Issues in Processing Similarity Queries for Multimedia Databases

Giuseppe Amato
IEI-CNR
amato@iei.pi.cnr.it

Fausto Rabitti
CNUCE-CNR
F.Rabitti@cnuce.cnr.it

Pasquale Savino
IEI-CNR
savino@iei.pi.cnr.it

Pavel Zezula
CNUCE-CNR
zezula@iei.pi.cnr.it

Abstract

The extension of data types in multimedia databases has resulted in an important change of the *content query paradigm* which significantly differs from that of traditional database systems and thus requires new approaches to the query evaluation. The most important difference is that a response to a query typically provides not only a set of objects, but also the grades with which these objects qualify. In this article, recent developments in this area are summarized and pointers to relevant original articles are provided. Open problems are also discussed and future research directions proposed.

1 Multimedia Query Processing

Multimedia queries cannot be as logically rigorous as the Boolean queries used in traditional database systems. Objects, such as images or videos, are not atomic units, so the object equality is not a particularly realistic predicate. Instead, searches tend to be based on *similarity*, and *resemblance* or *proximity* is more important than the perfect matching. However, all that is similar is not necessarily correct – this is not an entirely new concept, but its significance is increasing. In other words, the multimedia search paradigm rejects the idea that queries may be expressed in terms of necessary and sufficient conditions able to determine *exactly* which objects one wishes to retrieve. Instead, a query acts more like an *information filter*, defined in terms of specific properties (features, concepts), which reduces the user's search task by providing only a small number of candidates to be examined – it is more important that candidates which are likely to be of interest are not excluded than it is that possibly irrelevant candidates be included.

Depending on application, different types of similarity queries are required. The most frequently used types of similarity queries are:

Range query - *find all objects that are within a specific distance from a query object*;

Nearest neighbors query - *find the first k closest objects to a given query object.*

The process of executing range queries is sometimes designated as the *GradeSearch* and the process of executing the nearest neighbors queries the *TopSearch* - *ProbeSearch* designates the process of assessing scores for a given object. Some systems also require queries, such as "*find all similar sets of objects*", which are usually called the *all-pair* queries or the *similarity joins*. The aim is to find all pairs of objects that satisfy constraints of a similarity selection. The pair either comes from the same set or from two different sets of objects. Alternatively, this operation could find all-triples, etc.

As in the traditional database systems, real multimedia queries can take several properties into account. In order to illustrate, consider a query asking for retrieving images on which we can see the sun. Obviously, the shape and color should play an important role in this query formulation. In particular, the shape should be round, more or less a circle, and the color should be close to red, formally *(Shape = 'round')* $\wedge$ *(Color = 'red')*. Obviously, rather than requiring for the *exact match*, both the conditions in our example represent the *similarity* or *proximity* queries.

What makes such task of query processing particularly difficult is that each condition determines specific grades (or levels) of qualification expressed as numeric *scores* for individual objects – in traditional databases an object either entirely satisfies a condition or it does not qualify at all. Such approach has become quite standard in multimedia databases, but when conjunctions and disjunctions occur in multimedia queries, what grades should be assigned to objects considering the query as a whole?

The primary source of theoretical background in the current effort spend to deal with this problem has been sought in the *fuzzy logic*. According to this theory, the score of $A_1 \wedge A_2$ is the *min* of the scores of A_1 and A_2. Similarly, the score of the disjunction $A_1 \vee A_2$ is the *max* of the scores of A_1 and A_2.

Even though the form of a query language is certainly an important and not trivial issue, we consider the query evaluation processes to be even a more difficult challenge. Due to the shift from the traditional set oriented approach to the area of the *graded sets*, new rules for query evaluation are needed. In this field, a lot of research is expected to find methods for query execution or query optimization, in general. As an example, consider a query asking for the "ten best images containing red round objects". Provided two indexes, one for color and one for shapes, are available, the specific execution plan – able to achieve a sub-linear search time – is certainly not trivial. In particular, it is not clear in advance how many nearest neighbors are needed considering the colors and how many for the shapes so that these ten best objects are obtained respecting the combined criterion.

