concepts such as, "Return all video clips showing any US President speaking", may be be easy to specify, but difficult to perform. Unless comprehensive annotations are provided with every database entry, a multimedia database might be forced to rely on nothing more than query-by-example. In addition, measuring similarity between images then becomes an issue. Should one resort to commercial packages, the result may easily yield – at best – a metric space, as neither the features nor their relevance are known, where a metric space is defined via a distance function which obeys symmetry and the triangle inequality [4]. In this domain, query models demanding vector spaces fail completely. This paper presents a novel approach, *FALCON*, which allows easy specification of complex queries, within both vector and metric spaces, for multimedia and traditional databases.

Many retrieval methods represent database records as vectors, with the assumption that the closer are two vectors, the more similar are the corresponding records [9]. Distance functions are chosen to score dissimilarity between points within this space. Thus, one may query a database by performing either a range query or a nearest-neighbor search relative to a single point or hyper-surface within this space.

However, users may wish to perform more complicated queries. On a real-world database, a user may wish to retrieve a class of objects that does not map to a contiguous region according to the similarity metric. For instance, the aforementioned query regarding US Presidents speaking arguably could map to numerous rather disparate images due to change of venue and personnel – and if the image distances are generated via a black-box function, we now have an unusual query in a metric space. Finding similar objects within this space is related to clustering, which can be done

in metric spaces as per [16].

FALCON combines distances and incorporates user feedback in such a way as to "learn" the nature of such queries. This *aggregate dissimilarity* model applies to metric data sets, since it does not use any information about the data itself aside from that returned by a pairwise distance metric. Our experiments demonstrate that FALCON can provide high-quality results in terms of precision and recall after 5 to 20 iterations.

The paper is organized as follows: Section 2 summarizes a sample of existing related systems; the mechanisms underlying FALCON are presented in Section 3; the experiments and corresponding results are detailed in Section 4, and the subsequent analysis is presented in Section 5. The paper concludes with Section 6.

2 Related Work

A number of systems such as the following have been developed to handle example-based queries. Note they handle neither unusual disjoint queries, nor arbitrary metric spaces.

- Rocchio's relevance feedback mechanism [11], which generates hyper-spherical isosurfaces in feature space.
- MARS (Multimedia Analysis and Retrieval System) [10, 12, 13], which includes a query expansion model that can weight features from multiple objects. However, MARS limits the sums of weights, so that when using fixed thresholds it becomes difficult to specify a disjunctive query in which being similar to any one of the examples is sufficient to be considered good.
- MindReader, which uses the Mahalanobis distance to allow arbitrarily oriented ellipsoids [6]. The Mahalanobis distance $M(\vec{x}, \vec{y})$ is defined as $(\vec{x} - \vec{y})^T \times \mathbf{M} \times (\vec{x} - \vec{y})$ where $\vec{x}$ and $\vec{y}$ are n-dimensional column vectors and $\mathbf{M}$ is a $n \times n$ matrix. This corresponds to a weighted Euclidean distance, and permits effective rotation of the axes, but requires many examples to calculate the covariance matrix.

The original assumption behind the earliest systems is that there exists an ideal query vector. One can then try to determine both this vector and the optimal relative weights of the axes. This dependence on a vector space prevents these methods from generalizing to metric spaces.

Later systems, such as current versions of MARS, have query expansion models which permit actual elaboration by weighting the relative importance of different features of multiple positive examples [10]. However, one should note that MARS has been specialized for image databases with features, whereas MindReader generates isosurfaces consisting of single hyper-ellipsoids, and therefore does not handle disjunctive queries.

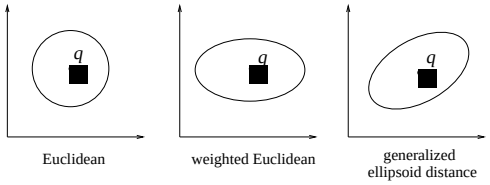


Figure 1: Generic isosurfaces for three previous methods.

See Figure 1 for an illustration of these types of isosurfaces.

Fox and Salton used the L_p metric to surpass fuzzy Boolean methods in the domain of text retrieval, replacing the use of minimum and maximum functions. Our objective is similar, but more complex; we wish to model arbitrarily disjunctive example-based queries in unbounded metric spaces using relevance feedback but lacking specific features [14].

FALCON does not rely on vector spaces, require user input beyond that of relevance feedback and examples, or sacrifice the ability to use disjunctive queries that correspond to arbitrary groupings in metric spaces. In addition, advanced indexing methods can be used to significantly speed up the search process. Spatial indexing methods such as R-trees [5] and R*-trees [1, 2] would serve if we were to limit ourselves to vector domains; M-trees [3] provide fast range-queries in general metric spaces.

3 Proposed Method

This section proposes the underlying mechanisms and assumptions made by FALCON. Table 1 lists the notation used in this section.

Let $\mathcal{X}$ be the set of objects in our metric data set. Then, let $d : \mathcal{X} \times \mathcal{X} \rightarrow \Re$ be the provided distance metric that defines our metric space.

