Cairo University
Faculty of Engineering
CMP 4030

Advanced Database

Semantic Search Engine

Phase #1

Team Members:

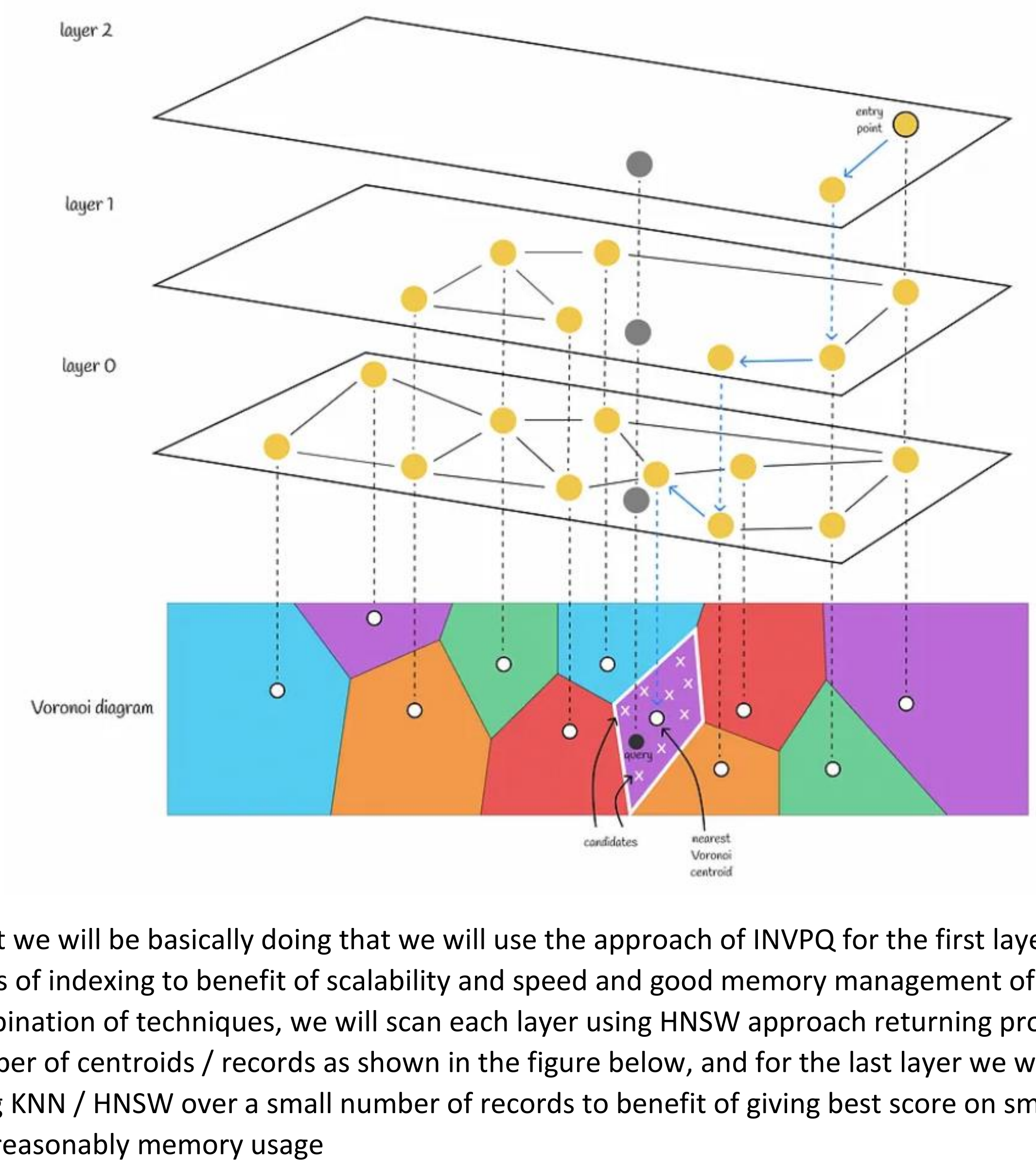
Name	Sec	BN	Code
Ahmed Hany Farouk	1	10	9202213
Mohab Zaghloul	2	28	9203568
Abdelrahman Fathi	2	2	9202846
Nour Ziad Almulhem	2	30	9204005

Presented to: Eng. Abdelrahman Kaseb

After investigating many search algorithms, we would like to provide what we found about each technique and the approach to implement our semantic search engine.

