Applications of Quick-Combine for Ranked Query Models

W-T. Balke*, U. Güntzer+, W. Kießling*,

*University of Augsburg, +University of Tübingen, Germany

{balke, kiessling}@informatik.uni-augsburg.de, guentzer@informatik.uni-tuebingen.de

Abstract

In digital libraries queries are often based on the similarity of objects, using several feature attributes like colors, texture or full-text searches. Such multi-feature queries return a ranked result set instead of exact matches. Recently we presented a new algorithm called Quick-Combine [5] for combining multi-feature result lists, guaranteeing the correct retrieval of the k top-ranked results. As benchmarks on practical data promise that we can dramatically improve performance, we want to discuss interesting applications of Quick-Combine in different areas. The applications for the optimization in ranked query models are manifold. Generally speaking we believe that all kinds of federated searches can be supported like e.g. content-based retrieval, knowledge management systems or multi-classifier combination.

1. Query Optimization For Ranked Query Models

Today's handling of multimedia data in information systems such as images, video or audio files poses an increasingly demanding problem. The query evaluation model typically does not retrieve a set of exact matches but rather a ranked result set, where an aggregated score is attached to each object returned. Only a few top-ranked results are normally of interest to the user. A query could for example ask for the top 10 objects from an image collection that are most similar to a fixed image in terms of visual properties; a query type which is often referred to as 'query by visual example'.

Query optimization needs to be adapted to this essentially different query model for multimedia data. Some systems have already been implemented, e.g. visual retrieval systems like IBM's QBIC [1] or Virage's VIR [2]. Database applications and middlewares like GARLIC [3] or the HERON project [4] have already started to use the capabilities of visual retrieval. A major challenge in all of these systems is that similarity between different objects cannot be defined precisely. To handle queries on similarity different kinds of information on the multimedia objects have to be stored. For example in the case of images this could be color histograms, shapes of occurring objects, features on textures and layout or related fulltext information describing the object. Queries on similarity do not have to focus on one single feature. In general multimedia queries will refer to at least some different features simultaneously. According to a potentially weighted combining function for each database object an *aggregated score value* is computed. The results are then sorted according to their scores and are returned with a *rank number* - the top-ranked object has the best score value of the entire database and so on. A query focusing on a single feature is called *atomic*. Complex multimedia queries are combinations of atomic subqueries.

2. The Quick-Combine Approach

The optimization challenge is to combine several ranked results of atomic queries in order to determine the k overall best objects from the database. A naive approach would calculate the aggregated score for all database objects according to a given combining function. With growing database sizes, this obviously results in unacceptable response times, requiring a linear scan of the entire database. But as only the k top objects have to be returned, not all database objects have to be accessed.

In general atomic queries can be posed in two ways. The first type is searching the database using a suitable index and retrieving the objects in the database ordered by descending score for a single feature, which we refer to as *enlarging an output stream* or *sorted access*. On the other hand a specific object's score in each atomic output stream could be of interest. This case is referred to as *random access*. In 1996 Fagin presented an approach [8] to process a complex query consisting of several atomic subqueries that may use any monotonous combining function, as for example the maximum or arithmetical mean. Fagin's work has also strongly influenced statistical approaches as [9]. There a different query model is presented using not only ranking expressions for query processing, but a filter condition like the WHERE-clause in SQL. During processing first the filter condition is evaluated and later on only objects satisfying the condition are ranked according to a ranking expression.

Using the Quick-Combine algorithm [5], the overall number of necessary accesses can be *minimized* with an adequate control flow and an efficient test of termination. For these tasks Quick-Combine does not only use the information of ranks in output streams, but also the scores which are assigned to all objects in each output stream, the specific form of the combining function and heuristics to force early termination. Quick-Combine is applicable for every monotonic combining function, even in the case of maximum and minimum, and is capable of processing weighted queries. Quick-Combine can even return the overall best database objects *successively*. Thus the top-scored object of the whole collection can be delivered to the user as soon as it is found, which of course also applies to all following ranks up to k, when the algorithm finally terminates. If the user is already satisfied by the first few ranks, the query execution for large values of k can be stopped during processing.