To start each query, the user specifies at least one "desirable" example that is representative of the intended query. This set of "good" examples is denoted by $\mathcal{G}$. Thus:

Problem 1: Query by Multiple Examples	
Given	$\mathcal{X}$, a metric data set d, its pairwise distance metric $\mathcal{G} = \{g_i\}$, the set of "good objects"
Find	Other desirable objects in $\mathcal{X}$ that are similar to $\mathcal{G}$.

Symbol	Description
$\mathcal{X}$	The set of objects in our data set.
x	Any single object from $\mathcal{X}$.
$\mathcal{G}$	The current set of user-specified "good points".
g_i	A member of $\mathcal{G}$.
$D_{\mathcal{G}}$	The aggregate dissimilarity function based on $\mathcal{G}$.
$D_{\mathcal{G}}(x)$	The aggregate dissimilarity value for object x to the current good set $\mathcal{G}$.
α	A constant that influences how $D_{\mathcal{G}}$ behaves.
d	The pairwise dissimilarity function.

Table 1: Notation used within this paper.

We propose solving Problem 1 by determining a scoring function that models the user's query. Namely, we seek a function $D_{\mathcal{G}} : \mathcal{X} \rightarrow \Re$ based on $\mathcal{G}$, such that this function varies inversely with the desirability of x.

3.1 Proposed "Aggregate Dissimilarity" Function

Once we find a function that fulfills our main requirement – ranking objects inversely according to their apparent desirability as compared to $\mathcal{G}$ – the problem of finding relevant objects reduces to that of sorting. We define such a function as follows:

Problem 2:	**Aggregate Dissimilarity**
Given	$x \in \mathcal{X}$, a candidate $\mathcal{G} = \{g_i\}$, the set of user-selected "good objects" d, pairwise distance metric for $\mathcal{X}$
Determine	$D_{\mathcal{G}}(x)$, its aggregate dissimilarity

We propose that the FALCON aggregate dissimilarity, $D_{\mathcal{G}}(x)$ be computed as the α^{th} root of the arithmetic mean of the α^{th} powers, of the pairwise distances, as expressed by

$$(D_{\mathcal{G}}(x))^{\alpha} = \begin{cases} 0 \text{ if } (\alpha < 0) \bigwedge \exists i \, d(x, g_i) = 0 \\ \frac{1}{k} \times \sum_{i=1}^{k} d(x, g_i)^{\alpha} \text{otherwise} \end{cases} \quad (1)$$

If the value of α is very high, the highest distance will have the largest impact on $D_{\mathcal{G}}(x)$, while the reverse is true for very low values of α.

Note that Equation 1 mimics a fuzzy OR if $\alpha < 0$, and a fuzzy AND if $\alpha > 0$.

Since it is not obvious which values of α are the most suitable, we empirically compare results for various values of α. Our upcoming experiments show that $\alpha = -5$ is a reasonable choice for the queries and datasets we selected; however, in specific applications some tuning with subsets may be appropriate.

Were a user to select parrot photos via query-by-example, the following might ensue.

1. The user chooses a data set, $\mathcal{X}$, consisting of bird photos. This data set is paired with a pairwise distance metric d, which relies primarily on coloration when comparing images.

2. The user first chooses the image of a popular green-and-red parrot. This becomes the first member of $\mathcal{G}$.

3. FALCON returns the best matches, according to $D_{\mathcal{G}}$; note that grey parrots might be ranked below cardinals – which are mostly red – and peacocks – which tend to be green.

4. The user adds a picture of a grey parrot to the good set, which alters the aggregate dissimilarity function $D_{\mathcal{G}}$. Other images of grey parrots will now be considered more relevant to the query.

5. Repeat as desired; depending upon the data and the metric, convergence may be fairly rapid.

One useful generalization of the FALCON distance is to allow for weights. Unlike a distance "combination" function that only uses the minimum distance, the FALCON distance is easily modified to accept a positive w_i term for numerical feedback from a user. The proposed extension takes the form:

$$(D_{\mathcal{G}}(x))^{\alpha} = \frac{1}{\sum_{i=1}^{k} w_i} \cdot \sum_{i=1}^{k} w_i (d(x, g_i))^{\alpha} \quad (2)$$

This has the effect that distance to the favored objects is penalized less if α is negative. We do *not* raise the weights to the α^{th} power, as we believe that this – which would have the opposite effect – would be much less intuitive. In addition, we retain the influence of all pairwise distances between candidate and examples in all cases except that of an exact match, unlike a pure minimization function.

3.2 Speed and Completeness

There is the obvious question of speed. Sequential scanning would entail computing aggregate dissimilarity via Equation 1 or 2 for each element of the database, which would require $\Theta(nk)$ given n objects in $\mathcal{X}$ and k in $\mathcal{G}$. Our objective is to return the same

results as a sequential scan, with less work on average per search.

Indexing structures which support fast range queries can be used to achieve significant speed-ups. First, we shift focus to the aggregate dissimilarity version of the range query:

Problem 3: Range Query by Multiple Example	
Given	$\mathcal{X}$, the database $\mathcal{G} = \{g_i\}$, the set of "good objects" ϵ, a threshold d, pairwise distance metric for $\mathcal{X}$
Find	Q, such that $Q = \{x : x \in \mathcal{X} \wedge D_{\mathcal{G}}(x) < \epsilon\}$ quickly

Theorem 1 *Consider Problem 3 above. Executing $k = |\mathcal{G}|$ separate range queries with threshold ϵ, will yield k sets, the union of which forms a superset of the actual answer. No object will be falsely discarded as a candidate by such a procedure.*

The proof is omitted for the sake of brevity [15].