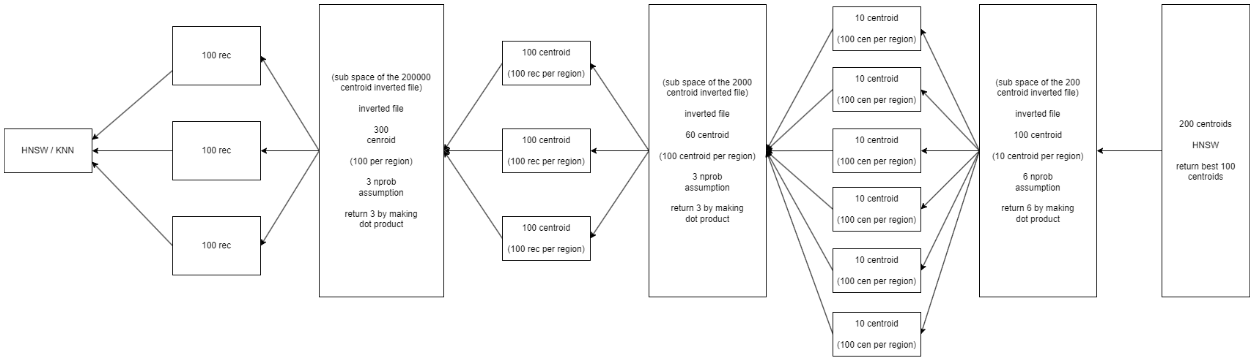
1. Our approach

HNSW can be combined with other methods like an inverted file index and product quantization to enhance performance. It plays the role of a coarse quantizer for IndexIVFPQ meaning that it will be responsible for finding the nearest Voronoi partition, so the search scope can be reduced.



What we will be basically doing that we will use the approach of INVPQ for the first layers or levels of indexing to benefit of scalability and speed and good memory management of this combination of techniques, we will scan each layer using HNSW approach returning proper number of centroids / records as shown in the figure below, and for the last layer we will be using KNN / HNSW over a small number of records to benefit of giving best score on small data and reasonably memory usage

The parameters of each level of this diagram are changeable according to the fine tuning we will do to achieve proper score and proper speed.



In this specified scenario, an initial search operation utilizes the Hierarchical Navigable Small World (HNSW) algorithm across a set of 200 centroids, each comprising 10 sub-centroids. The objective is to identify the top 100 centroids based on their relevance. Subsequently, a dot product computation is employed to ascertain the six nearest centroids, the exact number will be tuned later.

Proceeding to the subsequent indexing level, attention is directed towards the exploration of the three, the exact number will be tuned later, nearest centroids from a reduced pool of 60 centroids derived from the prior level.

Advancing to the subsequent indexing tier, a similar process is undertaken to identify the three, tunable, nearest centroids from a broader set of 300 centroids derived from the preceding level.

Conclusively, the final indexing stage involves a K-nearest neighbors (KNN) or HNSW search methodology within a dataset consisting of 300 records.

2. Inverted file indexing mechanism

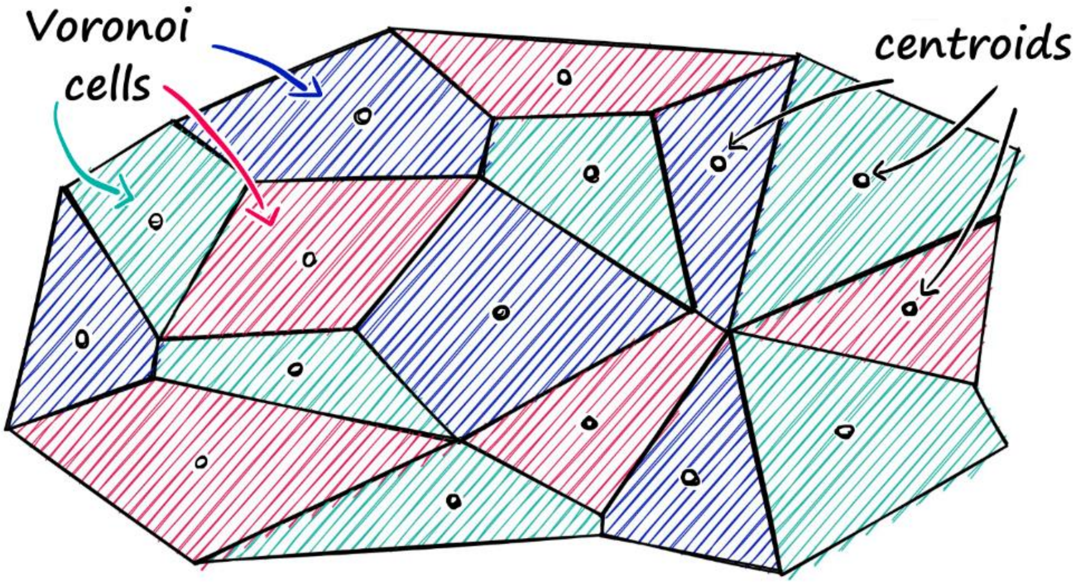
An Inverted File Index, also known as an inverted index or postings list, is a database index that maps content, like words or numbers, to their locations in a table, document, or set of documents. This index plays a crucial role in semantic search, reducing the search scope of database vectors when performing queries.