In combining different ranked result sets it is not only a problem how far the output streams have to be expanded, but also the sequence in which they are expanded and aggregated scores of objects are calculated. Since a correct result set is always guaranteed, the final result set will of course not differ, if the sequence is changed, but different sequences may strongly influence the efficient execution of the algorithm. To find an optimal order the Quick-Combine algorithm uses a set of *indicators* that rely on heuristic considerations and help to choose always the stream whose further expansion will have the greatest effect on the final result set. But the indicators are not only useful for equally weighted queries. Streams with low weights should naturally be regarded less important than highly weighted streams which should get prior evaluation. We will give an example of the composition and usefulness of these indicators in section 3.3 where the problem of multi-classifier combination is described.

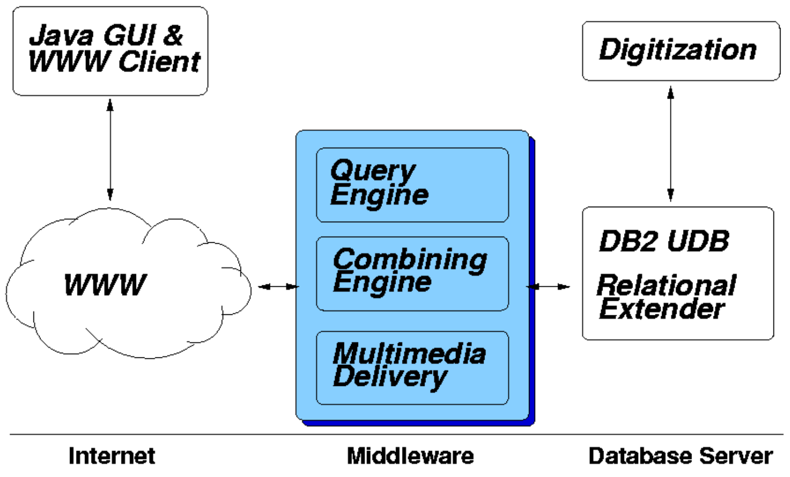

Fig. 1: Architecture of the HERON-system

The HERON system [4] has been used to test Quick-Combine on a collection of heraldic images. Besides components for query composition, user specific image delivery and efficient format conversions, the HERON-middleware features a combining engine that can use different visual retrieval systems and databases (cf. Fig. 1). For practical numbers of objects to retrieve Quick-Combine even accesses *30 times less objects* than previous approaches guaranteeing a correct result set. From our practical tests we gained insight into distributions that really occur in image retrieval. We argue that typical distributions from visual retrieval systems consist of a low percentage of objects having high and medium score values and a high percentage having very low scores. If text retrieval is included the percentage having high and medium scores even decreases. Quick-Combine was therefore tested on synthetic data, where different score distributions in the output streams have been analyzed. For these distributions similar benchmark results have been proven [5]. In these tests Quick-

Combine showed that even skewed data distributions can be handled efficiently and the algorithm scales with both, an increasing number of streams to combine and a growing database size.

3. Applications of Quick-Combine

The HERON-project investigates the capabilities of content-based retrieval in digital libraries. However, Quick-Combine seems not only useful in the field of content-based retrieval for multimedia data, but further typical areas of applications can be:

- Feature lists within the SQL/MM standard
- Content management in digital libraries
- Enterprise information portals
- Knowledge management systems
- Multi-classifier combination

3.1. The SQL/MM Standard

Together with the development of visual retrieval engines and multimedia databases an extension of the database query language SQL towards multimedia applications has been proposed. The SQL/MM standard [6] was defined to enable the handling of multimedia data in object-relational databases systems. Part 5 deals among others with content-based retrieval for still images, specifying image datatypes and methods for manipulation. This new standard will strongly influence the capabilities of SQL-engines in future multimedia databases. The standard provides an adequate datatype SI_StillImage to handle image data. This datatype provides a container for basic data and consists of several attributes that do not represent all information about the image.

