In Proc. 15th ACM Conf. on Information and Knowledge Management (CIKM'06), Arlington, U.S.A., 2006

Approximate Reverse k-Nearest Neighbor Queries in General Metric Spaces

Elke Achtert, Christian Böhm, Peer Kröger, Peter Kunath, Alexey Pryakhin, Matthias Renz

Institute for Computer Science, Ludwig-Maximilians Universität München
Oettingenstr. 67, D-80538 Munich, Germany
{achtert,boehm,kroegerp,kunath,pryakhin,renz}@dbs.ifi.lmu.de

ABSTRACT

In this paper, we propose an approach for efficient approximative RkNN search in arbitrary metric spaces where the value of k is specified at query time. Our method uses an approximation of the nearest-neighbor-distances in order to prune the search space. In several experiments, our solution scales significantly better than existing non-approximative approaches while producing an approximation of the true query result with a high recall.

Categories and Subject Descriptors: H.2.2 [Physical Design]: Access methods

General Terms: Algorithms, Performance

Keywords: Approximative similarity search, reverse nearest neighbor

1. INTRODUCTION

A reverse k-nearest neighbor (RkNN) query returns the data objects that have the query object in the set of their k-nearest neighbors. A naive solution of the RkNN problem requires $O(n^2)$ time, as the k-nearest neighbors of all of the n objects in the data set have to be found. In general, the RkNN problem appears in many practical situations such as geographic information systems (GIS), traffic networks, adventure games, or molecular biology where the database objects are general metric objects rather than Euclidean vectors. In most applications, the parameter k can change from query to query and is not known beforehand. In addition, the efficiency of the query execution is much more important than effectiveness, i.e. users want a fast response even if the results are only approximate (as far as the number of false drops and false hits is not too high). Existing approximative approaches for RkNN search [2, 3] are only designed for Euclidean vector data. All other approaches for the RkNN search are exact methods that usually produce considerably higher runtimes.

In this paper, we propose an efficient approximate solution for the RkNN problem based on the observation that if the distance of an object p to the query q is smaller than the 1-nearest neighbor distance of p, p can be added to the result set. Our solution is extends the work in [1] and is designed for general metric objects and allows RkNN queries for arbitrary (unbounded) values of k. The idea is to use a suitable approximation of the kNN distances for each k of every object in order to evaluate database objects as true hits or true drops without requiring a separate kNN search. The proposed concepts can be integrated into any hierarchically organized, tree-like index structure for metric spaces. In addition, it can also be used for Euclidean data by using a hierarchically organized, tree-like index structure for Euclidean data. In summary, our solution is the first approach that can answer RkNN queries for any $k \in \mathbb{N}$ in general metric databases. Since our solution provides superior performance but approximate results, it is applicable whenever efficiency is more important than complete results. However, we will see in the experimental evaluation that the loss of accuracy is negligible.

CIKM'06, November 5–11, 2006, Arlington, Virginia, USA.
ACM 1-59593-433-2/06/0011.

2. APPROXIMATE RKNN SEARCH

The only existing approach to RkNN search that can handle arbitrary values of k at query time and can be used for any metric objects is the MRkNNCoP-Tree [1]. This approach, however, is optimized for exact RkNN search and its flexibility regarding the parameter k is limited by an additional parameter k_{max}. This additional parameter must be specified in advance, and is an upper bound for the value of k at query time. If a query is launched specifying a $k > k_{max}$, the MRkNNCoP-Tree cannot guarantee complete results. In our scenario of answering approximate RkNN queries, this would be no problem but since the MRkNNCoP-Tree constraints itself to compute exact results for any query with $k \leq k_{max}$, it generates unnecessary computational overhead.

Our idea is to store one approximation of the kNN distances for any $k \in \mathbb{N}$. This approximation is represented by a function, i.e. the approximated kNN distance for any value $k \in \mathbb{N}$ can be calculated by applying this function. Similar to existing approaches, we can use an extended tree-like metric index, that aggregates for each node the one approximation of the approximations of all child nodes or data objects contained in that node. These approximations are again represented as functions. At runtime, we can estimate the kNN distance for each node using this approximation in order to prune nodes analogously to the way we can prune objects.

A suitable model function for the approximation of our kNN distances for every $k \in \mathbb{N}$ should obviously be as compact as possible in order to avoid a high storage overhead

and, thus, a high index directory. In our case, we can assume that the distances of the neighbors of an object o are given as a (finite) sequence

$$NNdist(o) = \langle nndist_1(o), nndist_2(o), \ldots, nndist_{k_{max}}(o) \rangle$$

for any $k_{max} \in \mathbb{N}$. Our task here is to describe the discrete sequence of values by some function $f_o : \mathbb{N} \to \mathbb{R}$ with $f_o(k) \approx nndist_k(o)$. As discussed above, such a function should allow us to calculate an approximation of the kNN distance for any k, even for $k > k_{max}$ by estimating the corresponding values.