This theorem shows that we can use existing indexing methods without fear of false dismissals. A post-processing step can then check every member of the union and discard any whose actual aggregate dissimilarity exceeds the threshold.

The question remains of selecting a suitable ϵ. One method would be to allow the user to flag examples that they consider to be about as dissimilar as they would accept; then, an ϵ can be generated as the maximum $D_{\mathcal{G}}$ for these borderline cases.

4 Experimental Setup

The core parts of FALCON were implemented in C, C++ and Perl, and tested on Intel Pentium II^{TM} workstations under Linux.

We tested FALCON with the intent of answering five central questions.

> (a) Does FALCON "learn" to model concave and disjunctive queries?
> (b) Does FALCON provide satisfactory precision/recall?
> (c) Does FALCON's distance function rapidly converge?
> (d) What is a suitable value of α?
> (e) How fast is FALCON?

Four data sets were used during the experiments, each with exactly one query. Two of the data sets were synthetic, and two consisted of real data. We used the standard Euclidean distance as the pairwise distance metric; with the $2D\_20K$ data set, we also experimented with L_∞.

Vector data sets were used primarily due to ease of generating consistent and objective feedback results; the FALCON system never examined the vectors themselves.

2D_50K: This synthetic data set consists of 50,000 points in 2-dimensional Cartesian space, randomly distributed approximately uniformly within the axis-aligned square (-2,-2) - (2,2).

2D_20K: This synthetic data set was generated using identical rules to that of the 2D_50K data set, but consists only of 20,000 points.

PEN: This database was obtained from the UCI repository [8], and consists of objects that correspond to handwritten digits, with features being the classification of each and the coordinates of spatially resampled points. We used the existing split of 3498 objects in the training set and 7494 in the test set.

STOCKS: Daily closing prices for up to five year periods were collected for 51 stocks from Yahoo's online quote server. They were split into 1856 non-overlapping vectors of length 32, which were then processed via the discrete wavelet transform (DWT). All 32 coefficients for each vector were used for L_2 distance computations.

4.1 Queries

One implicit query was associated with each data set. Queries were reflected via a seed set of five initial "good" objects, and appropriate feedback for each data item. A subset of each data set was selected as training sets, used for query refinement; the rest of the data served only for evaluation. It may in fact be feasible to implement a hierarchical view of a database – perhaps traversing something like an M-tree – in order to allow the user to easily select examples without wading through an entire database at once [3]. In addition, we varied α as there was no *a priori* reason to believe that one particular value would be optimal.

RING: Vectors within the 2D_50K data set were marked as positive examples if and only if they were between 0.5 and 1.5 units from the null vector, inclusive. The training set consisted of 1,000 vectors. 431 in the subset and 19,734 in the full set met this criterion. The five seeds were vectors of magnitude 1.4, with counter-clockwise angles from the positive X axis of 0, 72, 144, 216, and 288 degrees.

This query to tests performance on a contiguous, but non-convex set.

TWO_CIRCLES: Vectors within the 2D_20K data set were marked as positive examples if and only if they were within 0.5 units of either (-1,-1) or (1,1); 1899 points qualified. The subset consisted of 1000 points randomly drawn from the full set, including 99 positive examples. The seeds were randomly generated from within the two circles.

This query tests performance on a disjunctive set.

PEN: Vectors within the PEN data set that were classified as 4's were flagged positive. The existing training and test partition was used. 364 vectors in the training set and 780 in the full set were positive instances. The five seeds were drawn randomly from the positive instances in the training set.

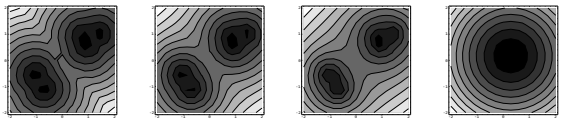
This query tests performance on real data.

STOCKS: Vectors within the STOCKS data set were marked as positive examples if and only if the slopes of their least-squares linear approximations were within the range -0.02 to 0.02. The training set consisted of 500 examples, of which 153 were positive. 540 in the full set were flagged positive. The five seeds were the vectors of DWT coefficients of five perfectly flat lines with y-intercepts of 8, 40, 80, 200 and 400.

This query also tests performance on real data.

4.2 Evaluation Methodology

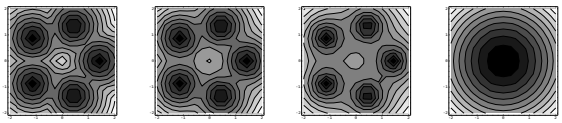
With each data set and its paired query, we tested with the following values of α: $-\infty$, which scores by minimum distance; -100, which approximates that; -10; -5; -2; 2; and 5.



$\alpha = -\infty$ $\alpha = -5$ $\alpha = -2$ $\alpha = 2$

Figure 2: Contour plots for the seeds of the TWO_CIRCLES query.

