Is Similarity Search useful for High Dimensional Spaces?

Roger Weber
Institute of Information Systems
ETH Zentrum, 8092 Zurich
weber@inf.ethz.ch

Pavel Zezula
Technical University
Údolni 19, Brno, Czech Republic
zezula@cis.vutbr.cz

Extended Abstract

In recent years, multimedia content-based retrieval has become an important research problem. In order to provide effective and also efficient access to relevant data stored in large (often distributed) digital repositories, advanced software tools are necessary.

Content-based retrieval works on the idea of abstracting the contents of an object, for example color or shape in the case of images, by so-called features – features are typically points in a high-dimensional vector space. Instead of determining the similarity of two objects based on their raw data, only the much smaller feature representations are used to estimate the objects' similarity. Given a reference (query) object represented by its features, similarity predicates are defined to retrieve a specific number of best cases or all objects satisfying a (distance) constraint. In this respect, we can distinguish between similarity range and nearest neighbor (NN) queries.

Usefulness of NN-search. The dimensionality of a single feature type may range from small (4-9) to large (several hundreds). Frequently, it is not sufficient to query one feature in isolation. Rather, a query typically combines several feature types to better reflect the notion of similarity. Given this, the number of dimensions of combined features is well above 10. Beyer et al. [2] have recently questioned NN-search in high-dimensional vector spaces. The result of their theoretical analysis over a uniformly distributed data set is that NN-search as an implementation of similarity search is questionable if the number of dimensions is large, e.g. above 16. On the other hand, experiments with feature data of a large image database have shown that their finding must not always hold when applied to real data. It is still an open question how to find out whether the retrieved result of a query is of satisfactory quality for the user.

Efficiency of NN-search. The feature approach to similarity search typically reduces the amount of searched data by orders of magnitude, e.g. from the tera-byte range to the giga-byte range. Since this reduction of data volume is not enough for large collections, researchers have proposed a number of methods which are mostly based on data-space partitioning. Index trees such as R-tree [5], X-tree [1], SR-tree [6], M-tree [3] or TV-tree [7] divide the data-space according to the distribution of data objects inserted or loaded into the tree. The main objective is to prune the search space in such a way that the NN can be found in logarithmic time. Although these access methods generally work well for low-dimensional spaces, their performance is known to degrade as the number of dimensions increases—a phenomenon which has been termed the *dimensional curse*. This phenomenon has been observed in a number of experiments with synthetic as well as real data [6, 8, 2]. Recently, it has been formally shown that the complexity of similarity search in trees tends towards $O(N)$ as dimensionality increases (N denotes the number of data items) [8]. As experiments and formal analysis confirm, a sequential scan outperforms tree-based approaches if the dimensionality is not small, e.g. larger than 10.

In order to obtain acceptable response times for similarity searches, we can see the following three main research directions which are worth investigation: 1) improving the performance by reducing the amount of data read during the search, 2) parallelization of search by using a network of workstations (NOW), 3) approximated similarity search, i.e. instead of the exact result, we provide a sufficiently good result.

Reducing the searched data. The recently developed *vector approximation file*, the so-called VA-File, compresses the vector data to reduce IO-costs during similarity searches [8]. The VA-File consists of two separated files: the vector file, containing the feature data, and the approximation file, containing rough, geometrical approximations of the vectors. Typically, the approximation file is 6 times smaller than the vector file. The search algorithm of the VA-

File first reads only the approximation file and filters out data items that cannot fulfill the similarity predicate. Then, it applies a refinement step to eliminate false hits: for each remaining item, the vector data is accessed and the similarity predicate is evaluated. The filtering step of the VA-File is so efficient that apart of the resulting items, only very few additional items are visited. Due to the reduced amount of data read, the VA-File performs up to a factor of 4 faster than a sequential scan through the feature data. Moreover, due to the compressed representation of the vectors, it is more likely that the entire approximation file fits into main memory than the entire feature data (or a tree-based index). Thus, NN-search with the VA-File is often only CPU-bound as opposed to other methods.

The *Generic multimedia object indexing*, GEMINI, approach was proposed by Faloutsos [4]. In principle, a user defined distance measure in high-dimensional space is replaced by a new distance measure in space with much lower dimensionality. Provided the new distance measure satisfies the so called *lower-bounding* property, the search can be performed in the low-dimensional space, that is the environment for which efficient implementations are known. Though such search can provide some *false drop*, no *false dismissals* are possible.

Parallelization of similarity search. Employing a network of workstation to solve data or computational intensive problems is a commonplace approach in computer science. The main advantage of this approach is the high value for the ratio of performance over cost. Both the VA-File and the sequential scan through the vector data bear considerable parallelization potential due to the small fraction of serial work, i.e. work that cannot be parallelized. In contrast, parallelization of a tree-based structure requires a high coordination effort and, thus, the serial fraction of the problem is much higher. Parallelization is especially beneficial for similarity searches in large collections, i.e. if the similarity search is IO-bound on a single component. In this case, we can easily eliminate the IO-bottleneck with a NOW by combining the memory resources of all components. This gives rise to a speedup that is much better than linear.

Approximated similarity search. Recently, the vagueness in query specification together with the high query execution costs led to investigating the following idea. Provided an approximated similarity search can be performed much faster than the precise search, the approximate similarity retrieval can play a useful role in the global process of iterative similarity retrieval. In other words, since processing of exact similarity queries is not always required, approximate answers would suffice, especially when obtained at much lower costs.

In general, there are two major (complementary) directions in which the performance of the similarity search can be improved - both of them use an advantage of relaxing the exact similarity search by a sort of approximation. These directions are, respectively, based on (1) a modification of indexed features and (2) a modification of search algorithms. A large number of approximations have already been proposed, but much more is still expected in the near future. However, approximation is certainly a tradeoff between the effectiveness (i.e. the quality) and the efficiency (i.e. the performance). At this place, the problem of the similarity search benchmarking is very important.

References

[1] S. Berchtold, D. Keim, and H.-P. Kriegel. The X-tree: An index structure for high-dimensional data. In *Proceedings of the International Conference on Very Large Databases (VLDB)*, pages 28–39, 1996.

[2] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbour" meaningful? In C. Beeri and P. Buneman, editors, *Proc. 7th Int. Conf. Data Theory, ICDT*, number 1540 in Lecture Notes in Computer Science, LNCS, pages 217–235. Springer-Verlag, 10–12 Jan. 1999.

[3] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *Proceedings of the International Conference on Very Large Databases (VLDB)*, Greece, 1997.

[4] C. Faloutsos. *Searching Multimedia Databases by Content*. Kluwer Academic Publishers, 1996.

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

[6] N. Katayama and S. Satoh. The SR-tree: An index structure for high-dimensional nearest neighbor queries. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 369–380, Tucson, Arizon USA, 1997.

[7] K.-I. Lin, H. Jagadish, and C. Faloutsos. The TV-tree: An index structure for high-dimensional data. *The VLDB Journal: The International Journal on Very Large Data Bases*, 3(4):517–549, Oct. 1994.

[8] R. Weber, H.-J. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proceedings of the International Conference on Very Large Databases (VLDB)*, volume 24, New York, USA, August 1998.