The retrieval methods focus on four different content-based methods. For image searches low-level features have to be used. The standard only implements average color, histogram color, positional color and texture features. But it also provides a composed feature called *SI_FeatureList*. It allows retrieval by any weighted combination of the basic features. The specific calculation of the combined result sets and the types of supported combining functions, however, is left to the implementation. Thus already four different output streams may be necessary for content-based retrieval, which of course have to be combined using an algorithm like Quick-Combine. Besides these feature lists complex SQL/MM queries almost always will involve several text-based output streams, too. Thus in future multimedia SQL queries the combination of between 5 and 10 different features can be assumed. To perform such multimedia queries efficiently high speed iterators for sorted accesses relying on multi-dimensional k-nearest-neighbor searches, e.g. [10], and effective multi-dimensional index structures for random accesses are needed in the DBMS-kernel.

3.2. The Portal Challenge

In recent years the amount of digital information has dramatically increased leading to a strong need of efficient ways to manage this knowledge. The first steps towards knowledge management were digital libraries, where a large amount of information has been made accessible efficiently for the first time. Also facilities as e.g. enterprise information portals have been created, especially for managing personalized user profiles or handling semi-structured data. The main idea of a portal is to provide a single, web-based interface to disconnected and incompatible data spread across a variety of separate applications and repositories. These portals thus provide one (possibly personalized) logical view and common query capabilities across different platforms and enterprise-wide information sources. Such information sources in general are not limited to relational databases, image archives or document repositories, but more sophisticated portals also allow external web data to be directly integrated into query result sets. In general the ideal goal is "just in time" information, retrieved and assembled as needed, freely accessible and exchangeable across diverse systems, and filtered to be both manageable and usable.

In general portals can be divided in two major groups. First-generation portals were those grown out of web search engines which can be considered as general purpose gateways onto the Web, like Yahoo!, Excite or Lycos. This group is often referred to as horizontal portals. A second quickly emerging group have become known as vertical portals, i.e. portals with a tightly focused content area for a specific audience. However, the area of application for portals is not restricted to the Internet but also begins to integrate a variety of datasources from local filesystems or databases via Intranets. Especially in the field of business intelligence the workflow can be strongly improved by providing vertical portals accessing not only internal business data over various departments, but also integrating relevant information from the Internet.

A major drawback in recent knowledge management approaches has been the cost-intensive overhead of administration and management of data. As all the information in portals or information brokering systems is collected from welldefined registered information sources the administrative overhead can be reduced considerably. For instance logging or workspace and resource management are typical tasks that help users to organize their information flow. But the needs have grown beyond merely administrative support, especially personalization has become an absolute necessity, because the amount of information provided via Inter-/Intranet has already overstrained the capacity of the user to process it. The use of user profiles allows individually tailored information feeds and displays by means of a customized combining function taking into account the information sources the user is interested in and his personal weightings for each source. Thus queries can be posed more intuitively, information can be adequately filtered and the results presented in a way suitable for the specific user and easy to understand and giving the users the possibility to concentrate on the items most relevant to them.

In all of these applications however, documents have to be retrieved from a variety of data sources according to the users' demands. So-called federated searches can be used to get all relevant and up-to-date information as a basis for decision support, benchmarking and data mining, workflow management or even complex tasks as interdisciplinary research. For each purpose the adequate search capabilities have to be provided. Main types of federated searches include *traditional full-text searches*, *fuzzy and phonetic search capabilities* including authority lists or thesauri, and *attribute-based queries*, e.g. documents of certain length or pictures in specific image formats. Obviously also in this case several ranked result sets are delivered providing ideal conditions for the application of Quick-Combine. Another application are e.g. every kind of meta-search engines which combine the ranked result sets from underlying retrieval systems. Since performance issues prohibit the materialization of all query results in the underlying systems, today's federated searches can often use only combining functions like the maximum taking only a few top-scored objects into account. With Quick-Combine it is sufficient to evaluate complex combining functions for the most relevant objects only instead of evaluating them for all objects. Nevertheless a correct result set can be guaranteed. These complex combining functions can exactly model the users' intention and thus considerably improve the quality of the retrieval result.

3.3. Multi-Classifier Combination

In the area of multi-classifier combination, the evaluation of similarity between objects is based on a variety of classifiers, each of which votes for a certain ranking of the result set. For example speech recognition will provide a by far higher recognition rate, if not only phonetic features are involved, but also e.g. visual features like lip-reading or features based on the semantic context of spoken sentences [7]. In this application spoken words can be assigned to certain word classes (given by an underlying grammar) with a specific certainty. As the ranking of the classifiers may differ, the overall highest certainties have to be calculated. Thus also in this case the combination of ranked result sets is necessary. Even in small grammars the resulting number of word classes has proven to be very high, thus also in this case the efficient combination without materialization of all the classes and certainties promises a considerable speed-up for e.g. modern speech recognition systems.

