Exploring State-of-the-Art Nearest Neighbor (NN) Search Techniques

Parth Nagarkar
nagarkar@nmsu.edu
New Mexico State University
Las Cruces, NM, USA

Arnab Bhattacharya
arnabb@cse.iitk.ac.in
Indian Institute of Technology
Kanpur, India

Omid Jafari
ojafari@nmsu.edu
New Mexico State University
Las Cruces, NM, USA

ABSTRACT

Finding nearest neighbors (NN) is a fundamental operation in many diverse domains such as machine learning, information retrieval, multimedia retrieval, etc. Due to the data deluge and the application of nearest neighbor queries in many applications where fast performance is necessary, efficient index structures are required to speed up finding nearest neighbors. Different application domains have different data characteristics, which require different types of indexing techniques. While the internal searches are often hidden from the top-level application, it is beneficial for a data scientist to understand these fundamental operations and choose a correct indexing technique to improve the performance of the overall end-to-end workflow. Choosing the correct index structure to solve a Nearest Neighbor query can be a daunting task. A wrong choice can potentially lead to low accuracy, slower execution times, or in worst cases, both. The objective of this tutorial is to present the audience with the knowledge to choose the correct index structure for their specific task application. We present the state-of-the-art Nearest Neighbor indexing techniques for different data characteristics. We also present the effect, in terms of time and accuracy, of choosing the wrong index structure for different application needs. We conclude the tutorial with a discussion on the future challenges in the Nearest Neighbor search domain.

CCS CONCEPTS

• **Information systems → Nearest-neighbor search**.

KEYWORDS

Nearest Neighbor Search, Similarity Search, High-dimensional Spaces, Data-independent Hashing, Product Quantization, Proximity-based ANN

ACM Reference Format:
Parth Nagarkar, Arnab Bhattacharya, and Omid Jafari. 2021. Exploring State-of-the-Art Nearest Neighbor (NN) Search Techniques. In *CODS-COMAD '21: ACM Conference on Data Sciences and Management of Data, January 02–04, 2021, Bangalore, India.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.XXX.XXX

CODS-COMAD '21, January 02–04, 2021, Bangalore, India

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.XXX.XXX

1 TUTORIAL OUTLINE

In this section, we present the outline of our proposed tutorial. In the first part of our tutorial, we will review the motivation, the Nearest Neighbor problem definition, and the seminal and state-of-the-art index structures that improved the efficiency and retrieval of Nearest Neighbor queries. We will particularly focus on the following distinctions: index structures designed for in-memory vs external memory and low-dimensional vs high-dimensional data.In the second part of our tutorial, we will include a hands-on coding session in which we will present the different state-of-the-art Nearest Neighbor techniques for different data and application needs. We will especially focus on the important challenge of parameter tuning and show the effects of wrong parameters on the overall query performance and/or accuracy.

Motivation: Finding nearest neighbors (NN) is a fundamental operation in many tasks in different domains. For example, finding nearby points of interest (such as restaurants, etc.) is a popular query operation in geospatial applications. While Convolutional Neural Networks (CNNs) are commonly used to extract important features from images, similar images are found by searching for nearest neighbors of the extracted features [20]. The main difference between these two applications is the data dimensionality. Generally, while geospatial applications produce data that are 2D (latitude and longitude information for each point), image feature vectors are high-dimensional in nature (> 100). While traditional hierarchical index structures, such as R-trees [10] or k-d trees [3], work effectively for low-dimensional (< 10) data, they suffer from the notorious *curse of dimensionality* for moderate and high-dimensional data, where their performance is often worse than a brute-force linear search. Additionally, when large data cannot be stored in main memory, index structures need to be particularly designed for external memory storage and retrieval. The goal of our tutorial is to present these differences in application requirements, data characteristics, and their effect of the index design.

Definition: We first formally define the Nearest Neighbor search problem. Given a d-dimensional database $\mathcal{D}$, $\mathcal{D}$ consists of n d-dimensional points that belong to a bounded space $\mathbb{R}^d$. Given a query q, the goal of the NN problem is to find a point $x \in \mathcal{D}$ such that $||q, x|| \leq ||q, p||$ for any point $p \in \mathcal{D}$. Here, $||x_1, x_2||$ denotes the Euclidean distance between two points x_1 and x_2.[1] When this definition is generalized to find k nearest neighbor points, the query is called a *k-NN* query. Similarly, a *range* query returns any nearby point p if $||q, p|| \leq R$ is satisfied for a specified search radius R.