Figure 2 shows contour plots for four values of α in the TWO_CIRCLES. Note that $\alpha = 2$ reflects only the center of the "good set", even on the TWO_CIRCLES query, and hence is expected to fail on such a disjunctive query.



$\alpha = -\infty$ $\alpha = -5$ $\alpha = -2$ $\alpha = 2$

Figure 3: Contour plots for the seeds of the RING query.

Figure 3 shows contour plots for four values of α in the RING. Again, $\alpha = 2$ stands out with contours that are not going to rank the points very well.

FALCON ranks objects by computed score, which permits evaluation via precision/recall. We define recall and precision as follows.

For arbitrary n, consider the n top-ranked objects. Let there be p positive examples among them, out of P positive examples total. Then recall is $\frac{p}{P}$, and precision is $\frac{p}{n}$. We can compute precision for any given level of recall for a full ranking by choosing n appropriately.

With each combination of a query and a value of α, we used the following procedure:

- Start with the seeds as the "good set". Compute precision/recall values over the full set. This gives us a baseline that varies only with the contours generated by α.
- Repeat the following as needed:
  - Find the top twenty which have not yet been picked for feedback. Twenty is arbitrary, but plausible.
  - Add any newly-found positive examples into $\mathcal{G}$.
  - Repeat the precision/recall procedure on the full set.

4.3 Speed

We also ran range-query speed tests. The query and data involved were that of TWO_CIRCLES as described previously. For the indexing structure, we used M-trees [3]. Sequential scanning served as a baseline with which to compare the method described in Theorem 1. We then measured the individual effects of threshold and number of seeds on elapsed time and computational cost. Note that the elapsed time includes everything done on a per query basis, including all range queries as well as the time required for merging and verifying the results.

5 Results and Discussion

We present our analysis of the results, organized with regard to the five central questions.

The following notes apply to the various graphs. For all the figures marked "Precision versus Recall", such as Figures 4, 6, and 9, one line is plotted per charted iteration. Not all iterations were charted, for purposes of readability. Each line is drawn with ten points, each of which shows precision at one level of recall from 10% to 100% at 10% increments. Thus, the general trend of the lines tracks the progress of FALCON as feedback is provided on increasing numbers of points.

Any figure that tracks "Precision versus Iterations", such as Figures 7 and 8, instead focuses on precision at one level of recall, 40%, for multiple values of α. Positive slopes indicate positive progress. 40% was arbitrarily chosen as a level at which it is non-trivial, but also not extremely difficult, to provide good precision.

Figures 10 and 11, labelled "Precision at Multiple Levels of α" also track precision, but after 5 and 20 iterations. The first shows precision at 40% recall; the second, average precision based on all 10 levels of recall.

5.1 Concave and Disjunctive Queries

The RING query is concave, but contiguous; the TWO_CIRCLES query is disjunctive. As one sees in Figure 4, FALCON can handle either data set with high levels of precision versus recall. This is a substantial improvement over existing methods such as MARS and MindReader, which would have difficulty with these two.

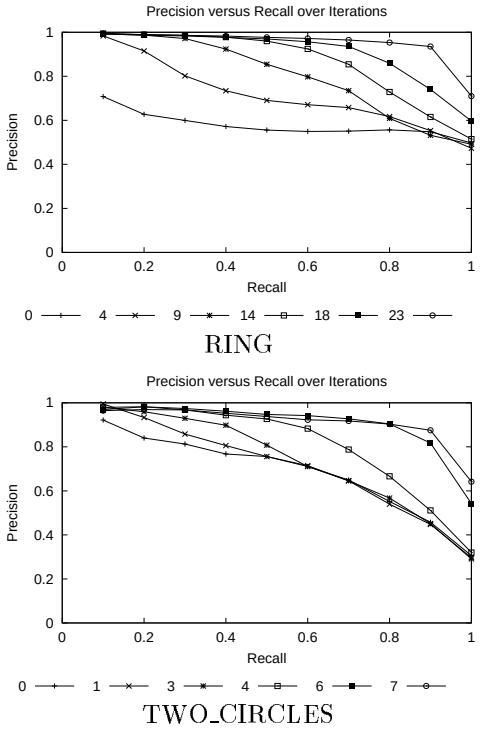


Figure 4: Precision versus recall for $\alpha = -5$, for RING and TWO_CIRCLES, using Euclidean pairwise distance.

The numbers in the legend indicate the number of feedback iterations to achieve that level of progress. We observe that in both cases, precision largely stabilizes after the first several iterations – especially for TWO_CIRCLES.

The success on the TWO_CIRCLES query is particularly notable because that set is completely disjunctive; there are two distinct regions of points that deserve high rankings, separated by points that do not. To further test the system, we ran the same query substituting L_∞ for the Euclidean distance as pairwise metric d; Figure 5 shows the resulting precision-recall.

5.2 Quality of Results on Real Data

Both the PEN and STOCKS queries are reasonable queries on real data. Consequently, FALCON's performance on these, as shown in Figure 6, is relevant to any question as to whether FALCON "works" on real data.

