Similarity Query Processing for High-Dimensional Data

Jianbin Qin
Shenzhen Institute of Computing Sciences, Shenzhen University
jqin@sics.ac.cn

Wei Wang
University of New South Wales
weiw@cse.unsw.edu.au

Chuan Xiao
Osaka University and Nagoya University
chuanx@ist.osaka-u.ac.jp

Ying Zhang
University of Technology Sydney
Ying.Zhang@uts.edu.au

ABSTRACT

Similarity query processing has been an active research topic for several decades. It is an essential procedure in a wide range of applications. Recently, embedding and auto-encoding methods as well as pre-trained models have gained popularity. They basically deal with high-dimensional data, and this trend brings new opportunities and challenges to similarity query processing for high-dimensional data. Meanwhile, new techniques have emerged to tackle this long-standing problem theoretically and empirically. In this tutorial, we summarize existing solutions, especially recent advancements from both database (DB) and machine learning (ML) communities, and analyze their strengths and weaknesses. We review exact and approximate methods such as cover tree, locality sensitive hashing, product quantization, and proximity graphs. We also discuss the selectivity estimation problem and show how researchers are bringing in state-of-the-art ML techniques to address the problem. By highlighting the strong connections between DB and ML, we hope that this tutorial provides an impetus towards new ML for DB solutions and vice versa.

PVLDB Reference Format:
Jianbin Qin, Wei Wang, Chuan Xiao, and Ying Zhang. Similarity Query Processing for High-Dimensional Data. *PVLDB*, 13(12): 3437-3440, 2020.
DOI: https://doi.org/10.14778/3415478.3415564

1. INTRODUCTION

Similarity query processing is a fundamental and essential procedure in applications of many domains, including databases (DB), machine learning (ML), multimedia, and computer vision. Numerous query processing algorithms have been proposed in the last few decades to deal with various kinds of data types and similarity functions. With the proliferation of deep learning, especially the prevalence of embedding, auto-encoders, and pre-trained models, similarity query processing for high-dimensional data become increasingly important. They benefit DB applications such as entity matching [48] and concept linking [12] as well as ML applications such as multimedia retrieval [7] and adversarial machine learning [1]. In ML community, similarity queries are also studied under the name of k nearest neighbors (k-NN) queries. k-NN itself is also an instance-based learning method for classification and regression.

Our tutorial discusses the importance of similarity query processing for high-dimensional data in a wide range of applications. We summarize existing solutions, especially recent advancements from both DB and ML communities, thereby highlighting the interplay between modern DB and ML technologies. We review technical challenges and various exact and approximate algorithms, including cover tree, locality sensitive hashing, product quantization, and proximity graphs. Moreover, we discuss the selectivity estimation of similarity query processing for high-dimensional data, and show how researchers are bringing in state-of-the-art ML techniques to address this problem. We expect that this tutorial will provide an impetus towards new ML for DB solutions and vice versa.

Scope. This tutorial aims to provide a comprehensive review of similarity query processing methods for high-dimensional data. In particular, we explain the reason why processing similarity queries for high-dimensional data – in contrast to sets and strings (see [45, 73] for survey) which have been extensively studied by the DB community – has become more important. We introduce various existing algorithms and analyze their strengths and weaknesses. We highlight the connections between DB and ML as well as their foci on this topic. Finally, we outline future research directions and open problems to be solved.

Intended Length and Target Audience. This is a **three-hour** tutorial targeting researchers, developers, and practitioners interested in managing high-dimensional data and ML for DB topics. We assume that the target audience is generally familiar with basic DB and ML terms, but there is no requirement for prior knowledge of specific algorithms.

Related Tutorials in Recent Years. This will be the first time that the authors present a tutorial on similarity query processing for high-dimensional data. To the best of our knowledge, two topically related tutorials were presented at recent data-centric research venues (WISE 2017 [55] and CIKM 2019 [40]). These two tutorials target sets and strings, respectively, whose query processing methods are substantially different from those will be presented at this tutorial.

2. TUTORIAL OUTLINE

This tutorial consists of five parts. The first part motivates the need for similarity query processing on high-dimensional data and introduces basic concepts. The second and third

Proceedings of the VLDB Endowment, Vol. 13, No. 12
ISSN 2150-8097.
DOI: https://doi.org/10.14778/3415478.3415564

parts delve into query processing algorithms. The fourth part covers selectivity estimation algorithms. The fifth part discusses miscellaneous issues such as the use of similarity queries with respect to the entire workflow of real applications, deployment in a distributed environment, as well as future directions and open problems.