Range Queries in Low-Dimensional Spaces: B+-tree is one of the most popularly used disk-based hierarchical index structure for

[1] Due to its popularity and wide applicability, we will assume that the distance metric is Euclidean in this tutorial.

efficiently solving range queries on one-dimensional data. For For spatial data, R-tree [11] and its variants group nearby data objects and represent them in Minimum Bounding Rectangles (MBRs) in each level of the tree. R-trees are very effective disk-based spatial index structures. Hierarchical index structures such as M-trees [5] leverage the triangular inequality property of metric distances. Instead of vector space representation of the data points, M-trees work directly on the distance between any two given points. Additionally, space-filling curves, such as the popular Z-order curves [17] and Hilbert curves [12], are used to convert low-dimensional data to one-dimensional data, which can then be indexed using B+-trees (or other hierarchical index structures).

Approximate Nearest Neighbors (ANN) in High-Dimensional Spaces: For moderate to high-dimensional data ($d > 10$), traditional hierarchical index structures are often out-performed even by linear scans [4] due to the well-known phenomenon *Curse of Dimensionality*. One common solution to the *Curse of Dimensionality* problem is to look for *approximate* results instead of exact results. In applications where 100% accuracy is not necessary, finding results that are *good enough* is much faster than searching for exact results [6]. Approximate solutions offer a trade-off between accuracy and performance. The approximate version of the nearest neighbor problem, also called *c-approximate Nearest Neighbor search* (c-ANN), is to return points that are within $c * R$ distance from the query object. Here, $c > 1$ is a user-defined approximation ratio and R is the distance of the query object from its nearest neighbor. When the *c-approximate Nearest Neighbor search* is also generalized to find k nearest neighbor points, it is called the *c-k-ANN* problem. Popular techniques for solving the Approximate Nearest Neighbor (ANN) problem can be classified into external-memory based techniques and in-memory based techniques.

External-memory based ANN Techniques: Locality Sensitive Hashing (LSH), first proposed in [9], is one of the most popular external-memory based ANN techniques. The goal of LSH is to create random projections and hash data points to these random projections. The intuition behind LSH is that points that are nearby in the original high-dimensional space will be mapped to same (or nearby) hash buckets in the random projections. While it was originally proposed for the Hamming distance, it was later extended to the Euclidean distance in the seminal work E2LSH [6]. There are two main benefits of LSH: 1) LSH provides sub-linear query performance in terms of the data size, and 2) LSH provides theoretical guarantees on the accuracy of the return result. While the original LSH architecture suffered from large index sizes, state-of-the-art LSH techniques, namely C2LSH [7] and QALSH [13], proposed a *Collision Counting* method that solved this issue. In [7], the authors theoretically showed that two nearby points in the original space would collide in at least l (out of m) hash layers for a user-input probability -1δ. By using this proposed technique of *Collision Counting*, C2LSH created only one hash function per hash layer, and hence reduced the index size while presenting accurate and fast results. QALSH [13] proposed *query-aware* hash functions where the hash bucket boundaries are decided after the query arrives, hence resulting in more accurate results. Recently, the state-of-the-art HD-Index [2] was proposed that utilized Hilbert key-based hierarchical trees to represent lower-dimensional disjoint partitions of the original space. HD-Index leverages the triangular inequality and the Ptolemaic inequality further prune the candidates for a given query.

In-memory based ANN Techniques: There are several in-memory based techniques proposed to solve the ANN problem. One of the popular in-memory based libraries to solve the ANN problem is Flann [18]. The goal of Flann is to automatically choose the best ANN algorithm and optimum parameters for a given dataset. In particular, Flann uses hierarchical k-means trees or multiple randomized k-d trees depending on the dataset characteristics. Another in-memory based method is to use Proximity graphs for efficient searching [1, 15, 16]. These techniques design approximations for graphs such as Delaunay graph, Navigable Small-World Networks, Relative Neighborhood Graphs, etc. Product Quantization [14] and its variants [8, 19] use the vector quantization approach to create compact codes for high-dimensional vectors. These techniques divide the original high-dimensional space into the Cartesian product of a low-dimensional subspaces. These subspaces are then quantized independently. The key benefit is the representation of the quantized subspaces using compact codes.

1.1 Coding Session

In the coding session, we will execute the well-known nearest neighbor search algorithms on several scenarios. The goal of this session is to experimentally show the audience how to run different NN search algorithms and familiarize them with the different user-defined parameters of each algorithm. We will specifically focus on presenting the following scenarios:[2]

- **Scenario 1:** We will consider multiple geospatial applications in this scenario. Our goal is to present techniques that work for low to moderate dimensional data. Specifically, we will use 2 datasets (UrbanGB and Travel Reviews) in this scenario.[3] We will look for top-k objects to different queries in these spatial datasets. We will present the results for using Space-filling curves with a B+-tree, R-tree, and a k-d tree. We will present the effect of parameters such as the block size in these index structures.
- **Scenario 2:** In this scenario, we will use the popular high-dimensional image dataset, Sift1M (dimensionality = 128).[4] The goal of this scenario is to present high-dimensional data and their corresponding queries. We will also present the effect of curse of dimensionality on hierarchical structures presented in Scenario 1. Importantly, we will also show the benefit of approximate searching, i.e. why approximate results are preferred over exact results for high-dimensional data. We will also compare the times of hierarchical structures with linear scan results to emphasize the *curse of dimensionality*. Additionally, we will present the popular ANN techniques of HD-Index, HNSW, and OPQ in this scenario. We will present the effect of important parameters such as allowed error probability.
- **Scenario 3:** In this scenario, we will use a popular real very high-dimensional dataset, P53 (dimensionality = 5409), that