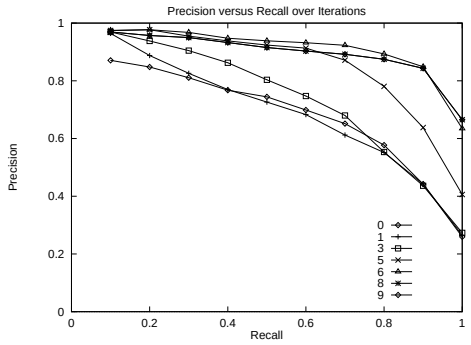


Figure 5: Precision versus recall for $\alpha = -5$, TWO_CIRCLES, and L_∞ as distance metric d

We note that, for the PEN query, convergence at all levels of recall below 100% is rapid; the rest of the iterations after the 4^{th} recall do not add precision except at the highest level of recall. The pattern with the STOCKS query is more interesting, with large gaps between the lines. One plausible explanation is that the positive examples were not evenly distributed in vector space, but instead clustered. The first member of a cluster to be added to $\mathcal{G}$ would immediately cause everything nearby to move upwards in the rankings. This is particularly plausible given that many of the stock vectors were consecutive slivers from the same stock over time, and therefore may have had similar properties.

Figure 6 shows that FALCON provides good precision at almost all levels of recall. In particular, it provides perfect precision at most levels of recall for the PEN data set and query, identifying objects that correspond to 4's. Note also that FALCON provides good precision on the RING and TWO_CIRCLES queries, as shown in Figure 4, as cited above.

5.3 Speed of Convergence

Figures 7 and 8, show precision at 40% recall versus the number of iterations. FALCON can quickly attain high levels of precision over high levels of recall, even with an early $\mathcal{G}$ that has not yet grown to include many of the "good points" in even the training set.

The STOCKS query is unlike other queries. Here, there are very wide gaps in performance until 12 iterations have elapsed; the others show more continuous gains.

Depending upon the complexity of the query and the data set, progress can be very fast (such as in the PEN data set), or slower (as in the RING data set). For 40% recall, precision exceeds 90% after only 4 to 11 iterations for $\alpha = -5$ on all four queries.

There can be overfitting, as seen in the TWO_CIRCLES data set. The law of diminishing returns applies to increasing $|\mathcal{G}|$, suggesting that one does not need to maximize $|\mathcal{G}|$ to attain near-optimal

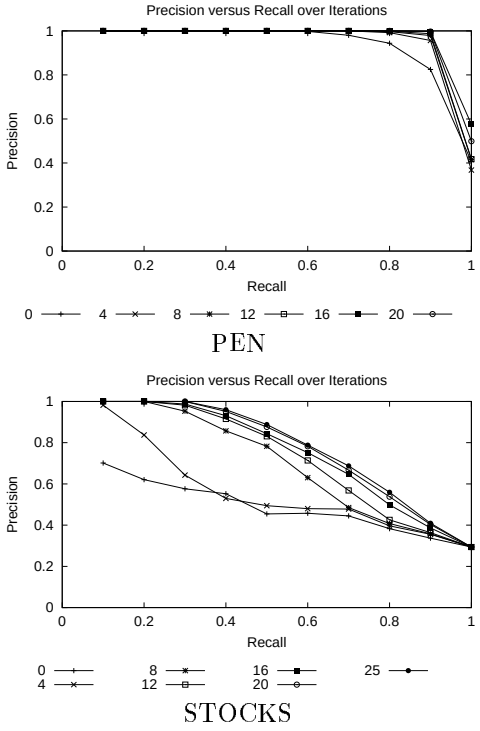


Figure 6: Precision versus recall for $\alpha = -5$, for PEN and STOCKS.

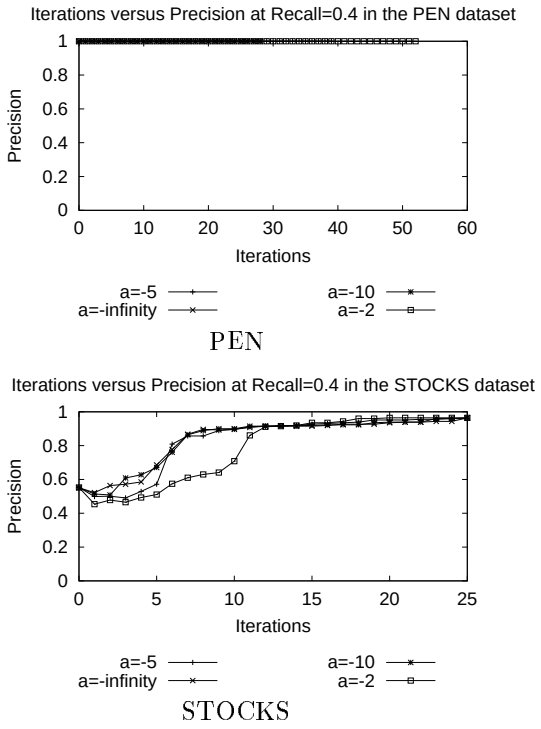


Figure 7: Precision versus iterations at recall=40%.

levels of precision, and therefore sampling can be reasonable.

5.4 Optimal Value of α

We hypothesized that $\alpha = -5$ would be a reasonable choice from our set of possible α's. We found that $\alpha = -100$ generally performed worse due to numerical precision problems, and therefore will not be discussed further.