2.1 Background and Preliminaries

In the introductory part of the tutorial, we first introduce applications and explains the increasing importance of similarity query processing on high-dimensional data, as stated in Section 1. Then we describe basic concepts: (1) data models and the way of which we convert raw data (text, images, etc.) to high-dimensional data; (2) similarity/distance functions, mainly Hamming distance for binary vectors, Euclidean distance, cosine similarity (angular distance), and inner product for real-valued vectors; (3) query types, i.e., search and join queries, or thresholded and k-NN queries, depending on the dimension of categorization; (4) a summary of the solutions that will be elaborated in the rest of the tutorial.

2.2 Exact Query Processing

Exact query processing methods aim to find all the results that satisfy the similarity constraint. Researchers are interested in this type of solutions as it does not pose any uncertainty to the pipelines that apply similarity query processing as a component. It also simplifies empirical comparison as only speed and space consumptions are key evaluation criteria. Existing exact methods usually answer queries by looking up one or more (overlapping or non-overlapping) regions in the original or a transformed space. Partitioning techniques are often employed. These methods can be classified into the following three categories:

Tree-based Methods. These methods partition the database in a hierarchical manner. To process queries, triangle inequality is often used to determine the nodes to be traversed. Representative methods are M-tree [11] and cover tree [5, 27].

Space Partitioning Methods. These methods partition the original space and bound the overall distance using the distance in each subspace. Some methods require a sequential scan of the database, e.g., the vector approximation file (VA-file) [67]. For fast retrieval, indexing methods were proposed to deal with Hamming distance using the pigeonhole principle [51, 54, 56, 57].

Dimensionality Reduction Methods. These methods project objects to another space to reduce dimensionality. They are basically early attempts that deal with the disk-resident case and aim at reducing disk I/O [3, 52, 9, 74, 28]. Most of them transform the original space to a 1-dimensional space and utilize B^+-trees for indexing.

2.3 Approximate Query Processing

It is commonly believed that it is hard to compute the exact results of queries with a sub-linear cost. Instead, computing approximate results is sufficiently useful for many practical problems, and these solutions empirically achieve significantly higher efficiency and scalability than exact ones [37]. Approximate methods either adopt a space-first (i.e., looking up regions in a space) or an object-first (i.e., looking up objects directly) strategy to find query results.

Locality Sensitive Hashing. Locality sensitive hashing (LSH) is a data-independent space-first approach with probabilistic guarantees on the worst-case performance [26, 20, 15, 62]. It relies on a family of hash functions that maps objects to another space such that similar objects are mapped to the same hash codes with higher probability than dissimilar objects. Recent development focuses on supporting various similarity measures [46, 76] and space-efficient indexing [61, 25, 75].

Learning to Hash. Learning to hash (L2H) is a data-dependent space-first approach that maps data to another space by exploiting the data distribution. The main principle of most methods in this category is to preserve the similarity information within an appropriate neighborhood. Additional heuristics and optimizations are often added to further reduce the information loss caused by the mapping or increase generalization to unseen data. According to the optimization objective to preserve similarity, L2H algorithms can be grouped into pairwise-similarity persevering class [68, 22, 39], multiwise-similarity persevering class [64, 63], and implicitly-similarity persevering class [30, 31]. Recently, deep learning-based L2H methods were proposed, in both supervised and unsupervised manner [6, 41, 70, 59].

Space Partitioning Methods. This category is a space-first approach that divides the high-dimensional space into multiple regions. Partition is often carried out in a recursive way, so the index is represented by a tree or a forest. Based on the way of partitioning, there are mainly three classes of methods: Pivoting methods divide the objects based on the distance from the object to some (usually randomly chosen) pivots; e.g., VP-Tree [71] and ball tree [8]. Hyperplane partitioning methods recursively divide the space by a hyperplane with a random direction (e.g. Annoy [4], random projection tree [14]) or an axis-aligned separating hyperplane (e.g., randomized kd-trees [60, 49]). Compact partitioning methods either divide the objects into clusters [18] or create possibly approximate Voronoi partitions [50, 5] to exploit locality. Another line of methods is based on product quantization [29, 19, 32, 24], with the unique ability to handle billions of objects.