When a new object is introduced, distances to all Voronoi centroids are calculated. The centroid with the lowest distance is chosen, and vectors in that partition become candidates. This approach significantly reduces the search scope, making the process much faster than traditional methods of searching through all dataset vectors.

This introduces the Edge problem, which means this technique does not guarantee that the found object will always be the nearest, especially when the queried object is near the border of another region. To mitigate this, the search scope can be increased by choosing several regions based on the top **M** closest centroids to the object, reducing errors in such edge cases.

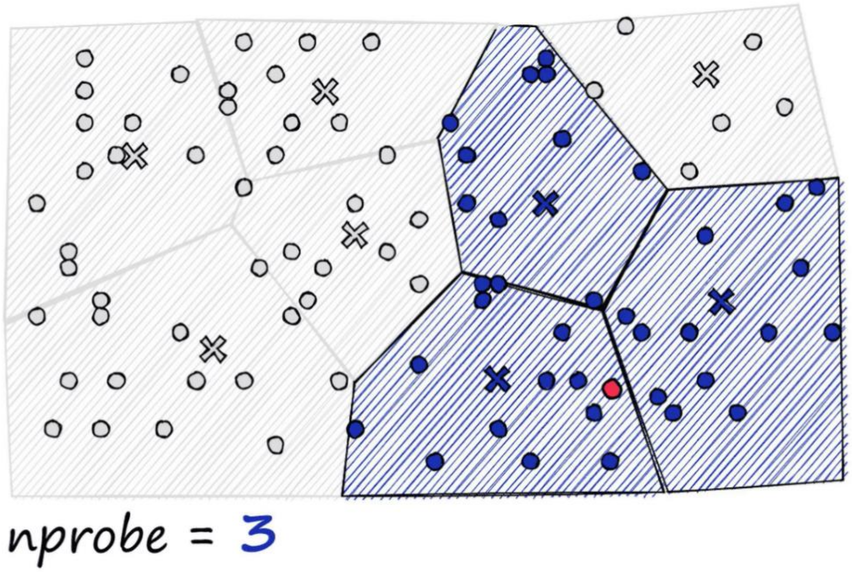
Referring to: https://medium.com/towards-data-science/similarity-search-knn-inverted-file-index-7cab80cc0e79

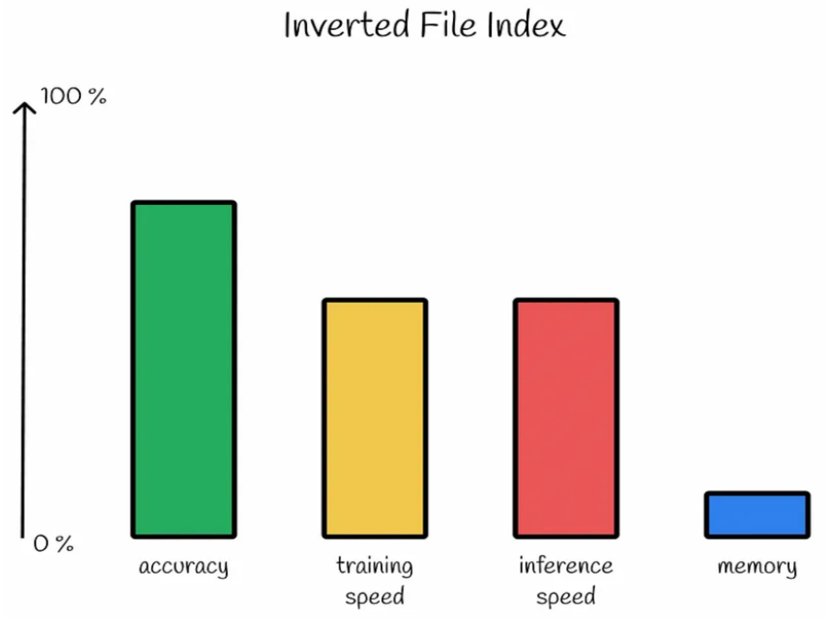
https://www.pinecone.io/learn/series/faiss/vector-indexes/



