The Evolution of Vector Search: Data Engineering and ML Aspects

Yannis Papakonstantinou
*Google Cloud, yannispap@google.com

1 Introduction

Approximate Nearest Neighbor Search (ANNS) has been an active area of research for the last decades. More recently, the emergence of *Generative AI (GenAI)* and the emergence of powerful embedding models has led to a big surge of research activity. It also led to a large number of vector search offerings both from purpose-built vector databases as well as from databases and data warehouses that added vector and ANNS abilities.

While ANNS is not a new research problem, we believe that this area deserves further research for multiple reasons: The first (and obvious) reason is that whenever an area obtains a much higher importance it becomes worthy to dive deeper into its research and strive for perfection. ANNS is no exception to this. Thus, we appropriately see many research works that bring forth indexing improvements, quantization improvements and algorithmic improvements to improve the performance/recall Pareto of ANNS.

But we also believe that there are genuinely novel research needs in ANNS, beyond pure ANNS research, which emerged along with the recent surge. In this special issue of the Data Engineering Bulletin, we highlight the data engineering challenges in bringing vector search to databases (Section 2) and in expanding from pure ANNS to the larger problem of raising the quality of search(Section 3). We highlight also how ML can automate the solution to many of the data engineering problems we describe.

2 Vector Search meets Databases

Vector search is quickly becoming a staple of database applications, which use it for improved semantic search and in GenAI applications as an enabler of the *Retrieval Augmented Generation (RAG)* pattern. Similarly, in data warehouses it fuels analytics applications, such as classification, entity resolution and deduplication, search and anomaly detection in log analytics and other applications thanks to its powerful semantic matching ability.

This expansion brings myriads of database developers and data engineers to ANNS. But these developers are used to SQL's environment, which provides transactional consistency, declarative interfaces and physical-logical independence, which means that the query optimizer will figure out the best algorithm. Furthermore, we see that the new use cases of semantic search in databases typically involve SQL queries that perform ANNS along with filters on the structured attributes of the table that has the vectors and/or filters on attributes of tables that join with it.

2.1 Ease of Use

SQL developers, accustomed to SQL's declarativeness and physical-logical independence face an unfamiliar situation where they have to learn about their index choices, the esoteric parameters of each index and the respective index creation and search algorithms. The developers often need to choose between

different indexing methods, the most prominent being tree-quantization based indices and graph-based indices. Then they need to correctly configure their chosen index. Finally they also need to configure search parameters, which are particular to each algorithm. For example, configuring a graph-based index requires setting the memory space and (at least) two esoteric parameters, which are the number of edges per node and the number of candidates from which these edges will be chosen. Similarly for a tree-based index the user has to set the number of levels, vectors per leaf and fanout of the internal nodes.

What the developers truly care about is to meet the latency, queries per second (qps) and quality (recall) requirements of their application. The only way to meet them is by heavy experimentation with the index build parameters and the search parameters. Especially for SQL developers and users the manipulation of such parameters is a regression to pre-SQL times; during the last 3-4 decades SQL users are accustomed to just declaring indices and queries and expect the system to take care of the rest. Data engineering research is needed on effective, high level interfaces that receive the requirements and automate optimization. We believe that ML techniques will have to play a major role in automating optimization. As an alternate to complete automation, efficient and credible experimentation and tuning systems can allow database developers to semi-automate the process of achieving their performance goals.

Ease-of-use includes the capability of databases and data warehouses to automatically compute embeddings for the unstructured data and appropriately index them, so that the developers need not take care of the pipeline themselves. In the same spirit, ease-of-use also includes transactional consistency between the source data, their embeddings and the indices. Sufficient transactional consistency has been achieved by many recent releases that rely on the usual disk-based durability. Database developers want vector search to perform well from disk but also work very efficiently when there is plenty of memory. Database research is needed on how to combine the best of both worlds: Transactionally store the data in disk, yet achieve performance commensurate with a purpose-built main memory solution when the database instance has enough memory at its disposal, and the queries and associated datasets are non-trivial.

2.2 Challenges in queries that combine SQL filters, joins and vectors

Searches are often accompanied by structured data filters. Querying the structured data directly from the tables that they are found is far more convenient and manageable than the approach that is followed by purpose-built vector databases, where the structured data are organized into the vector index. The combination of filters, joins and vector-based ranking creates novel query execution and query optimization problems. The conventional method had been a choice between prefiltering (i.e., first evaluate the conditions and then do brute force evaluation of distances for the qualified rows/vectors) and postfiltering, i.e., first use the ANNS index to get a large number of candidates, then evaluate the conditions. We see significant promise in inline filtering techniques where the conditions are evaluated as the search procedure navigates the index. We believe there is important database research to be done on both the plan execution primitives and the optimization of their usage.

Works that create special index structures for accelerating filtered search queries have also recently emerged. We expect further advances in this area and the emergence of a plurality of approaches since there are multiple requirements, which will contradict each other: Some approaches will excel in particular types of filters while others will have broader scope of conditions. Some approaches will rely on "heavy" special vector index structures (eg, structures that weave the structured data in the vector index) and thus have the expected downsides on maintenance and memory footprint, while other approaches will work with minimal (or even no) modifications to the vector indices themselves.