Figure 9 show precision versus recall for the TWO_CIRCLES data set, using $\alpha = -2$ and $\alpha = -10$. As usual, the numbers in the legend indicate iterations. In both cases, all good points in the sample were added to $\mathcal{G}$ in no more than 9 iterations of feedback. We also note that the progress in precision over multiple levels of recall was fairly steady throughout.

Precision was generally similar for all negative values of α, especially once $\mathcal{G}$ had reached its maximum size. With positive values of α, precision was quite low at most levels of recall; see Figures 10 and 11 for some results.

The first figure shows precision at 40% recall for each level of tested α; each line tracks the precisions yielded by one particular value of α for the different queries. This allows us to compare α both early and late in the process. In the case of TWO_CIRCLES, whose $\mathcal{G}$ stabilized in fewer than 20 iterations, we used the final results. Observe that the differences among the negative values of α largely disappear by the 20^{th} iteration of feedback.

As we can see in the latter figure, difference in average precision over all levels of recall after 20 iterations are a bit more marked than at 40%, but again the negative values of α yield fairly similar average precision.

We still favor $\alpha = -5$, as it provided good performance as empirically observed, and in no case is it significantly inferior. Similar values appear to yield similar results. This may be due to the limited range of values tested, but also reflects the fact that this value permits a significant amount of flexibility, in that this corresponds to a quite fuzzy OR.

5.5 Speed

Empirically, we have found that merging the results of k separate range queries as per Theorem 1 is satisfactory in terms of both the number of distance computations, and the elapsed time. Some of these results may be noted in Figures 12 and 13.

The two graphs in Figure 12 compare the elapsed (real) time and the pairwise distance computation costs incurred in searching the TWO_CIRCLES training data set for points within a variable aggregate dissimilarity threshold from the seeds. These results demonstrate that the k range queries can be merged into one.

We note that the cross-over point where sequential scan performs as quickly as merging occurs at a thresh-

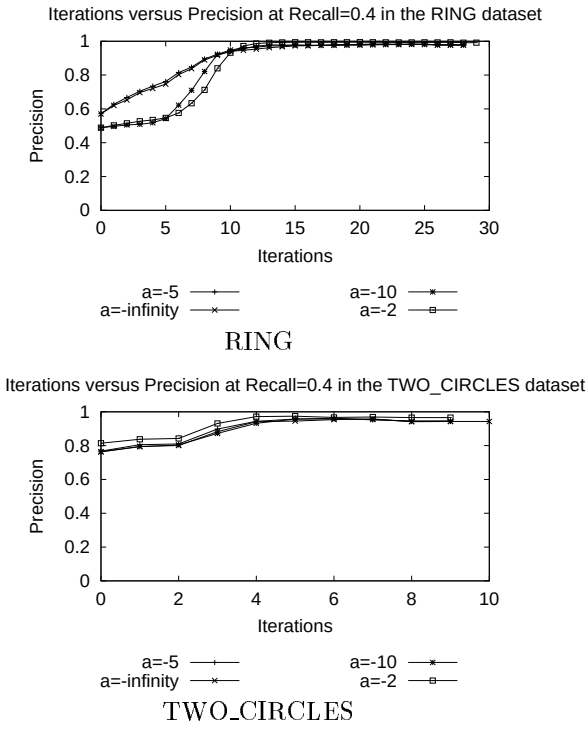


Figure 8: Precision versus iterations at recall=40%

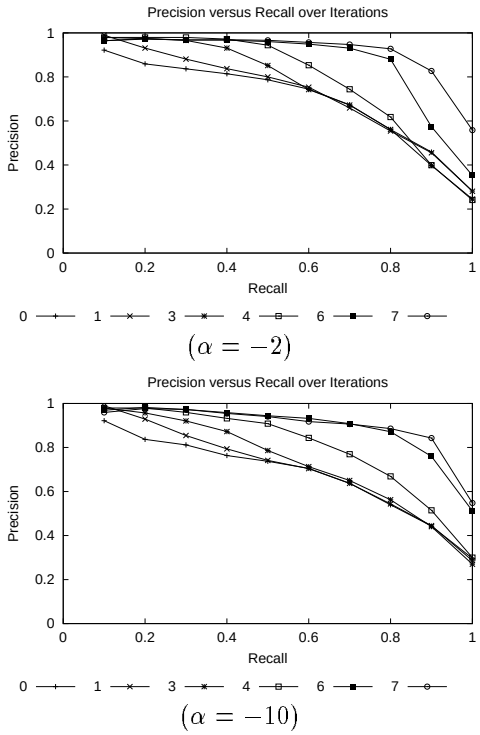


Figure 9: Precision versus recall on the TWO-CIRCLES data set.

old of approximately 0.7; such a threshold accepts approximately 20% of the database. It is our belief that users of large interactive databases will rarely be interested in queries of such scope, and thus the merging method is worthwhile.

As shown in Figure 13, both measurements of performance cost appear to scale with the number of seeds. These results are taken without caching the distance computations from previous searches; these searches were all independent. These two graphs compare the elapsed (real) time and the pairwise distance computation costs incurred in searching the TWO_CIRCLES training data set for points with aggregate dissimilarity less than or equal to 1, varying the number of seeds.