And to overcome the edge problem we decided to tune the parameter of **Nprobe** so we could achieve better results considering the trade-off we will face between the accuracy (score) and time, as a bigger **Nprobe** means less speed but more reasonable results.

Instead of searching in the region of the nearest centroid we will be searching on many near regions depending on the number of **Nprobe.**





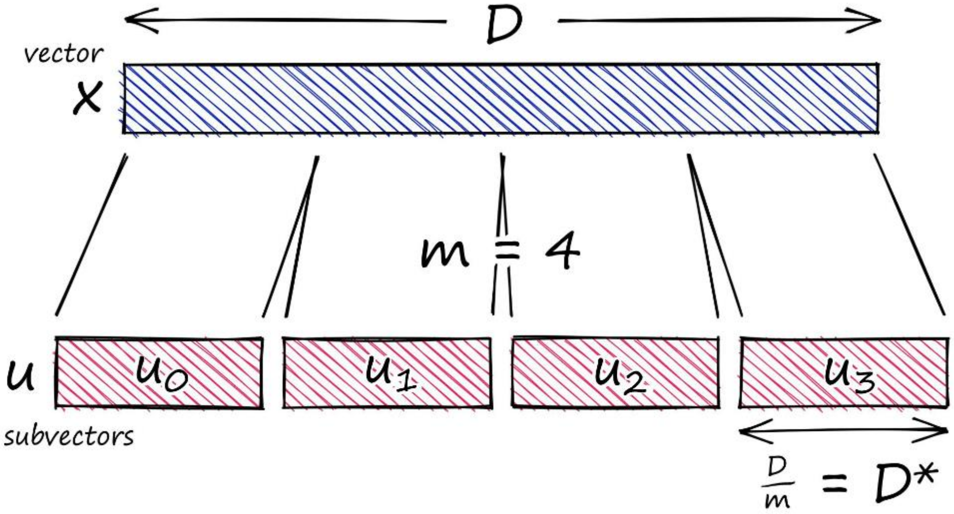
3. Product quantization

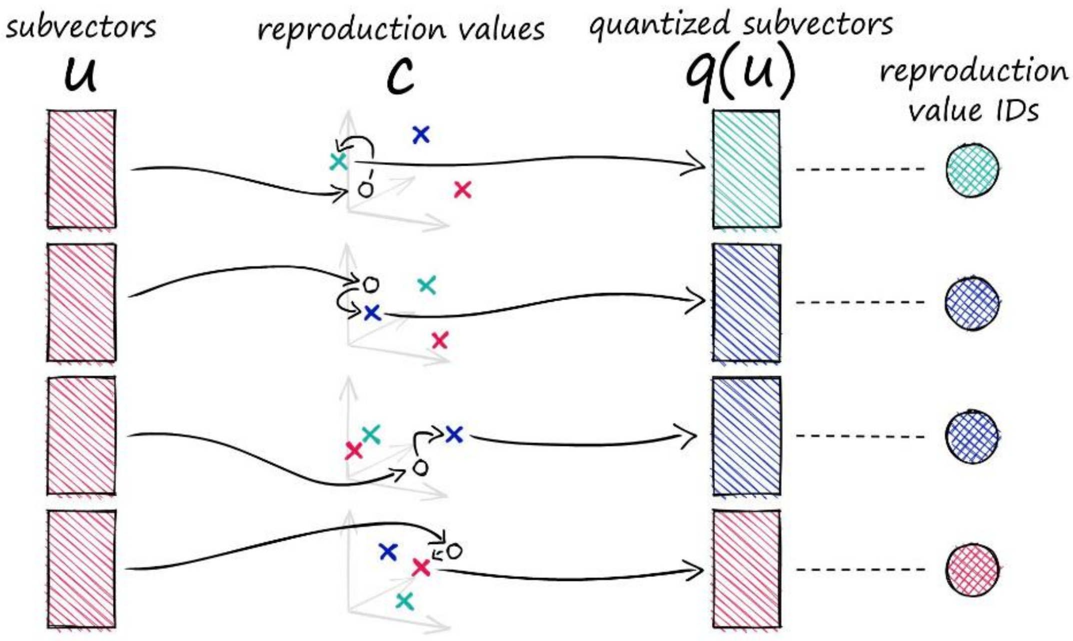
Product quantization is a process that transforms dataset vectors into memory-efficient representations called PQ codes. It is a lossy-compression method, sacrificing some prediction accuracy for significant memory efficiency, making it particularly effective in practice.

