Indian J.Sci.Res. 17(2): 603-607, 2018 ISSN: 0976-2876 (Print)
ISSN: 2250-0138(Online)

AUTHENTICATED MULTI-STEP NEAREST NEIGHBOR SEARCH IN DISTRIBUTED SERVERS

[1]G.Sridevi, [2]C Manipushpa, [3]G Vikranthi
[1,2,3] APGC, Ramanthapur, Telangana.

Abstract:-The multi-step NN framework is motivated by applications that entail expensive distance computations. Specifically, let DST(Q, P) be the actual distance between a query Q and a data point P□DB. The applicability of the multi-step framework rests on the existence of a filter distance metric dst, which is cheap to evaluate and satisfies the lower bounding property, i.e., for every possible Q and P: dst(Q, P) Š DST(Q, P).Assuming that DB is indexed by an R*-tree, themulti-step kNN algorithm first retrieves the k Euclidean NNs of Q using an incremental algorithm. These points are inserted into a result set RS, and their network (DST) distances are computed. Let DSTmax be the network distance1 between Q and its current kth NN Pk. The next Euclidean NN P is then retrieved. As long as dst(Q, P) <DSTmax, the algorithm computes DST(Q, P) and compares it against DSTmax. If DST(Q, P) <DSTmax, P is inserted into RS, the previous Pk is expunged, and DSTmax is updated.

KeyWords: Multi Step Framework, Query Processing, Dimension Security

I. High-DimensionalSimilaritySearchUsingMulti-StepkNN

Several applications including Image, Medical, Time Series and Document Databases involve high-dimensional data. Similarity retrieval in these applications based on low- dimensional indexes, such as the R*-Tree [1], is very expensive due to the dimensionality curse. Specifically, even for moderate dimensionality (i.e., D = 20) a sequential scan that computes DST(Q, P) for every P □ DB is usually cheaper than conventional NN algorithms using the index. Consequently, numerous specialized structures have been proposed for exact [8] and approximate [22] kNN search in high dimensions.

Itfollows a different approach, combining multi-step search with a dimensionality reduction technique that exhibits the lower bounding property. Specifically, each record P DB is mapped to a low-dimensional representation p in d dimensions (d << D). The resulting d-dimensional dataset db is indexed by an R*- tree, or any low-dimensional index. The query Q is also transformed to a d-dimensional point q and processed using a multi-step method. For instance, in the algorithm of Figure 1, DST (resp. dst) computations involve high (low) dimensional points. The index prunes most nodes and records using the cheap, filter (dst) distances, whereas the expensive DST computations are necessary only for the points in result RSand false hit set*FH*.Their effectiveness is measured by the number of records that they can prune using only the low dimensional representations (i.e., it is inversely proportional to the cardinality of *FH*). Ding et al experimentally compare various techniques, concluding that their effectiveness depends on the data characteristics.

II. Authenticated QueryProcessing

In authenticated query processing, a server maintains a dataset *DB* signed by a trusted authority (e.g., the data owner, a notarization service). The signature *sig* is usually based on a *public-key cryptosystem* (e.g., RSA [16]). The server receives and processes queries from clients. Each query returns a result set *RS* □ *DB* that satisfies certain predicates. Moreover, the client must be able to establish that *RS* is *correct*, i.e., that it contains all records of *DB* that satisfy the query conditions, and that these records have not been modified by the server or another entity. Since *sig* capturestheentire*DB*(includingrecordsnotinthequeryresult), in addition to *RS*, the server returns a *verification object* (*VO*). Given the *VO*, the client can verify *RS* based on *sig* and the signer's public key.

A. TheMR-Tree

The MR-Tree [23] combines the concepts of the MH-Tree and the R*-Tree . A leaf node contains entries *elf* of the form (*pgP*, *P*), where *P* is an indexed point, and *pgP*is a pointer to the page accommodating the record of *P*.

Upon receiving a range query QR, the server performs a depth-first traversal of the MR-Tree, using the algorithm to retrieve the set RS of points in QR. Furthermore, it generates a VOR that contains: (i) all the points outside QR that reside in a leaf MBR overlapping QR, and (ii) a pair (MBRN, hN), for every node N pruned during query processing. In the example of Figure 2, given the shaded range QR, we have RS = {P2, P3, P7}, and VOR = [[[P1, result, result] (MBRN5, hN5)] [[result, P8, P9] (MBRN7, hN7)]]. The token result signifies an object in RS according to the order of appearance in RS. For instance, [P1, result, result] corresponds to node N4; the first occurrence of result refers to P2, and the second one to P3. In order to distinguish the type of each element in the VO, MR_Range includes a header prior to each token, digest,

[1]Corresponding Author

and point in the VO. This header consumes 3 bits, which suffice to represent 8 different element types. For simplicity, we omit the headers in our presentation since the element type is implied by its name.