[2] For experiments that take longer time to finish, we will precompute the results and present the execution times.
[3] http://archive.ics.uci.edu/ml
[4] http://corpus-texmex.irisa.fr

represents biophysical models of mutant p53 proteins.[5]. We will show how popular ANN techniques fare for very high-dimensional datasets.

In our experiments, we plan to show indexing time, query processing time, space usage, and the I/O usage (for external-memory based algorithms).

2 DURATION

The proposed tutorial will be 3 hours long. We will spend 1.5 hours reviewing the most important, along with state-of-the-art, works in the NN domain. We will then spend 1.5 hours with hands-on coding (especially parameter tuning) of different index structures for different data characteristics.

3 DETAILS OF PREVIOUS TUTORIALS

The authors have not presented any prior tutorials on this topic.

4 GOALS OF THIS TUTORIAL

Finding Nearest Neighbors (NN) is a well studied problem in the database community. Traditional tree-based structures are well-suited to solve the low-dimensional NN problem, while hashing-based index structures are popularly used to find nearest neighbors in high-dimensional spaces. Due to its application in various domains (with potentially different characteristics), different indexing techniques are necessary to solve the NN problem. Our main goal in this tutorial is to present the audience with the knowledge to choose the correct index structure given specific requirements. Often, the internal data structures used to find Nearest Neighbors are hidden from the top-level application. The efficiency and accuracy (for finding approximate nearest neighbors) of an index structure can change drastically for different data characteristics. Understanding these differences between different index structures (and their appropriate usage) can lead to a significant performance improvement of the overall end-to-end workflow. This tutorial is specifically designed for data scientists who need to efficiently find Nearest Neighbors in their end-to-end application workflow (e.g. using machine learning techniques to find similar multimedia objects).

5 TARGET AUDIENCE

The tutorial will include the review of technical details of seminal works and state-of-the-art index structures for finding nearest neighbors. The technical material in the tutorial is appropriate for anyone with a Bachelors degree in Computer Science. The codes presented in the tutorial will be in popular languages such as C++ and Java. Knowledge about metric spaces, Euclidean distance measure, and traditional data structures such as Binary trees, B+-trees, etc. is necessary. We will present the formal definitions of the nearest neighbor problem (and its variants), its usage in various real-world applications, and the popular state-of-the-art index structures that efficiently solve the NN problem in this tutorial.

6 PROPOSERS

The tutorial will be presented by the following presenters:

- **Parth Nagarkar** is an Assistant Professor in Computer Science at New Mexico State University. His research is broadly in the exciting area of big data management. He is particularly interested in building scalable index structures and distributed systems which empower efficient large-scale, high-dimensional data processing. In the tutorial, he will be in charge of introducing the motivation and the Nearest Neighbor problem, and then present the Approximate Nearest Neighbor (ANN) problem in high-dimensional spaces, along with different techniques that are commonly used to solve the ANN problem.
- **Arnab Bhattacharya** is an Associate Professor in Computer Science at the Indian Institute of Technology, Kanpur. His research interests are broadly in database querying. His recent works have focused on high-dimensional indexing and graph querying. In this tutorial, he will be in charge of presenting the Nearest Neighbor problem in low-dimensional spaces and the popular techniques used to efficiently solve this problem. He will also discuss the future challenges in the NN domain.
- **Omid Jafari** is a Ph.D. student at New Mexico State University. His research focuses on improving Approximate Nearest Neighbor techniques such as Locality Sensitive Hashing. He will be responsible for presenting the different state-of-the-art techniques in the Nearest Neighbor problem domain during the coding session of the tutorial.

6.1 Availability

The above three members of this tutorial team will be presenting during the tutorial.

7 REFERENCES

In this tutorial, we will focus on the following seminal and state-of-the-art works in the Nearest Neighbor problem domain: R-trees [10], kd-trees [3], M-trees [5], Locality Sensitive Hashing (LSH) and its variants [6, 7, 9, 13], HD-Index [2], Proximity graph-based techniques [1, 15, 16] and Product Quantization and its variants [8, 14, 19].