Considering the problem of natural speech recognition the main approach is dividing words or sentences into classes. We will take single words as an example, but analogous considerations can be made for both phonemes or sentences and sequences of words. Starting with the acoustical or visual input different kinds of classifiers will recognize specific words from each set of phonemes with different certainties using e.g. Hidden Markov Models. These certainties can be improved by using meta-information like grammars, dictionaries or collections of possible phrases (e.g. possible combinations of three word phrases, etc.) that are often used in a postprocessing step as feedback to choose the right recognition results. In the recognition step each classifier may rank the recognized words according to the certainty of recognition assigned as score value. At the beginning of each list there will be words appearing or sounding most similar to the input and as the score values decrease also the similarity in appearance or sound will decrease with respect to the classifier. In practice due to noise and the properties of the classifier not all of these output lists will show words in the same order. However, the chance that one of the overall top-scored words will be correct, i.e. exactly matches the spoken word, is high.

The distribution in applications like this can generally be expected to be very skewed. There will always be a small set of similar words, but words will get less similar quite soon and the majority of words in a language will not be similar to the recognized word at all, i.e. get assigned a rather small score value. Quick-Combine features *indicators* [5] that consist of the derivative of functions correlating score values to the ranks on which they occur for each output stream and the partial derivative of the combining function for each stream. These indicators are calculated for each stream and the highest indicator determines the stream that is to be expanded next by sorted access. Due to this indicator the Quick-Combine algorithm will always preferably enlarge and analyze those streams showing the most rapid decline of score values. These are the streams where the discrimination between words can be assumed to be quite high. As the algorithm is capable of working with skewed distributions a considerable speedup compared to traditional approaches has been achieved. Another advantage is that almost any number of different classifiers can be used for recognition.

The recognition rate can even be increased, if e.g. a dictionary or grammar is used. The dictionary can be seen as one more *output stream that consists of exact matches only*. That means any word that is contained in the language or dictionary is assigned a score value of 1 and every other word will get a score value of 0. Depending on the area of application the dictionary stream should be assigned a higher or lower relevance in finding an overall best object. This can for instance be achieved by assigning high weights within the combining function. Obviously, as there is no specific order between the words in the dictionary stream (all have a score value of 1), it will not be sensible to enlarge this stream by sorted access. However, since the indicator of Quick-Combine takes score distributions into account, any stream of that kind will be excluded from sorted accesses immediately. Quick-Combine is thus also capable of combining ranked result sets and exact match results in a very intuitive way. The lookup of for instance a word in a dictionary does not have to be a separate step in processing the input, but can be implemented by just adding another output stream containing the respective information about the language. Quick-Combine will automatically adapt its retrieval strategy to the nature of those streams containing only exact match information.

The problem of *combining exact matches and ranked result sets* is quite a difficult matter in modern database technology. Though today's database systems like IBM's DB2 are able to perform a combination within the WHERE-clause of SQL-statements, the semantic of such a combination is to filter the ranked result only taking those objects into account for the final result set that satisfy the exact match condition. However, there are applications where the exact match result set might be highly relevant though an absolute filter effect on the retrieval result might not be wanted. Consider for instance our dictionary example from above. If all the classifiers have recognized a certain word or sequence of phonemes, but it does not occur in the dictionary, the phrase may nevertheless not be wrong. It might be an neologism or very unusual phrase. In this case it would be unwise to simply ignore the unison result of all the classifiers. In the Quick-Combine approach the weighting of the dictionary stream can be individually adapted to the needs of the field of application. Unlike traditional database systems the Quick-Combine algorithm may integrate streams in a sophisticated way ranging from hard filter constraints to low-weighted proposals.

4. Summary and Outlook

In this paper we investigated the application of combining algorithms for federated searches. For this task the Quick-Combine algorithm that has been developed within the HERON-project, has proven to combine ranked output streams with high performance. We have presented several areas of applications in which the combination of output streams is vital. In future database technology combination algorithms will have to be implemented in several parts like the SQL engine or front ends relying on portal technology. In addition, the combination of output streams is also important in other applications. As an example we have pointed out multi-classifier combination for natural speech recognition.