The verification process of the client is also identical to the one performed for range queries. However, as an adaptation of the R*-tree, the MR-Tree also suffers from the dimensionality curse. Therefore, the application of the afore-mentioned method on high dimensional data has very limited pruning power. Specifically, for numerous dimensions, nearly all leaf nodes must be visited (leading to high server cost); consequently, the majority of points are inserted in the *VO* (leading to high communication overhead); finally, the client has to verify almost the entiredataset.

III. Authenticated Multi-StepNN

It has been proven effective in non-authenticated similarity retrieval, especially for numerous (i.e., D > 100) dimensions, where even high-dimensional indexes fail3; (ii) it can be extended to authenticated query processing based on a low dimensional ADS, i.e., the MR-Tree, whereas, currently there are no authenticated high-dimensional structures; (iii) it is general, i.e., it can also be applied when the expensive distance computations are due to the nature of the distance definition (e.g., network distance), rather than the data dimensionality (in which case D = d).

A. False Hit Reduction Algorithm

Ideally, for each false hit P, ReduceFH should derive the subset SP with the minimum length. Intuitively, this task is at least as difficult as the Knapsack Problem; we need to select a subset of items (SP of P values), each assigned a cost (communication overhead) and a weight (distance DST(SQ, SP)), such that the sum of costs is minimized and the sum of weights exceeds DSTmax. An additional complication is that, when we select one item, the cost of the rest changes (i.e., unlike knapsack, where the cost is fixed).

B. DistributedServers

We assume a client-server architecture, where the server maintains data signed by a trusted authority. There are two versions of the signed dataset: a *D*-dimensional *DB* and a *d*- dimensional *db* (*d* <<*D*), produced from *DB* using any dimensionality reduction technique that satisfies the lower bounding property. For instance, *DB* may be a set of high-dimensional time series and *db* their low dimensional representations obtained by DFT. There is a single signature sig, generated by a public key cryptosystem (e.g., RSA), that captures both DB and db. DST (dst) refers to the distance metric used in the D(d)-dimensional space. For ease of illustration, we use Euclidean distance for both the DST and dst metrics. Nevertheless, the proposed techniques are independent of these metrics, as well as of the underlying dimensionality reduction technique.

Next, we compare SD-AMN and ID-AMN considering that the database is horizontally partitioned over m servers. Recall that the methods first collect distance information, based on which they determine the range that contains the result. The NNs and the false hits are obtained during the verification of this range, which is identical in SD-AMN and ID-AMN. Thus, when measuring the communication cost, we focus on their differences, which regard the transmission of query points and the distance information. The CPU overhead is based again on elementary distance computations. Finally, due to the identical verification process, the client cost is similar, and the corresponding experiments are omitted.

It shows the communication cost as a function of the number m of servers. Since we do not count the common data transmissions, the dominant factor is the number of high-dimensional query (Q) transmissions. SD- AMN sends Q to all servers yielding an overhead of D□m values. On the other hand, ID-AMN transmits Q only to candidate servers. In the best case, all results may be found in a single server, and the rest are eliminated using the dst bound; in the worst case, Q must be sent to all servers, if they all constitute false candidates. In general, the number of eliminated servers increases with their total number, leading to the savings of ID-AMN.Figure 22 compares the two methods on elementary distance computations at the server versus m. The retrieval of a kNN set involves a number of computations linear to (k+|FH|)□(d+D) because the distances of all results and false hits must be evaluated in both low and high-dimensional spaces. In SD-AMN, each of the m servers must retrieve the k NNs; thus, the total cost increases linearly with both m and k. In ID-AMN a server has to perform a number of computations that is proportional to its contribution ki (Šk) in the result set. The value of m affects the number of computations only indirectly, by increasing the false candidates. In general, ID-AMN clearly outperforms SD- AMN in all settings.