In the following, we discuss fundamental strategies proposed for query evaluation independently for the similarity range and the nearest neighbor queries.

1.1 Combining range predicates

The problem of executing queries in which multiple range predicates occur, has been studied in depth in [CG96]. In that article, such queries are designated as *filter conditions* with atomic conditions of the form *Grade(attr, value)(o)* $\geq$ *grade* – an object *o* satisfies

this condition if the grade of match between its value *o.attr* for attribute *attr* and a constant value, *value*, is at least *grade*. More complex filter conditions are generated from the atomic conditions using the $\wedge$ and $\vee$ boolean connectives. Obviously, such queries evaluate for specific objects to either *true* or *false*, even though a grade can certainly be associated with each of the retrieved objects.

The use of *GradeSearch* for each atomic condition, not to miss any object that satisfies the entire condition, is claimed to be necessary for a disjunction of atomic conditions in [CG96]. However, we believe that the *ProbeSearch* of the entire file can be more effective in many cases – the *GradeSearch* cost for many attributes and given grades can be quite high.

In case the search condition is a conjunction of atomic conditions, several execution alternatives are possible. As an extreme, the *GradeSearch* can be used for all atomic conditions to obtain *oids* of objects qualifying for individual conditions. Consequently, the intersection of such *oid* sets is performed to obtain the final result. Alternatively, only one condition can be processed through the *GradeSearch* and the rest of the conditions only applied for objects determined by this *GradeSearch*.

In [CG96], the *search-minimal* class of execution alternatives is used, where only a minimal set of conditions is used for *GradeSearch* and the others are evaluated using *Probe*. The cost model for query optimization, assuming that the atomic conditions are independent, consists of the following three components:

- *Selectivity Factor Sel(a):* Fraction of the objects in the repository that satisfy the condition a.
- *Search Cost SC(a):* Cost of retrieving the *ids* of the objects that satisfy the range condition a through *GradeSearch*.
- *Probe Cost PC(a, p):* Cost of checking the condition a for p objects using the *probe* access model.

An optimization algorithm is proposed, which works properly provided the search and probe costs are estimated correctly. However, cost prediction for multimedia indexes seems to be a hard problem.

1.2 Nearest neighbors of complex queries

Recently, the nearest neighbor search has become the primary type of searching in multimedia databases due to the specific nature of attributes (or features) which are used for multimedia content-based retrieval. However, even though specific algorithms for the *TopSearch* has been defined for important storage structures, such as the R-tree [Gu84] and M-tree [CPZ97] (many systems which specialize in the Information Retrieval are also able to provide a specific number of "best matches" to a given query), combination of partial results is certainly a problem.

The pioneering work of applying the fuzzy logic to processing queries containing ranking expressions is reported in [Fag96]. The proposed algorithm uses the *TopSearch* access method to retrieve, or better to say access, objects.

In fact, there are not many problems with processing disjunctions because after retrieving k top objects for all conditions involved in the query, we are sure that the k top objects for the complete query are included – the *max* grade scoring is used for disjunctions.

The execution of conjunction queries is more difficult (or tricky). The main problem is that it is never sure before the execution starts how many (top) objects should be retrieved for individual sources (conditions) so that there are at least k objects in the intersection. That is why the so called *sorted access* is the assumption – with the sorted access facility, it is always possible to ask for another nearest neighbor(s) for each of the conditions.

Assuming independence of conditions, the expected number of objects that should be retrieved for individual conditions has been derived in [Fag96] as

$$k_i = k^{1/m} N^{1-1/m} \tag{1}$$

where k_i is the number of nearest neighbors for the condition i, N is the number of objects in the database, k the number of nearest neighbors for the conjunction, and m the number of conditions in the conjunct. Presumably, such estimate cannot avoid the need for application of the sorted access in practical cases. Notice that due to the assumption about the attribute independence, k_i is the same for all the conditions.