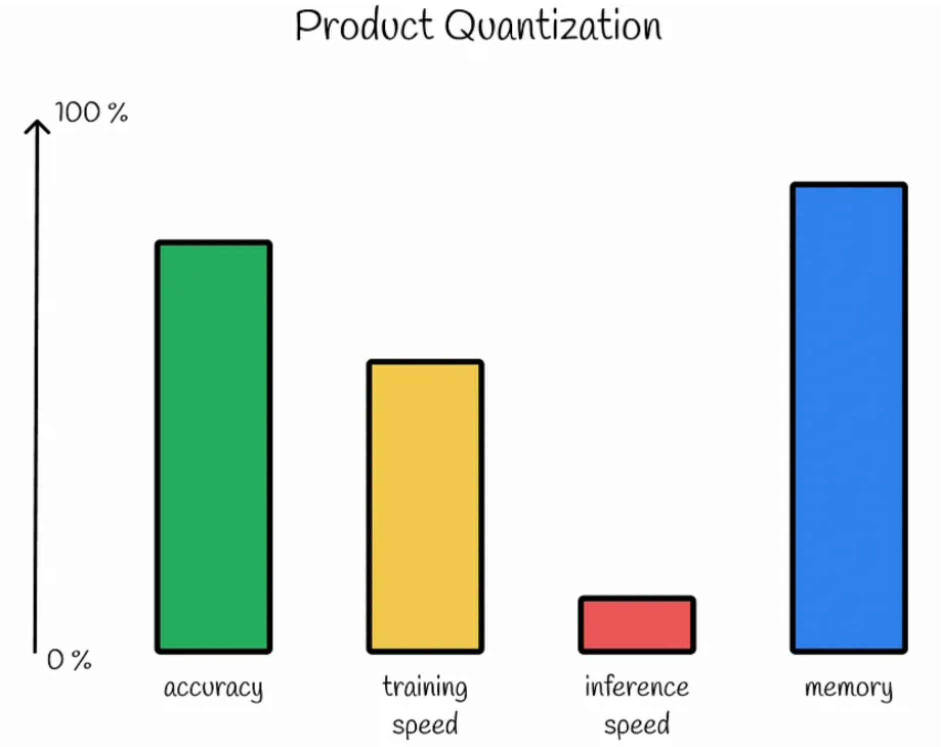
The algorithm divides each vector into sub vectors, processes each subspace independently, and applies a clustering algorithm to create centroids. These centroids encode sub vectors, forming a codebook. The encoding involves assigning cluster IDs to sub vectors, resulting in a compressed representation of the original vector. The memory required for an encoded vector is proportional to the number of sub vectors and the logarithm of the number of clusters.

How it works:

- A high-dimensional vector is divided into equally sized sub vectors, creating distinct subspaces, the number of sub vectors will be tuned to make sure we achieve reasonable results.
- Each sub vector is represented by its position identifier (j).
- Clustering algorithms are applied to each subspace independently, creating centroids.
- Each subspace has its set of clusters, forming a codebook.
- The number of clusters (k) is usually chosen as a power of two for memory efficiency.
- Each sub vector is assigned to a centroid, referred to as a reproduction value.
- Reproduction values are created for each subspace, forming an overall list of reproduction values.
- Sub vectors are replaced with specific centroid vectors.
- Unique IDs are assigned to each centroid vector, creating a compact representation.