C. AMN In DistributedServers

In this setting we assume that the database is horizontallysuccessfully, and for every point *P* in each FH^i it holds *DST*(*Q*, *P*) Š *DSTmax*, then the client is assured that *RS* is correct.Partitioned and distributed over *m* (>1) servers. Specifically, each server S^i stores a subset DB^i such that: DB^1 □ ..□ DB^m= *DB* and DB^i □ DB^j = □, □ 1 Š *i, j* Š *m*. In addition, S^i maintains an MR-Tree on the corresponding reduced data set db^i, which is signed by a signature sig^i.A query result comprises the *k*NNs over all servers. Minimization of***Client t***ransmissions (of the high-dimensional data) is particularlyimportant for this setting, especially for large values of *m*. SD-AMN (short for

simple distributed AMN), used as a benchmark in our experimental evaluation.

D. Simple DistributedAMN

In SD-AMN, a client sends its *k*NN query *Q* to all servers. Each server S^i retrieves the partial result RS^i on the local DB^i using the conventional multi-step algorithm and generates a vector $\boldsymbol{kDST^i}$ with the *distance values* of the *k*NN set RS^i in S^i. The client collects the vectors from the servers and determines the global k^{th}nearest distance*DSTmax*over all $m \square k$collected distances. Then, it transmits a rangeqR=(q,*DSTmax*).EachserverS^iexecutesqRusingitsMR -Tree and returns VO^i, RS^i and FH^i. VO^i has the same.

E. Incremental DistributedAMN

Lines 18-20 simply verify the range qR= (q, *DSTmax*) in eachserver. All the result points (RS^i), as well as false hits (FH^i) are transmitted during this step. The client generates the final result *RS* locally from the union of all RS^i. C-AMN can be applied to reduce the size of false hits. Note that Line 12 may call *get_next_ smallest_DST*multiple times on the same server S^i. In this case, the client needs to transmit the full query *Q* only the first time; for subsequent requests, it suffices to send the query ID.SD-AMN is optimal in terms of high-dimensionalpoint transmissions because the client receives *D*-dimensional representations only for points in *qR*. All these points (results and false hits) are necessary to establish correctness anyway. However, it must transmit *Q* to all servers. Moreover, each server S^i has to compute RS^i although none of the points of RS^i may participate in the global result (e.g., S^4 in Figure 12). ID-AMN avoids these problems by gradually eliminating servers that cannot contribute results. Specifically, ID-AMN incrementally retrieves distance values from servers to compute the final *DSTmax*, postponing local NN computations at the servers until they are required. We present the pseudo code of ID-AMN (at the client) in Figure 14, and explain its functionality by continuing the example of Figure 13 (k =3).

The while loop (Lines 8-17) starts by eliminating eachserver such that DST i Š DSTmax (initially DST i = dst i). For instance, DST 4 = 7 Š DSTmax = 5, and the client discards S4 without sending Q. Since the subsequent verification of S4 does not require Q either, there is no transmission of high- dimensional data (query, or points) between the client and S4. Line 11 selects the candidate server Si with the minimum DST i, and asks for the distance DSTnew of the next NN in Si. If DSTnew Š DSTmax, Si is purged. Assuming that the selected server is S 3 (DST 3 = DST 2 = 2), then DST(Q, P3) = 5.

Algorithm *ID-AMN_client*(*Q, k*)

1. For each serverS^i
2. Set*Candidate*[i]=1;
3. (dst^i, $kdst^i$,)= *get_ smallest_dist*(q,S^i)
4. $DST^i = dst^i$
5. Let S^j be the server with the minimum$kdst^j$
6. Set vector $\boldsymbol{kDST}$= *get_k_smallest_DSTs*(Q,S^j)
7. Set *DSTmax*= maximum value in $\boldsymbol{kDST}$; Set*Candidate*[j]=0
8. While there are candidateservers
9. For each serverS^i
10. If DST^i Š *DSTmax*, Set*Candidate*[i]=0
11. Select candidate server S^i with minimumDST^i
12. Set *DSTnew*= *get_next_ smallest_DST*(Q, S^i) from serverS^i
13. If *DSTnew*□ *DSTmax*, Set *Candidate*[i]=0
14. Else // *DSTnew*<*DSTmax*
15. Insert *DSTnew*into $\boldsymbol{kDST}$
16. Set *DSTmax*= maximum value in $\boldsymbol{kDST}$;
17. $DST^i = DSTnew$
18. For each serverS^i
19. (VO^i, RS^i, FH^i) = *MR_Range*((*q, DSTmax*),$root^i$)
20. *Verify*(VO^i) and incorporate RS^i into*RS*