In order to optimize execution of complex nearest neighbor queries, the *TopSearch* strategy for individual conditions has been replaced by the *RangeSearch* strategy in [CG96]. In order to make this idea work, k_i is computed first for the individual atomic conditions using the Equation 1. Then the nearest neighbor query is transformed into a range query: grades g_i, used in the range query are estimated by using selectivity statistics and by requiring that the expected number of objects having grade g_i for atomic condition i, should be at least k_i. The main advantage of this approach is that all the optimization steps proposed in [CG96] for processing range predicates can be applied as well. However, the estimation of the grades is not always easy and even when it is correctly done, the problem of not obtaining sufficiently large intersection still remains due to the unrealistic assumption about the independence of attributes.

1.3 Similarity conditions with weights

In order to outline the problem of similarity conditions with weights, consider our sample query *(Shape = 'round')* $\wedge$ *(Color = 'red')* again and assume that the user's idea of the sun suggests that a proper shape is more important than the color. Intuitively it is clear that, provided different *weights* are attributed to different atomic conditions, the method for computing grades, usually the *min max* scoring, must be modified.

The problem has been studied in depth in [FW97]. Provided both the grades g_i and the weights θ_i for $i = 1, 2, \ldots, m$ (m is the number of atomic conditions) are from the closed interval $[0, 1]$, the weights sum to 1, $\sum_{i=1}^{m} \theta_i = 1$, and $\theta_1 \geq \theta_2 \geq \ldots \geq \theta_m$, the following formula has been proposed to compute scores for conjunctions of m atomic conditions with weights:

$$f_{(\theta_1,\ldots,\theta_m)}(g_1, \ldots, g_m) = \sum_{i=1}^{m-1} [i \cdot (\theta_i - \theta_{i+1}) \cdot min\{g_1, \ldots, g_i\}] + m \cdot \theta_m \cdot min\{g_i, \ldots, g_m\} \tag{2}$$

In order to compute scores for a disjunction of m atomic conditions, similar formula can be used where the scoring function *min* is substituted by the function *max*, in particular:

$$f_{(\theta_1,\ldots,\theta_m)}(g_1,\ldots,g_m) = \sum_{i=1}^{m-1}[i\cdot(\theta_i-\theta_{i+1})\cdot max\{g_1,\ldots,g_i\}]+m\cdot\theta_m\cdot max\{g_i,\ldots,g_m\} \quad (3)$$

2 Extending Access Methods to Support Complex Similarity Queries

Even though several access methods able to support execution of similarity predicates exist, the typical processing time is still considerably high. In this respect, additional research has been published recently proposing strategies to evaluate a complex query at one run of the index and providing analytical tools able to predict an index performance.

2.1 An Index Structure for Complex Queries

An efficient evaluation of complex similarity queries whose predicates refer to a single feature and are defined through a specific language $\mathcal{L}$ is considered in [CPZ98]. Contrary to the language level, which is supposed to deal only with similarity scores, the proposed evaluation process is based on distances between feature values, because known spatial or metric indexes use distances to evaluate predicates. This solution suggests that the index should process a complex query *as a whole*, thus evaluating multiple similarity predicates at a time. The flexibility of such approach is demonstrated by considering three different similarity languages (not just only fuzzy logic). It is also demonstrated how the Metric Tree (M-tree) [CPZ97] can be extended to this purpose. Experimental results clearly show that performance of the extended M-tree version is consistently better compared to the previous approaches.

2.2 Performance Prediction

In order to predict performance of spatial access methods, the concept of *fractal dimension* is used in [FK94]. Using measurements on real data sets, it is demonstrated that real data sets are skewed and behave as mathematical fractals with a measurable non-integer fractal dimension. Such knowledge is then used for predicting the performance of spatial access methods, specifically of the R-tree [Gu84]. Provided formulas can estimate the number of disk accesses for range queries, given only the fractal dimension of the point set and its count.