Referring to : https://towardsdatascience.com/similarity-search-product-quantization-b2a1a6397701

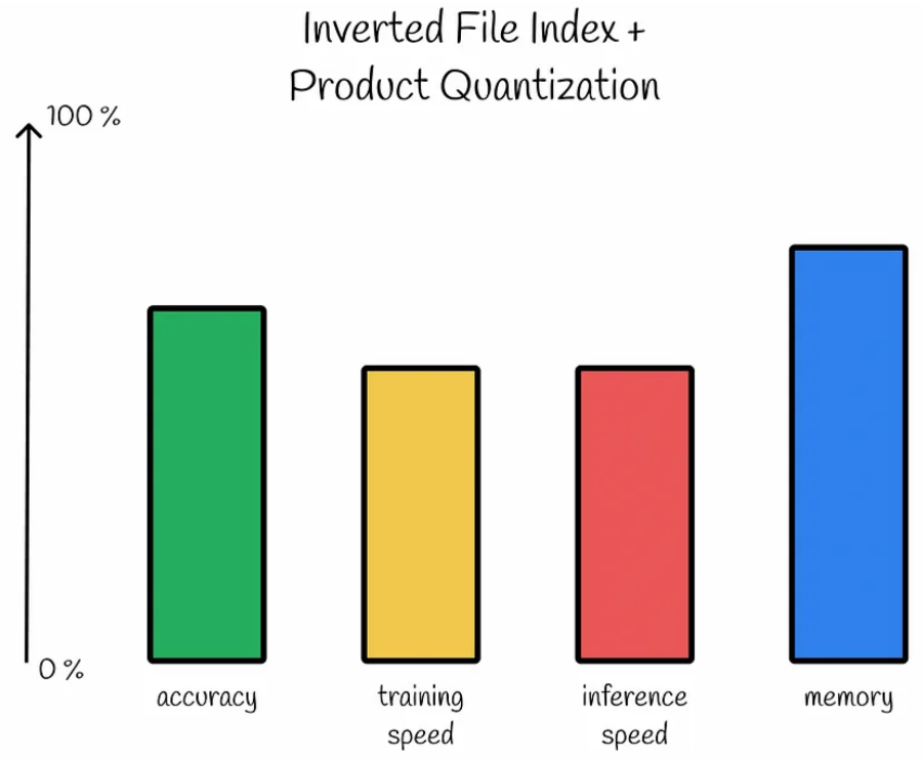
https://www.pinecone.io/learn/series/faiss/product-quantization/

- **Combining INV and PQ**

to create a new algorithm that is both fast and memory efficient. The ideas discussed in the article are primarily based on a specific paper.

The key concept introduced is that of residual vectors. After executing a clustering algorithm to create clusters with centroids, the residual of a point is defined as the offset of that point from its centroid. This involves subtracting the mean of the cluster from each point, effectively centering the points around zero. Importantly, the relative positions of vectors to each other remain unchanged when replaced with their residuals.

The performance of this combined algorithm leverages the strengths of both INV and PQ. The trade-off involves the need to compute and store distance matrices for each Voronoi partition, but the overall advantages make it a favorable solution.



Referring to:

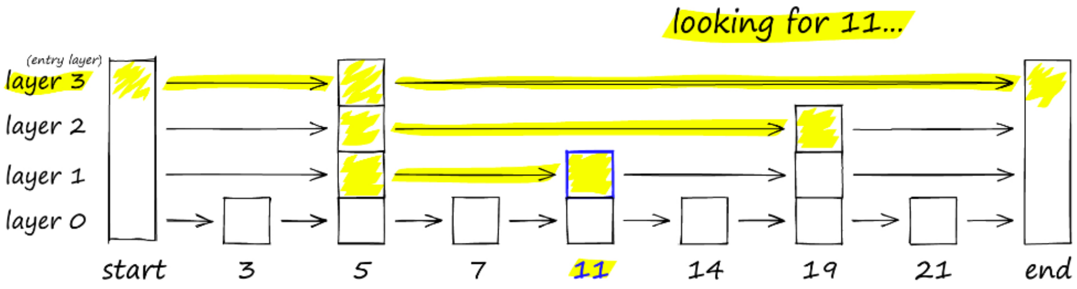
https://towardsdatascience.com/similarity-search-blending-inverted-file-index-and-product-quantization-a8e508c765fa?gi=0a13a076ae6d

4. HNSW

Hierarchical Navigable Small World (HNSW) is an innovative algorithm that excels in approximate nearest neighbors (ANN) searches, offering rapid search speeds and exceptional recall. While it might seem complex, breaking it down unveils two key techniques: the probability skips list and navigable small world graphs.