Following the theory of self-similarity it can be assumed that the kNN distances also follow the power law, i.e. $k \propto nndist_k(o)^{d_f}$, where d_f is the fractal dimension. Transferred into log-log space, we have a linear relationship:

$$\log(nndist_k(o)) \propto \frac{1}{d_f} \cdot \log(k).$$

From this observation, it follows that it is generally sensible to use a model function which is linear (and thus compact) in log-log space, corresponding to a parabola in non-logarithmic space. In the following, we consider the pairs $(\log(k), \log(nndist_k(o))$ as points of a two-dimensional vector space (x_k, y_k). Like in most other applications of the theory of self-similarity, we need to determine a classical regression line that approximates the true values of $nndist_k(o)$ with least squared errors. This line is exactly the approximation of the kNN distances we want to aggregate. In other words, for each object $o \in \mathcal{D}$, we want to calculate the function $f_o(x) = m_o \cdot x + t_o$ that describes the regression line of the point set $\{(\log k, \log nndist_k(o)) \mid 1 \leq k \leq k_{max}\}$.

From the theory of linear regression, the parameters m_o and t_o can be determined as

$$m_o = \frac{(\sum_{k=1}^{k_{max}} y_k \cdot \log k) - k_{max} \cdot \bar{y} \cdot \frac{1}{k_{max}} \sum_{k=1}^{k_{max}} \log k}{(\sum_{k=1}^{k_{max}} (\log k)^2) - k_{max} \cdot (\frac{1}{k_{max}} \sum_{k=1}^{k_{max}} \log k)^2}$$

where $\bar{y} = \frac{1}{k_{max}} \sum_{k=1}^{k_{max}} \log nndist_k(o)$, and

$$t_o = \bar{y} - m_o \cdot \frac{1}{k_{max}} \sum_{k=1}^{k_{max}} \log k.$$

Using these concepts, an accurate approximation for each object of the database can be generated. When using a hierarchically organized index structure, the approximation can also be used for the nodes of the index to prune irrelevant sub-trees. Usually, each node N of the index is associated with a page region representing a set of objects in the subtree which has N as root. In order to prune the subtree of node N, we need to approximate the kNN distances of all objects in this subtree, i.e. page region. If the distance between the query object q and the page region of N, called MINDIST, is larger than this approximation, we can prune N and thus, all objects in the subtree of N. The MINDIST is a lower bound for the distance of q to any of the objects in N. The aggregated approximation should again estimate the kNN distances of all objects in the subtree representing N with least squared error. This can be done in a straight-forward manner.

The concepts presented here can be integrated into any hierarchically organized index for metric objects or into Euclidean index structures. Then, the algorithm for approximate RkNN queries is similar to the exact RkNN query algorithms of the MRkNNCoP-Tree. However, using our concepts, the index can be used to answer RkNN queries for any k specified at query time. Let us point out that the value of k is not bound by a predefined k_{max} parameter, although the approximation of the kNN distances are computed by using only the first k_{max} values, i.e. the kNN distances with $1 \leq k \leq k_{max}$. The kNN distance for any $k > k_{max}$ can be extrapolated by our approximations in the same way as for any $k \leq k_{max}$.

A query q is processed by traversing the index from the root of the index to the leaf level. A node N needs to be refined if the distance between q and N is smaller than the aggregated kNN distance approximation of N. Those nodes having distance to q larger than their aggregated kNN distance approximation are pruned. The traversal ends up at the data node level. Then, all points p inside the obtained nodes are tested using their approximation $f_p(x) = m_p \cdot x + t_p$. A point p is a hit if $\log(dist(p,q)) \leq m_p \cdot \log k + t_p$. Otherwise, if $\log(dist(p,q)) > m_p \cdot \log k + t_p$, point p is a miss and should be discarded.

In contrast to other approaches that are designed for RkNN search for any k, our algorithm directly determines the results. In particular, we do not need to apply an expensive refinement step to a set of candidates. This further avoids a significant amount of execution time.

3. RESULTS

We integrated our concepts into an M-Tree and compared our concepts with the methods proposed in [1] using two real-world metric datasets. On both datasets, our approach clearly outperforms the competing MRkNNCoP-Tree. The performance gain of our approach over the existing method also grows with increasing database size.

We also executed RkNN queries on the metric databases with varying k and compared the scalability of both competing methods. The parameter k_{max} was set to 100 for both approaches in all experiments. With increasing k, the performance gain of our method over the competitor rapidly grows.

In summary, in almost all parameter settings, our novel solution turned out to be clearly faster than the MRkNNCoP-Tree which computes an exact solution but is limited by the k_{max} parameter. On the other hand, our solution is not limited to any $k \leq k_{max}$ but only designed for approximate answers. However, in all our experiments, we achieved high recall values of clearly above 90%. Furthermore, the recall does not decrease significantly when answering RkNN queries with $k > k_{max}$. This is important, since it indicates that our solution is very efficient and produces high quality results.

4. REFERENCES

[1] E. Achtert, C. Böhm, P. Kröger, P. Kunath, A. Pryakhin, and M. Renz. Efficient reverse k-nearest neighbor search in arbitrary metric spaces. *Proc. SIGMOD*, 2006.

[2] A. Singh, H. Ferhatosmanoglu, and A. S. Tosun. High dimensional reverse nearest neighbor queries. *Proc. CIKM*, 2003.

[3] C. Xia, W. Hsu, and M. L. Lee. Erknn: efficient reverse k-nearest neighbors retrieval with local knn-distance estimation. *Proc. CIKM*, 2005.