LMDiskANN.jl: An Implementation of the Low Memory Disk Approximate Nearest Neighbors Search Algorithm

25 January 2025

Summary

LMDiskANN.jl is a Julia package that implements the Low Memory Disk Approximate Nearest Neighbor search algorithm (LM-DiskANN) [@pan2023lm], extending DiskANN-based methods [@jayaram2019diskann; @singh2021freshdiskann] for fast, accurate billion-point nearest neighbor search while significantly reducing in-memory usage. By leveraging memory-mapped files, a dynamic graph-based index, and tunable BFS expansions, **LMDiskANN.jl** enables large-scale similarity search on commodity hardware. This package integrates well with embedding-based workflows common in vector databases and modern machine learning pipelines. It also allows for insertions and deletions after the index has been constructed with operations to keep the index pruned of unnecessary connections in the graph.

Features include:

- Low-memory adjacency storage on disk using memory maps.
- Dynamic insert and delete operations on the graph index, adapting to changes in datasets.
- Configurable search expansions (`EF_SEARCH`, `EF_CONSTRUCTION`, etc.) to tune performance vs. recall.
- Optional user-key LevelDB mapping for flexible ID to and from key lookups.

By combining these capabilities, **LMDiskANN.jl** aims to reduce the memory footprint and overhead for large-scale nearest neighbor searches. It can be incorporated into any workflow that requires efficient embedding retrieval or similarity search for vectors in high-dimensional spaces.

Statement of need

Approximate Nearest Neighbor (ANN) search is a crucial component in domains such as recommendation systems, information retrieval, and representation learning (e.g., embeddings for natural language or computer vision). Traditional approaches can suffer from excessive memory usage and slow scaling when dealing with billions of points [@nene1997simple; @wang2021comprehensive]. By persisting adjacency structures on disk rather than in memory, **LMDiskANN.jl** addresses some of these bottlenecks, providing:

1. **Reduced Memory Overhead**: Only a minimal fraction of data needs to reside in RAM, making it feasible to handle larger datasets on modest machines.

2. **Dynamic Updates**: Graph-based insertions and deletions support real-time or streaming scenarios where data is continually changing.

3. **High Recall**: Tuning BFS expansions and adjacency degrees can yield high-quality nearest neighbor results.

4. **Insertions and Deletions**: Ability to insert and delete from a built index.
5. **Scalable Architecture**: Built on Julia's high-performance ecosystem, bridging native disk operations and advanced numeric libraries.

This approach benefits practitioners who need large-scale nearest neighbor indexing without specialized cluster infrastructures or extremely large memory capacities. The set up is made to have minimal requirements and has a simple installation proceedure. Using the package involves a few number of steps and examples are provided in the documentation.

State of the field

There are various open source ANN implementations and variants in other languages that are standalone or reside within different packages. Within the Julia ecosystem there are fewer options for users to choose from. The package SimilaritySearch.jl [@tellez2022similarity, @Tellez2022], offers the most mature codebase. Other notable options exist such as HNSW.jl and NearestNeighborDescent.jl. For applications involving massive datasets that exceed available RAM these options do not leverage disk space. These alternative do not offer the user the ability to insert and delete entries from an already constructed index which is essential for online framework integrations. This was the main motivation for [@singh2021freshdiskann] which describes the need for such utility.

Acknowledgements

Thanks to the Julia community for their continued support of open-source scientific computing. We also acknowledge the authors of LMDiskANN [@pan2023lm] and other works it is based on, [@jayaram2019diskann; @singh2021freshdiskann] for foundational ideas in disk-based graph indexing.

References