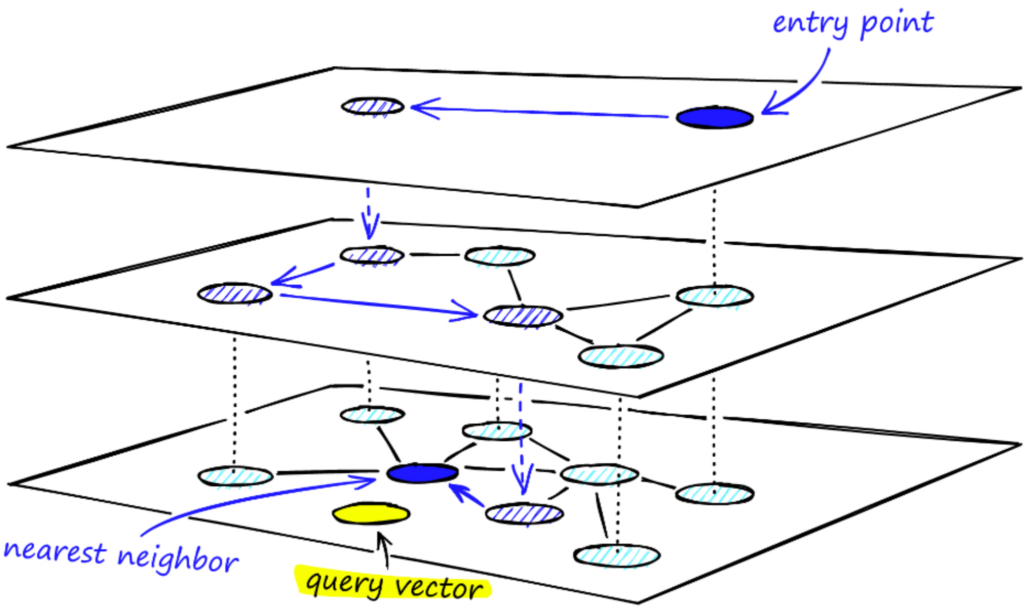
Probability Skip List:

This technique, introduced by William Pugh in 1990, facilitates fast searches akin to a sorted array. It combines a linked list structure for easy insertion with layers of linked lists. Starting at the top layer, searches move along edges, dropping to the next layer if the current key exceeds the target. HNSW adopts this layered format, employing longer edges in higher layers for fast search and shorter edges in lower layers for accurate search.



Navigable Small World Graphs:

Vector search using Navigable Small World (NSW) graphs, introduced from 2011-14, involves creating a graph where each vertex connects to several others, known as friends. The search starts at a designated entry point, moves to nearby vertices, and repeats the process by greedily traversing to connected vertices until finding a local minimum.



The search process through the multi-layer structure of an HNSW graph.

HNSW, evolving from NSW, introduces a multi-layered graph structure with longer links in the top layers and shorter, more numerous links in the lower layers. During a search, it begins in the top layer, gradually moving down through layers until reaching the local minimum in layer zero. The graph construction involves iteratively inserting vectors, with the number of layers represented by parameter L.

Choosing the maximum layer (l) for each node involves a random assignment with an exponentially decaying probability distribution. The insertion process includes finding the nearest node at each layer and, on reaching layer l, searching for efConstruction nearest neighbors, choosing M of them, and connecting edges. The authors suggest setting mL, the level multiplier, to 1/ln(M) for optimal performance.

Key Construction Parameters:

- M: Values between 5 and 48 are recommended, with smaller values suitable for lower recalls or low-dimensional data, and higher values for high recalls or high-dimensional data.
- efConstruction: Choose a value that results in recall close to 0.95–1 during training.

- M_max: Optimal value close to two * M to avoid performance degradation and excessive memory usage.

The insertion complexity is O(logn), and the overall construction complexity is O (n * logn).

Referring to: https://towardsdatascience.com/similarity-search-part-4-hierarchical-navigable-small-world-hnsw-2aad4fe87d37