F. Incremental distributed AMN (client)

Proof of Correctness. The client obtains all results and false hits at the end through the verifiable range (Lines 18-20). As shown in the proof SD-AMN, any DST misreporting that leads to the computation of a DST'max□ DSTmax can be detected by the client. Let us now consider that some server Si sends false dsti and kdsti. The value of kdsti is only used as an estimator for the selection of the initial server (Line 5), and it only affects the efficiency (but not the correctness) of the algorithm. For instance, if S3 reports kdst3 = 1 (instead of 9), it will become the initial server, increasing the communication overhead (S4 cannot be immediately eliminated), without however altering the result. Moreover, as discussed in Section 5.1, any false distance smaller than DSTmax will be caught by the verification. Similarly, dsti is used as a lower bound for DSTi. If Si sends a value of dsti lower than the actual one, itcan only be selected earlier than necessary during the while loop without affecting correctness. On the other hand, if the transmitted dsti

exceeds the actual one, (i) Si is selected later during the loop, or (ii) eliminated altogether if the reported dsti exceeds DSTmax. Case (i) only affects the efficiency, whereas case (ii) is detected during the verification because Si has to send objects within the range qR = (q, DSTmax).

IV. ExperimentalEvaluation

We use four real datasets that capture different combinations of dimensionality *D*, cardinality *N*, and application domain: (i) *Corel* (*D* = 64, *N* = 68040), (ii) *Chlorine* (*D* = 128, *N* = 4310), (iii) *Stock* (*D*= 512, *N* = 10000),and (iv) *Mallat*(*D* = 1024, *N* = 2400). *Corel* [5] can be downloaded from *archive.ics.uci.edu/ml/*, while the rest are available at: *www.cs.ucr.edu/~eamonn/time_series_data/*. We decrease the dimensionality of each dataset using Chebyshev polynomials [4]. The value of *d* is a parameter with range [2, 12] and default value 8. Each reduced dataset is indexed by an MR-Tree using a page size of 4KB. Every digest is created by SHA-1 [12]. We assume that both *DST* and *dst*are based on the Euclidean distance. Section 6.1 compares AMN and C-AMN considering a single server. In this section it evaluates SD-AMN and ID-AMN assuming multiple servers.

A. SingleServer

The measures of interest are the communication overhead, and the CPU cost at the server and the client. We assess the communication overhead based on the verification information sent to the client. The transmission of the query and the result is omitted because it is necessary in any method. The CPU cost is measured in terms of the elementary distance computations. Specifically, D elementary computations are required to derive the Euclidean distance of two D-dimensional points. We exclude the I/O cost at the server because it is identical for both AMN and C-AMN (and similar to that of the conventional multi-step algorithm) since in any case, we have to retrieve the low dimensional NNs using the MR-Tree. For each experiment we select a random data point as the query, and report the average results over 10 queries.

Specifically, the overhead is measured in Mbytes, assuming that each value consumes Sv=8 bytes (a double precision number) and each digest is Sh=20 bytes (typical size for SHA-1). We indicate the number |FH| of false hits below the x-axis. As d increases,|FH| drops because the reduced representation captures more information about the corresponding point. In all cases, C-AMN leads to a significant decline of the overhead. The savings grow with D, and exceed an order of magnitude for Mallat, because long series provide more optimizations opportunities. On the other hand, the gains decrease as d grows due to the smaller FH. In order to demonstrate the effect of the false hits, we include inside each column of the diagrams, the contribution of FH as a percentage of the total overhead. For high D and low d, FH constitutes the dominant factor, especially for AMN (e.g., at least 98% in Mallat), corroborating the importance of C- AMN.

The absolute overhead is lower (in both AMN and C-AMN) for high values of d due to the decrease of |FH|. The exception is Corel, where the communication cost actually grows when d exceeds a threshold (8 for AMN, 4 for C-AMN). This is explained as follows. A typical record (i.e., image) in Corel has very low values (<0.005) on most (>60) dimensions, and relatively large values (>0.1) on the rest. Furthermore, the large values of different records usually concentrate on different dimensions. Dimensionality reduction using Chebyshev polynomials [4] captures effectively those important dimensions even for low d. Consequently, there is a small number of false hits (for d=2,|FH| □ 0.28% of N, whereas in the other datasets |FH| is 50-75% of N). As d grows, |FH| does not drop significantly; on the other hand, the verification information transmitted to the client contains more boundary records and node MBRs, increasing the VO size.