3 Next Generation Search: Raising the Search Quality

Organizations want to enable higher result quality, which typically translates to recall, for their users' searches. This leads to the use of technologies beyond plain vector search and plain embeddings, whose inclusion in the search architecture is more intricate than a mere change of the embedding model. Consequently, these technologies invite respective research questions on the engineering of the search. Many of these technologies are already in widespread adoption and many more will likely be widely adopted soon. We overview a few of them next and present related data engineering research opportunities.

3.1 Hybrid Search

It is widely accepted that search achieves higher quality when semantic search (that is, vector search) is combined with classic text search. While vector search captures the semantics aspect of the search, classic text search solves the needle-in-the-haystack problem by ensuring that the data mentioning the specific requested keywords/terms are not missed. The currently prevalent methods of combining vector search and text search perform vector search in isolation from text search and vice versa. Then they combine the results of the text search and the results of the vector search afterwards. These methods are open to simplification and also to performance and quality optimization by intertwining the two types of search closer.

3.2 Semantic Search that trades off Quality, Performance and Simplicity

Diving deeper, we see that a stack of techniques emerges around retrieval and reranking, where as we go up the stack these techniques increase the accuracy (quality) of semantic search but also the latency/cost and/or the difficulty of deploying. We list characteristic techniques here in order of ascending quality.

1. Vector search over single embeddings remains the cheapest/fastest semantic retrieval method. Interestingly, there is not yet a benchmark to evaluate out-of-the-box solutions that would choose embedding models and autoconfigure vector search.

2. The fairly mature chunking and the associated crowding (during search) are simple techniques that slightly increase cost and latency by splitting long text data and thus having multiple vectors per doc.

3. Recently emerging supervised vector adaptation techniques (eg, [1] improve both quality and performance but require the customer to provide a small number of examples of desired results.

4. Multivector search (eg [2]) elevates quality by partially bringing in the benefits of cross-attention relevance ranking (#5) while keeping the efficiency benefits of computing embeddings and indexing them in advance. Data engineering challenges will appear in incorporating multivector retrieval in current vector systems (eg, see [3]). Furthermore, the increased number of options invites ML techniques for automating configuration and search optimization.

5. Cross-attention relevance (re-)rankers conceptually are functions relevance(Question, Data) that return a score for the relevance of the question to the data. Since a score based on cross attention cannot be precomputed, it is far more expensive than vector search. However, for their contribution in raising recall, they emerge as a common tool for reranking the results of vector search or hybrid search.

6. LLMs provide the highest flexibility as the user/developer can create a prompt that guides the LLM in how to rank. LLM-based algorithms may solicit relevance scores from the LLM for each to-be-ranked data item. But other algorithms are also possible that do not base ranking on relevance scoring; for example, feeding the question and the full set of to-be-ranked data items to the LLM prompt and asking it to spot the most relevant to the question.

Ultimately higher level abstractions are needed to give users the ability to achieve optimization of their goals on performance, cost and latency without having to learn the esoteric aspects of each technology. Until such time where optimization is sufficiently automated, efficient and easy-to-use experimentation techniques will be needed to semi-automate the optimization of combinations of these technologies.

4 Conclusion

Recent embedding models of high search quality as well as the interest in RAG have turned vector search into a focus for industry and research. While pure ANNS has a long history in the research community, we believe that we are in the early stages of the larger problem: Quality search that includes the structured data, textual indices and many techniques that improve semantic search. This research will entail both novel ML-based advances but also advances in ANNS and in the data engineering of end-to-end systems that use these techniques.

5 Bio

Yannis Papakonstantinou is a Distinguished Engineer, working on Query Processing and GenAI, at Google Cloud. Naturally vector search and, more broadly, powerful search is among the topics he works on. He is also an Adjunct Professor of Computer Science and Engineering at the University of California, San Diego, following many years of having been a UCSD regular faculty member. Previously he was an architect in query processing & ETL at Databricks. Earlier, he was a Senior Principal Scientist at Amazon Web Services from 2018-2021 and was a consultant for AWS since 2016. He was the CEO and Chief Scientist of Enosys Software, which built and commercialized an early Enterprise Information Integration platform for structured and semistructured data. The Enosys Software was OEM'd and sold under the BEA Liquid Data and BEA Aqualogic brand names and eventually acquired by BEA Systems. He has published over one hundred twenty research articles that have received over 20,000 citations. Yannis holds a Diploma of Electrical Engineering from the National Technical University of Athens, MS and Ph.D. in Computer Science from Stanford University (1997).

References

[1] Jinsung Yoon, Yanfei Chen, Sercan Aric and Thomas Pfister. Search-Adaptor: Embedding Customization for Information Retrieval. *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024.

[2] Omar Khattab and Matei Zaharia. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT *SIGIR*, 2020.

[3] Laxman Dhulipala, Majid Hadian, Rajesh Jayaram, Jason Lee and Vahab Mirrokni. MUVERA: Multi-Vector Retrieval via Fixed Dimensional Encodings https://arxiv.org/abs/2405.19504