A similar problem for generic metric spaces has been studied in [CPZ98a]. Contrary to vector spaces where information on data distribution can be exploited, there is no such possibility in generic metric spaces, and a different approach must be taken. Since only distances between pairs of objects can be measured, distance distribution is proposed to be used. In particular, this approach is based on the assumption that the indexed data set comes from a domain which is homogeneous enough (in a probabilistic meaning) to allow cost estimations even if the distance distribution with respect to a specific query object

is unknown. A specific cost model is developed for the M-tree, [CPZ97]. The model has also been successfully validated over both real and synthetic data sets. It is also shown how such model can be used to tune the M-tree in order to minimize a combination of CPU and I/O costs.

3 Future Research Work

Though the problem of processing similarity queries has already attracted several research groups developing multimedia, genetic, and several other emerging data management systems, the topic still needs many issues to be addressed. In general, future research should still continue to improve both the effectiveness and the efficiency.

Standard fuzzy logic with its *max/min* scoring rules has proved to be a possible platform for managing scores of complex predicates, but it is certainly not the only way how to deal with this problem. As [CPZ98] demonstrates, other approaches are possible. But applying different strategies to the same data sets, different ranking of database objects can be obtained. So the problem of a suitable score management concept, or say a language, for a specific application still remains as a big question.

The retrieval process which involves similarity queries does not seem to be a one step process. The query formation process need not be a trivial issue and the query must typically be reformulated several times before its suitable form is found. Consider an image database example where a query can be formulated by either some form of a painting or a sketch by the user or by selection from example images. In this way, sources of color and texture patterns, sketches, layout or structure descriptions, or other graphical information needed for query formation is used.

However, painting a sketch is not always easy and even if the user is able to do it well, it is certainly time consuming. Also, a set of example images available is always restricted, thus none of the images may be suitable for specific query formation. In such situation, techniques such as the *relevance feedback* should be studied and applied to this purpose.

In this respect, users might also be satisfied with *approximate* similarity queries, that is queries that do not necessarily provide objects most similar to a give reference, but guarantee a user defined quality which should be sufficient for formulating a new query. Presumably, evaluation of such queries should be much more efficient, which would reduce the high cost which execution of similarity indexes still require. However, such approach does not decrease the importance of a research aiming at providing new and more efficient storage structures.

References

[CG96] S. Chaudhuri and L. Gravano. Optimizing Queries over Multimedia Repositories. Proceedings of the *ACM SIGMOD 96*, Montreal, Canada, 1996, pp. 91-102.

[CPZ97] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An Efficient Access Method for Similarity Search in Metric Spaces. Proceedings of *VLDB97*, Athens, Greece, August 1997.

[CPZ98] P. Ciaccia, M. Patella, and P. Zezula. Processing Complex Similarity Queries with Distance-based Access Methods. Proceedings of *EDBT97*, Valencia, Spain, March 1998, accepted.

[CPZ98a] P. Ciaccia, M. Patella, and P. Zezula. Similarity Queries in Metric Spaces: A Model for Predicting M-tree Performance. Submitted for publication.

[Fag96] R. Fagin. Combining Fuzzy Information from Multiple Systems. Proceedings of the *ACM PODS 96*, Montreal, Canada, 1996, pp. 216-226.

[FW97] R. Fagin and E.L. Wimmers Incorporating User Preferences in Multimedia Queries. Proceedings of the *International Conference on Database Theory.* Delphi, Greece. *Lecture Notes in Computer Science*, Vol. 1186, Springer, 1997, ISBN 3-540-62222-5, pp. 247-261

[FK94] C. Faloutsos and I. Kamel. Beyond Uniformity and Independence: Analysis of R-trees Using the Concept of Fractal Dimension. Proceedings of the *ACM PODS 94*, Minneapolis, Minnesota, pp. 4-13.

[Gu84] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data*, pages 47–57, Boston, MA, June 1984.