Neighborhood-based Methods. This category is an object-first approach that constructs a proximity graph where nodes represent objects and edges connect nearby objects. The main idea is to perform a search for similar objects atop the proximity graph. They achieve top accuracy and speed trade-off in many empirical evaluations [37, 2]. The first class of these methods tries to build a k-NN graph [16] or its variant [37] which records the k-NN of each object. Then nearest neighbor search is conducted by the hill-climbing strategy. The second class employs the navigable small world graph [43, 44], an undirected graph that contains an approximation of the Delaunay graph and has long-range links with the small world navigation property [34]. Hierarchical navigable small world [44] is one of the most efficient algorithms thus far and support incremental update. Recently, learning-based methods were proposed to provide a more efficient search path in the graph [53]. The third class is based on the relative neighborhood graph [17], which considers connectivity, degree, shortest path length, and index size to achieve robust empirical performance.

2.4 Selectivity Estimation

Selectivity estimation outputs the approximate number of objects that satisfy a selection criterion. Due to its use in density estimation, outlier detection, image retrieval, and query optimization, this problem has received considerable attention recently. For example, hands-off entity matching systems [21, 13] extract paths from random forests and take each path (a conjunction of similarity predicates over multiple attributes)

as a blocking rule, and thus selectivity estimation is useful for choosing the execution order of query plans that involve multiple similarity predicates. A traditional database method is based on importance sampling [69]. Kernel density estimation [23, 47] tailored to this problem has also been developed. A recent trend is to formalize it as a regression task and utilize ML methods, e.g., by XGBoost [10], LightGBM [33], the mixture of expert model [58], or the recursive model indexes [35]. Another line of work targets monotonic estimation by employing deep lattice network [72], deep regression with incremental prediction [65], or piece-wise linear functions [66].

2.5 Future Opportunities

We highlight a number of promising directions for future research: (1) It is interesting to explore ML models as solutions to query processing (e.g., learned indexing or sampling). (2) Whilst many existing studies target search queries, we expect that join queries will be explored, especially for the cold start case. (3) Answering composite queries (e.g., conjunctive queries) over multiple attributes will receive more attention, since many DB tasks deal with multi-attribute data and the advancement of deep learning methods will enable us to embed more attributes for semantic comparison. (4) Another direction is to develop efficient algorithms for query processing in data science platforms such as Pandas/R dataframe.

3. BIOGRAPHIES OF PRESENTERS

The four presenters have rich experience in the research on similarity queries for high-dimensional data, and have made significant contributions [36, 37, 38, 54, 56, 57, 61, 42, 65, 66].

Jianbin Qin is a Research Scientist with Shenzhen Institute of Computing Sciences, Shenzhen University. He received the Ph.D. degree from the University of New South Wales in 2013. His research interests include similarity query processing, data integration, textual databases, and information retrieval. He has given a tutorial at WISE 2017.

Wei Wang is a Professor with the University of New South Wales. He received the Ph.D. degree from the Hong Kong University of Science and Technology in 2004. His research interests include high-dimensional data management, similarity query processing, data integration, knowledge graphs, natural language processing, and adversarial machine learning. He has given tutorials at SIGMOD 2009 and ICDE 2011.

Chuan Xiao is an Associate Professor with Osaka University and Nagoya University. He received the Ph.D. degree from the University of New South Wales in 2010. His research interests include similarity query processing, data integration, textual databases, spatio-temporal databases, and graph databases. He has given a tutorial at WISE 2017.

Ying Zhang is a Professor and ARC Future Fellow with the University of Technology, Sydney, and the Head of the Database Group at the Centre for Artificial Intelligence. He received the Ph.D. degree from the University of New South Wales in 2008. His research interests include high-dimensional data management, scalable data analytics, data streams, and graph databases. He has given a tutorial at ICDE 2019.

Acknowledgments. This work was supported by JSPS 17H06099, 18H04093, and 19K11979, NSFC 61702409, Guangdong Basic and Applied Basic Research Foundation 2019A1515111047, 2019A1515011064, Guangdong Project 2017B030314073 and 2018B030325002, ARC DP170103710, DP180103096, DP180103411, and FT170100128, and D2D CRC DC25002 and DC25003. We thank Yaoshu Wang (SICS) for his kind advice.