8 RESEARCH CHALLENGES

At the end of the tutorial, we will discuss potential research directions in the NN domain. With the ever increasing data and the need for faster answers, approximate query processing is a very important research topic. Many approximate techniques, such as LSH, give theoretical guarantees on the query results. Existing approximate techniques focus on giving theoretical guarantees on the recall, but not on the often used evaluation metric, mean average precision (mAP). We will discuss these challenges in this tutorial. Additionally, we will present the potential research directions in creating distributed and parallel versions of popular techniques in the NN domain.

[5] https://archive.ics.uci.edu/ml/datasets/p53+Mutants

REFERENCES

[1] Kazuo Aoyama, Kazumi Saito, Hiroshi Sawada, and Naonori Ueda. 2011. Fast Approximate Similarity Search Based on Degree-Reduced Neighborhood Graphs. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge*

Discovery and Data Mining (KDD '11). Association for Computing Machinery, New York, NY, USA, 1055–1063. https://doi.org/10.1145/2020408.2020576

[2] Akhil Arora, Sakshi Sinha, Piyush Kumar, and Arnab Bhattacharya. 2018. HD-index: Pushing the Scalability-accuracy Boundary for Approximate kNN Search in High-dimensional Spaces. *Proc. VLDB Endow.* 11, 8 (April 2018), 906–919. https://doi.org/10.14778/3204028.3204034

[3] Jon Louis Bentley. 1975. Multidimensional Binary Search Trees Used for Associative Searching. *Commun. ACM* 18, 9 (Sept. 1975), 509–517. https://doi.org/10.1145/361002.361007

[4] Edgar Chávez, Gonzalo Navarro, Ricardo Baeza-Yates, and José Luis Marroquín. 2001. Searching in Metric Spaces. *ACM Comput. Surv.* 33, 3 (Sept. 2001), 273–321. https://doi.org/10.1145/502807.502808

[5] Paolo Ciaccia, Marco Patella, and Pavel Zezula. 1997. M-Tree: An Efficient Access Method for Similarity Search in Metric Spaces. In *Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB '97)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 426–435.

[6] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. 2004. Locality-Sensitive Hashing Scheme Based on p-Stable Distributions. In *Proceedings of the Twentieth Annual Symposium on Computational Geometry (SCG '04)*. Association for Computing Machinery, New York, NY, USA, 253–262. https://doi.org/10.1145/997817.997857

[7] Junhao Gan and Jianlin Feng. 2012. Locality-sensitive hashing scheme based on dynamic collision counting. In *In SIGMOD Conference*. 541–552.

[8] T. Ge, K. He, Q. Ke, and J. Sun. 2013. Optimized Product Quantization for Approximate Nearest Neighbor Search. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*. 2946–2953.

[9] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. 1999. Similarity Search in High Dimensions via Hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases (VLDB '99)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 518–529.

[10] Antonin Guttman. 1984. R-Trees: A Dynamic Index Structure for Spatial Searching. *SIGMOD Rec.* 14, 2 (June 1984), 47–57. https://doi.org/10.1145/971697.602266

[11] Antonin Guttman. 1984. R-Trees: A Dynamic Index Structure for Spatial Searching. In *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data (SIGMOD '84)*. Association for Computing Machinery, New York, NY, USA, 47–57. https://doi.org/10.1145/602259.602266

[12] D. Hilbert. 1891. Über die stetige Abbildung einer Linie auf ein Flächenstück.

[13] Qiang Huang, Jianlin Feng, Yikai Zhang, Qiong Fang, and Wilfred Ng. 2015. Query-aware Locality-sensitive Hashing for Approximate Nearest Neighbor Search. *Proc. VLDB Endow.* 9, 1 (Sept. 2015), 1–12. https://doi.org/10.14778/2850469.2850470

[14] H. Jégou, M. Douze, and C. Schmid. 2011. Product Quantization for Nearest Neighbor Search. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33, 1 (2011), 117–128.

[15] Yury Malkov, Alexander Ponomarenko, Andrey Logvinov, and Vladimir Krylov. 2014. Approximate nearest neighbor algorithm based on navigable small world graphs. *Information Systems* 45 (2014), 61 – 68. https://doi.org/10.1016/j.is.2013.10.006

[16] Yu. A. Malkov and D. A. Yashunin. 2016. Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs. arXiv:cs.DS/1603.09320

[17] G. Morton. 1966. A computer oriented geodetic data base and a new technique in file sequencing.

[18] M. Muja and D. G. Lowe. 2014. Scalable Nearest Neighbor Algorithms for High Dimensional Data. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36, 11 (2014), 2227–2240.

[19] Mohammad Norouzi and David J. Fleet. 2013. Cartesian K-Means. In *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '13)*. IEEE Computer Society, USA, 3017–3024. https://doi.org/10.1109/CVPR.2013.388

[20] F. Radenović, G. Tolias, and O. Chum. 2019. Fine-Tuning CNN Image Retrieval with No Human Annotation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41, 7 (2019), 1655–1668.