The results for the same test, but with the underlying distance function $d = L_\infty$, are very similar and are not presented here [15].

No direct experimental comparisons are shown with previous methods. This is because our queries were specifically designed to include queries of disjunctive and other highly non-convex behavior in general metric spaces, and thus previous methods simply do not apply.

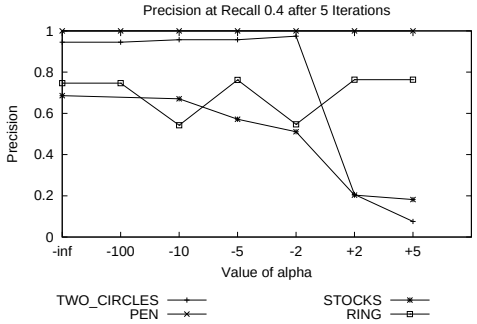
6 Conclusions

We proposed a method to handle queries by multiple examples, on arbitrary vector or metric databases.

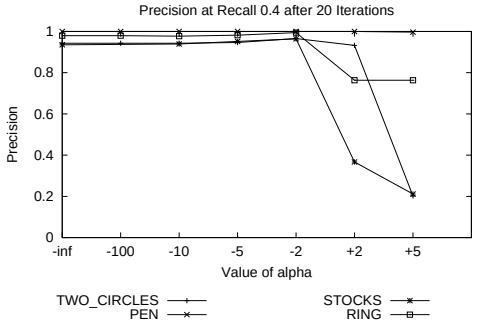
Our method applies to general metric spaces, as the distance combination function depends on only the pairwise distances and not the actual nature of the data. In addition, it handles disjunctive and concave queries that are fundamentally *impossible* for traditional relevance feedback methods. Using our method, a user could, without any domain-specific query language, specify a disjunctive query merely by labelling as "good" objects representative of the different classes. We argue that this combination of general applicability, power and ease-of-use makes this method more valuable than other existing systems today.

The heart of our method is the FALCON aggregate dissimilarity, or $D_{\mathcal{G}}$, which is able to "learn" disjunctive queries via relevance feedback. Additional contributions include the following:

- Theorem 1, which shows that we can use indexing structures that support range queries, to speed up our search, guaranteeing zero false dismissals.
- Experiments on real and synthetic data, that show that the proposed method ("FALCON") achieves good precision and recall. For instance, with all queries, $\alpha = -5$ yielded at least 80% precision at 50% recall with 10 iterations.
- Experiments that show that FALCON needs at most 10 feedback iterations to reach high precision/recall, and will reach a "steady state" for

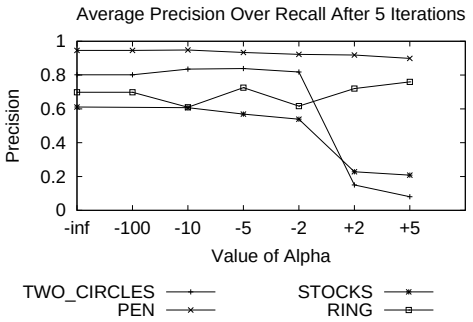


After 5 Iterations

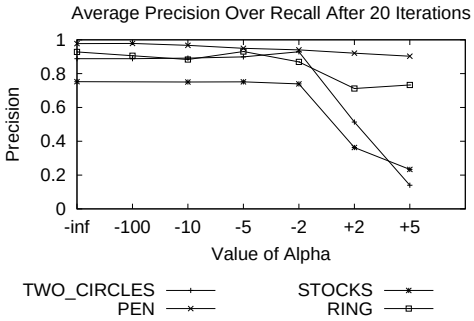


After 20 Iterations

Figure 10: Precision at 40% recall



After 5 Iterations



After 20 Iterations

Figure 11: Average precision at multiple values of α.

all levels of recall below 100% within approximately 20 iterations. Beyond 20 iterations, minor progress is made for 100% recall.

- Experiments that show that a good range for α includes -10 to -2; on many queries, the sensitivity of the performance on α is low.
- Experiments that show that we can use the method described by Theorem 1 to gain up to 200% observed performance improvements versus sequential scanning.

Possible avenues for future work include using $D_{\mathcal{G}}$ for other data mining tasks, such as classification and representative selection [7].

Acknowledgements

This research was partially funded by the National Science Foundation under Grants No. IRI-9625428, DMS-9873442, and IIS-9910606; also, by the National Science Foundation, ARPA and NASA under NSF Cooperative Agreement No. IRI-9411299; DARPA/ITO through Order F463, issued by ESC/ENS under contract N66001-97-C-851; and the Office of Naval Research Grant N-00014-96-1-1222. Additional funding was provided by donations from NEC and Intel. Views and conclusions contained in this document are those of the authors and should not be interpreted as representing official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or of the United States Government.

All trademarks are property of their respective owners.

References

[1] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-tree: An efficient and robust access method for points and rectangles. *ACM SIGMOD*, pages 322–331, May 23-25 1990.