4. REFERENCES

[1] M. Alzantot, Y. Sharma, A. Elgohary, B. Ho, M. B. Srivastava, and K. Chang. Generating natural language adversarial examples. In *EMNLP*, pages 2890–2896, 2018.

[2] M. Aumüller, E. Bernhardsson, and A. J. Faithfull. Ann-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. *Inf. Syst.*, 87, 2020.

[3] S. Berchtold, C. Böhm, and H. Kriegel. The pyramid-technique: Towards breaking the curse of dimensionality. In *SIGMOD*, pages 142–153, 1998.

[4] E. Bernhardsson. Annoy at github `https://github.com/spotify/annoy`, 2015.

[5] A. Beygelzimer, S. Kakade, and J. Langford. Cover trees for nearest neighbor. In *ICML*, pages 97–104, 2006.

[6] D. Cai, X. Gu, and C. Wang. A revisit on deep hashings for large-scale content based image retrieval. *CoRR*, abs/1711.06016, 2017.

[7] Y. Cao, M. Long, B. Liu, and J. Wang. Deep cauchy hashing for hamming space retrieval. In *CVPR*, pages 1229–1237, 2018.

[8] L. Cayton. Fast nearest neighbor retrieval for bregman divergences. In *ICML*, pages 112–119, 2008.

[9] K. Chakrabarti and S. Mehrotra. Local dimensionality reduction: A new approach to indexing high dimensional spaces. In *VLDB*, pages 89–100, 2000.

[10] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *KDD*, pages 785–794, 2016.

[11] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *VLDB*, pages 426–435, 1997.

[12] J. Dai, M. Zhang, G. Chen, J. Fan, K. Y. Ngiam, and B. C. Ooi. Fine-grained concept linking using neural networks in healthcare. In *SIGMOD*, pages 51–66, 2018.

[13] S. Das, P. S. G. C., A. Doan, J. F. Naughton, G. Krishnan, R. Deep, E. Arcaute, V. Raghavendra, and Y. Park. Falcon: Scaling up hands-off crowdsourced entity matching to build cloud services. In *SIGMOD*, pages 1431–1446, 2017.

[14] S. Dasgupta and Y. Freund. Random projection trees and low dimensional manifolds. In *STOC*, pages 537–546, 2008.

[15] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *SoCG*, pages 253–262, 2004.

[16] W. Dong. *High-dimensional similarity search for large datasets*. Princeton University, 2011.

[17] C. Fu, C. Xiang, C. Wang, and D. Cai. Fast approximate nearest neighbor search with the navigating spreading-out graph. *PVLDB*, 12(5):461–474, 2019.

[18] K. Fukunaga and P. M. Narendra. A branch and bound algorithms for computing k-nearest neighbors. *IEEE Trans. Computers*, 24(7):750–753, 1975.

[19] T. Ge, K. He, Q. Ke, and J. Sun. Optimized product quantization. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(4):744–755, 2014.

[20] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *VLDB*, pages 518–529, 1999.

[21] C. Gokhale, S. Das, A. Doan, J. F. Naughton, N. Rampalli, J. W. Shavlik, and X. Zhu. Corleone: hands-off crowdsourcing for entity matching. In *SIGMOD*, pages 601–612, 2014.

[22] J. He, W. Liu, and S. Chang. Scalable similarity search with optimized kernel hashing. In *KDD*, pages 1129–1138, 2010.

[23] M. Heimel, M. Kiefer, and V. Markl. Self-tuning, GPU-accelerated kernel density models for multidimensional selectivity estimation. In *SIGMOD*, pages 1477–1492, 2015.

[24] J. Heo, Z. Lin, X. Shen, J. Brandt, and S. Yoon. Shortlist selection with residual-aware distance estimator for k-nearest neighbor search. In *CVPR*, pages 2009–2017, 2016.

[25] Q. Huang, J. Feng, Q. Fang, W. Ng, and W. Wang. Query-aware locality-sensitive hashing scheme for lp norm. *VLDB J.*, 26(5):683–708, 2017.

[26] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *STOC*, pages 604–613, 1998.