The Quick-Combine algorithm was especially designed for use over data sources where common identifier for all objects are given. However, in today's applications also highly heterogeneous environments may play an important role [11]. For these environments score values in different output streams cannot simply be mapped onto a single object, but expensive comparisons have to be made. Random accesses are thus much more expensive than sorted accesses and special algorithms for these environments are needed. As we have shown in section 3.2 a useful application for this kind of combination are e.g. meta-search engines in the area of semi-structured data. An efficient example of this kind of algorithm, called the Stream-Combine algorithm, is published in [12].

Since the Quick-Combine algorithm was developed as a general middleware approach, the applications for efficient combination of multiple features or output streams may spread over a variety of research areas. For instance we have shown the application in natural speech processing and the new way of integrating dictionaries into the algorithm by simply adding another output stream. We would thus appreciate other research communities to discuss the approach and point out promising applications. We would also welcome suggestions of cooperation for a prototypical implementation of Quick-Combine with respect to the specific research area.

Bibliography

[1] Faloutsos, Barber, Flickner, Hafner, Niblack, Petkovic, Equitz. Efficient and Effective Querying by Image Content, Journal of Intelligent Information Systems, Vol. 3 (1994), pp. 231-262

[2] Bach, Fuller, Gupta, Hampapur, Horowitz, Humphrey, Jain, Shu. Virage Image Search Engine: An Open Framework for Image Management. In: Storage and Retrieval for Image and Video Databases (SPIE) 1996, pp. 76-87

[3] Cody, Haas, Niblack, Arya, Carey, Fagin, Flickner, Lee, Petkovic, Schwarz, Thomas, Tork Roth, Williams, Wimmers, Querying Multimedia Data from Multiple Repositories by Content: The Garlic Project. In: Proc. of the Working Conference on Visual Database Systems (VDB-3), Lausanne, Switzerland. 1995

[4] Kießling, Erber-Urch, Balke, Birke, Wagner. The HERON Project – Multimedia Database Support for History and Human Sciences. In: 28. Annual Conference of the German Computer Society (GI): INFORMATIK98, Magdeburg, Germany. 1998, pp. 309-318

[5] Güntzer, Balke, Kießling. Optimizing Multi-Feature Queries for Image Databases. In: Proc. of the Intern. Conf. on Very Large Databases VLDB 2000, Cairo, Egypt. 2000, pp. 419-428

[6] ISO/IEC FCD 13249-5:1999 SQL/MM SAF-005. Information Technology - Database Languages - SQL Multimedia and Application Packages - Part 5: Still Image. 1999

[7] Yu, Jiang, Bunke. Combining Acoustic and Visual Classifiers for the Recognition of Spoken Sentences. In: Proc. of the Intern. Conf. on Pattern Recognition ICPR 2000, Barcelona, Spain. 2000, pp. 491-494

[8] Fagin. Combining fuzzy information from multiple systems. In: Proc. of the 15th ACM Symposium on Priciples of Database Systems, Montreal. Canada. 1996, pp. 216-226

[9] Chaudhuri, Gravano. Optimizing Queries over Multimedia Repositories, In: Proc. of the 16th ACM Symposium on Priciples of Database Systems, Tucson, USA. 1997, pp. 91-102

[10] Berchtold, Keim, Kriegel, Seidl. Indexing the Solution Space: A New Technique for Nearest Neighbor Search in High-Dimensional Space. In: IEEE Transactions on Knowledge and Data Engineering (TKDE), Vol. 12, No. 1, 2000, pp. 45-57

[11] Cohen. Integration of Heterogeneous Databases Without Common Domains Using Queries Based on Textual Similarity. In: Proc. of the ACM SIGMOD Intern. Conf. on Management of Data, Seattle, USA. 1998, pp. 201-212

[12] Güntzer, Balke, Kießling. Towards Efficient Multi-Feature Queries in Heterogeneous Environments. In: Proceedings of the IEEE International Conference on Information Technology: Coding and Computing (ITCC 2001), Las Vegas, USA, 2001