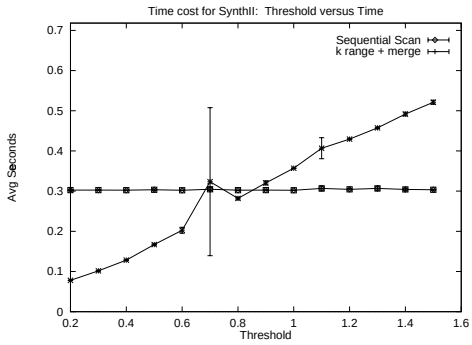
[2] Thomas Brinkhoff, Hans-Peter Kriegel, and Bernhard Seeger. Efficient processing of spatial joins using R-trees. In *Proc. of ACM SIGMOD*, pages 237–246, Washington, D.C., May 26-28 1993.

[3] Paolo Ciaccia, Marco Patella, and Pavel Zezula. M-tree: An efficient access method for similarity search in metric spaces. *VLDB*, pages 426–435, 1997.

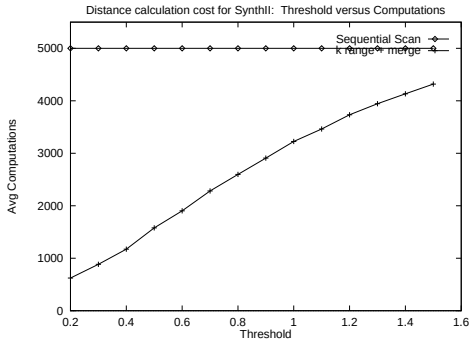
[4] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis.* John Wiley and Sons, 1973.

[5] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *Proc. ACM SIGMOD*, pages 47–57, Boston, Mass, June 1984.

[6] Yoshiharu Ishikawa, Ravishankar Subramanya, and Christos Faloutsos. Mindreader: Querying

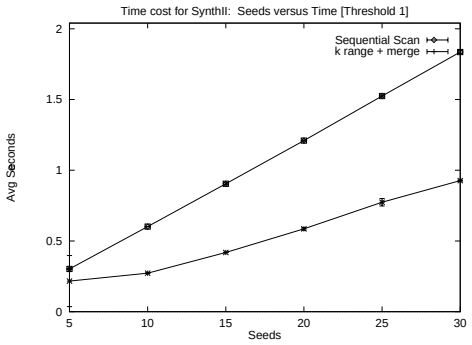


Elapsed Time

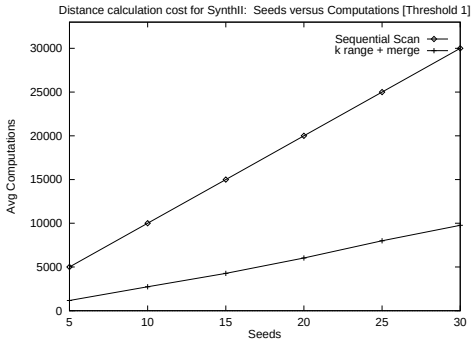


Distance Computations

Figure 12: Time and computational cost for TWO_CIRCLES, varying threshold.



Elapsed Time



Distance Computations

Figure 13: Time and computational cost for TWO_CIRCLES, varying seeds.

databases through multiple examples. Technical report, 1998.

[7] M.V. Boland M.K. Markey and R.F. Murphy. Towards objective selection of representative microscope images. *Biophys. J.*, 1999. in press.

[8] P.M. Murphy and D.W. Aha. UCI Repository of Machine Learning Databases. Department of Information and Computer Science, University of California, Irvine, CA. [http://www.ics.uci.edu/~mlearn/MLRepository.html], 1994.

[9] T.R. Payne. *Dimensionality Reduction and Representation for Nearest Neighbour Learning.* PhD thesis, The University of Aberdeen, Scotland, 1999.

[10] Kriengkrai Prokaew, Sharad Mehrotra, Michael Ortega, and Kaushik Chakrabarti. Similarity search using multiple examples in mars. In *1999 International Conference on Visual Information Systems*, June 1999.

[11] Joseph John Rocchio. Relevance feedback in information retrieval. In Gerard Salton, editor, *The SMART Retrieval System – Experiments in Automatic Document Processing*, pages 313–323. Prentice Hall, Englewood Cliffs, N.J., 1971.

[12] Yong Rui, Thomas S. Huang, and Sharad Mehrotra. Content-based image retrieval with relevance feedback in MARS. In *Proceedings of IEEE International Conference on Image Processing '97*, Santa Barbara, CA, October 1997.

[13] Yong Rui, Thomas S. Huang, and Sharad Mehrotra. Human perception subjectivity and relevance feedback in multimedia information retrieval. In *Proceedings of IS&T and SPIE Storage and Retrieval of Image and Video Databases VI*, San Jose, CA, January 1998.

[14] G. Salton, E.A. Fox, and H. Wu. Extended boolean information retrieval. *CACM*, 26(11):1022–1036, November 1983.

[15] Leejay Wu, Christos Faloutsos, Katia Sycara, and Terry R. Payne. Falcon: Feedback adaptive loop for content-based retrieval. Technical report, Carnegie Mellon University, 2000.

[16] C.T. Zahn. Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Trans. on Computers*, C-20(1):68–86, January 1971.