[27] M. Izbicki and C. R. Shelton. Faster cover trees. In *ICML*, pages 1162–1170, 2015.

[28] H. V. Jagadish, B. C. Ooi, K.-L. Tan, C. Yu, and R. Zhang. idistance: An adaptive b^+-tree based indexing method for nearest neighbor search. *ACM Trans. Database Syst.*, 30(2):364–397, 2005.

[29] H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(1):117–128, 2011.

[30] Z. Jin, Y. Hu, Y. Lin, D. Zhang, S. Lin, D. Cai, and X. Li. Complementary projection hashing. In *ICCV*, pages 257–264, 2013.

[31] A. Joly and O. Buisson. Random maximum margin hashing. In *CVPR*, pages 873–880, 2011.

[32] Y. Kalantidis and Y. Avrithis. Locally optimized product quantization for approximate nearest neighbor search. In *CVPR*, pages 2329–2336, 2014.

[33] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T. Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *NIPS*, pages 3149–3157, 2017.

[34] J. M. Kleinberg. Navigation in a small world. *Nature*, 406(6798):845, 2000.

[35] T. Kraska, A. Beutel, E. H. Chi, J. Dean, and N. Polyzotis. The case for learned index structures. In *SIGMOD*, pages 489–504, 2018.

[36] M. Li, Y. Zhang, Y. Sun, W. Wang, I. W. Tsang, and X. Lin. I/O efficient approximate nearest neighbour search based on learned functions. In *ICDE*, 2020.

[37] W. Li, Y. Zhang, Y. Sun, W. Wang, M. Li, W. Zhang, and X. Lin. Approximate nearest neighbor search on high dimensional data-experiments, analyses, and improvement. *IEEE Trans. Knowl. Data Eng.*, 2019.

[38] W. Liu, H. Wang, Y. Zhang, W. Wang, and L. Qin. I-LSH: I/O efficient c-approximate nearest neighbor search in high-dimensional space. In *ICDE*, pages 1670–1673, 2019.

[39] W. Liu, J. Wang, S. Kumar, and S. Chang. Hashing with graphs. In *ICML*, pages 1–8, 2011.

[40] J. Lu, C. Lin, J. Wang, and C. Li. Synergy of database techniques and machine learning models for string similarity search and join. In *CIKM*, pages 2975–2976, 2019.

[41] J. Lu, V. E. Liong, and J. Zhou. Deep hashing for scalable image search. *IEEE Trans. Image Processing*, 26(5):2352–2367, 2017.

[42] K. Lu, H. Wang, W. Wang, and M. Kudo. VHP: approximate nearest neighbor search via virtual hypersphere partitioning. *PVLDB*, 13(9):1443–1455, 2020.

[43] Y. Malkov, A. Ponomarenko, A. Logvinov, and V. Krylov. Approximate nearest neighbor algorithm based on navigable small world graphs. *Inf. Syst.*, 45:61–68, 2014.

[44] Y. A. Malkov and D. A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *CoRR*, abs/1603.09320, 2016.

[45] W. Mann, N. Augsten, and P. Bouros. An empirical evaluation of set similarity join techniques. *PVLDB*, 9(9):636–647, 2016.

[46] G. Marçais, D. F. DeBlasio, P. Pandey, and C. Kingsford. Locality-sensitive hashing for the edit distance. *Bioinform.*, 35(14):i127–i135, 2019.

[47] M. Mattig, T. Fober, C. Beilschmidt, and B. Seeger. Kernel-based cardinality estimation on metric data. In *EDBT*, pages 349–360, 2018.

[48] S. Mudgal, H. Li, T. Rekatsinas, A. Doan, Y. Park, G. Krishnan, R. Deep, E. Arcaute, and V. Raghavendra. Deep learning for entity matching: A design space exploration. In *SIGMOD*, pages 19–34, 2018.

[49] M. Muja and D. G. Lowe. Scalable nearest neighbor algorithms for high dimensional data. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(11):2227–2240, 2014.

[50] G. Navarro. Searching in metric spaces by spatial approximation. *VLDB J.*, 11(1):28–46, 2002.

[51] M. Norouzi, A. Punjani, and D. J. Fleet. Fast exact search in hamming space with multi-index hashing. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(6):1107–1119, 2014.