The server computations versus the number of required neighbors. The cost increases slightly with k, but similar to the effect is not as pronounced as that of d. Note that the diagrams do not include the I/O cost, which is identical to both methods. I/O operations normally dominate the processing overhead (since large records must be retrieved from the disk) and the performance difference of the two methods in terms of the overall cost diminishes. Moreover, the difference of C- AMN and AMN would decrease further (in Figures 3-4), if DST were based on a more expensive distance function than dst (e.g., DTW vs. Euclidean distance as in [10]) and applied the optimization . This is because ReduceFH would entail only cheap dst computations, which would be dominated by the more expensive DST calculations, common in both methods.

The number of elementary distance computations at the client as a function of d. C-AMN leads to significant gains, sometimes dropping the processing cost by more than an order of magnitude. Since this cost is proportional to the amount of data received by the client, the diagrams are co-related to those in Figure 6; accordingly, the benefits of C-AMN are more significant for settings that involve large values of D, and |FH|. Similar to Figures 2 and 4, the CPU cost increases with k, but the impact of k is rather small.

Summarizing, compared to AMN, D-AMN for an increasing number of modern database applications, efficient support of similarity search becomes an important task. Along with the complexity of the objects such as images, molecules and mechanical parts, also the complexity of the similarity models increases more and more. Whereas algorithms that are directly based on indexes work well for simple medium-dimensional

similarity distance functions, they do not meet the efficiency requirements of complex high-dimensional and adaptable distance functions. The use of a multi-step query processing strategy is recommended in the cases of high dimensional and adaptable distance functions because the number of candidates which are produced in the filter step and exactly evaluated in the refinement step is a fundamental efficiency parameter. After revealing the strong performance shortcomings of the state-of-the-art algorithm for k-nearest neighbor search, a novel multi-step algorithm which is guaranteed to produce the minimum number of candidates.

V. Conclusion

The importance of authenticated query processing increases with the amount of information available at sources that are untrustworthy, unreliable, or simply unfamiliar. This is the first work addressing authenticated similarity retrieval from such sources using the multi-step kNN framework. We show that a direct integration of optimal NN search with an authenticated data structure incurs excessive communication overhead. Thus, we develop C-AMN, a technique that addresses the communication-specific aspects of NN, and minimizes the transmission overhead and verification effort of the clients. Furthermore, we propose ID-AMN, which retrieves distance information from distributed servers, eliminating those that cannot contribute results.

References

[1]. Beckmann, N., Kriegel, H.-P., Schneider, R., Seeger, B. The R*- Tree: An Efficient and Robust Access Method for Points and Rectangles. SIGMOD, 1990.

[2] Beyer, K., Goldstein, J., Ramakrishnan, R., Shaft, U. When is ″Nearest Neighbor″ Meaningful? ICDT, 1999.

[3] Bryan R. The Digital rEvolution: the Millennial Change in Medical Imaging. Radiology, November 2003.

[4] Cai, Y., Ng, R. Indexing Spatio-Temporal Trajectories with Chebyshev Polynomials. SIGMOD, 2004.

[5] Ding, H., Trajcevski, G., Scheuermann, P., Wang, X., Keogh, E. Querying and Mining of Time Series Data: Experimental Comparison of Representations and Distance Measures. PVLDB, 2008.

[6] Faloutsos, C., Ranganathan, M., Manolopoulos, Y. Fast Subsequence Matching in Time-Series databases. SIGMOD, 1994.

[7] Hjaltason, G., Samet, H. Distance Browsing in Spatial Databases. ACM TODS, 24(2), pp.265-318, 1999.

[8] Jagadish, H., Ooi, B., Tan, K., Yu, C., Zhang, R. i-Distance: an Adaptive B+-tree Based Indexing Method Nearest Neighbor Search. ACM TODS, 30(2), pp.364-397, 2005.

[9] Kellerer, H., Pferschy, U., Pisinger, D. Knapsack Problems. Springer, 2004.

[10] Keogh, E., J., Ratanamahatana, C., A. Exact Indexing of Dynamic Time Warping. Knowl.Inf. Syst., 7(3), 2005.

[11] Korn, F., Sidiropoulos, N., Faloutsos, C., Siegel, E., Protopapas, Z. Fast Nearest Neighbor Search in Medical Image Databases. VLDB, 1996.

[12] Kundu, A., Bertino, E. Structural Signatures for Tree Data Structures. PVLDB, 2008.

[13] Li, F., Hadjieleftheriou, M., Kollios, G., Reyzin, L. Dynamic Authenticated Index Structures for Outsourced Databases. SIGMOD, 2006.