[52] B. C. Ooi, K. Tan, C. Yu, and S. Bressan. Indexing the edges - A simple and yet efficient approach to high-dimensional indexing. In *PODS*, pages 166–174, 2000.

[53] L. Prokhorenkova. Graph-based nearest neighbor search: From practice to theory. *CoRR*, abs/1907.00845, 2019.

[54] J. Qin, Y. Wang, C. Xiao, W. Wang, X. Lin, and Y. Ishikawa. GPH: similarity search in hamming space. In *ICDE*, pages 29–40, 2018.

[55] J. Qin and C. Xiao. Set similarity query processing. In *WISE*, 2017.

[56] J. Qin and C. Xiao. Pigeonring: A principle for faster thresholded similarity search. *PVLDB*, 12(1):28–42, 2018.

[57] J. Qin, C. Xiao, Y. Wang, W. Wang, X. Lin, Y. Ishikawa, and G. Wang. Generalizing the pigeonhole principle for similarity search in hamming space. *IEEE Trans. Knowl. Data Eng.*, 2019.

[58] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. V. Le, G. E. Hinton, and J. Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *CoRR*, abs/1701.06538, 2017.

[59] F. Shen, Y. Xu, L. Liu, Y. Yang, Z. Huang, and H. T. Shen. Unsupervised deep hashing with similarity-adaptive and discrete optimization. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(12):3034–3044, 2018.

[60] C. Silpa-Anan and R. I. Hartley. Optimised kd-trees for fast image descriptor matching. In *CVPR*, 2008.

[61] Y. Sun, W. Wang, J. Qin, Y. Zhang, and X. Lin. SRS: solving c-approximate nearest neighbor queries in high dimensional euclidean space with a tiny index. *PVLDB*, 8(1):1–12, 2014.

[62] Y. Tao, K. Yi, C. Sheng, and P. Kalnis. Quality and efficiency in high dimensional nearest neighbor search. In *SIGMOD*, pages 563–576, 2009.

[63] J. Wang, W. Liu, A. X. Sun, and Y. Jiang. Learning hash codes with listwise supervision. In *ICCV*, pages 3032–3039, 2013.

[64] J. Wang, J. Wang, N. Yu, and S. Li. Order preserving hashing for approximate nearest neighbor search. In *MM*, pages 133–142, 2013.

[65] Y. Wang, C. Xiao, J. Qin, X. Cao, Y. Sun, W. Wang, and M. Onizuka. Monotonic cardinality estimation of similarity selection: A deep learning approach. In *SIGMOD*, pages 1197–1212, 2020.

[66] Y. Wang, C. Xiao, J. Qin, R. Mao, M. Onizuka, W. Wang, and R. Zhang. Consistent and flexible selectivity estimation for high-dimensional data. *CoRR*, abs/2005.09908, 2020.

[67] R. Weber, H. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *VLDB*, pages 194–205, 1998.

[68] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, pages 1753–1760, 2008.

[69] X. Wu, M. Charikar, and V. Natchu. Local density estimation in high dimensions. In *ICML*, pages 5293–5301, 2018.

[70] Z. Xia, X. Feng, J. Peng, and A. Hadid. Unsupervised deep hashing for large-scale visual search. In *IPTA*, pages 1–5, 2016.

[71] P. N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *SODA*, pages 311–321, 1993.

[72] S. You, D. Ding, K. R. Canini, J. Pfeifer, and M. R. Gupta. Deep lattice networks and partial monotonic functions. In *NIPS*, pages 2981–2989, 2017.

[73] M. Yu, G. Li, D. Deng, and J. Feng. String similarity search and join: a survey. *Frontiers Comput. Sci.*, 10(3):399–417, 2016.

[74] R. Zhang, B. C. Ooi, and K.-L. Tan. Making the pyramid technique robust to query types and workloads. In *ICDE*, pages 313–324, 2004.

[75] B. Zheng, X. Zhao, L. Weng, N. Q. V. Hung, H. Liu, and C. S. Jensen. PM-LSH: A fast and accurate LSH framework for high-dimensional approximate NN search. *PVLDB*, 13(5):643–655, 2020.

[76] E. Zhu, F. Nargesian, K. Q. Pu, and R. J. Miller. LSH ensemble: Internet-scale domain search. *PVLDB*, 9(12):1185–1196, 2016.