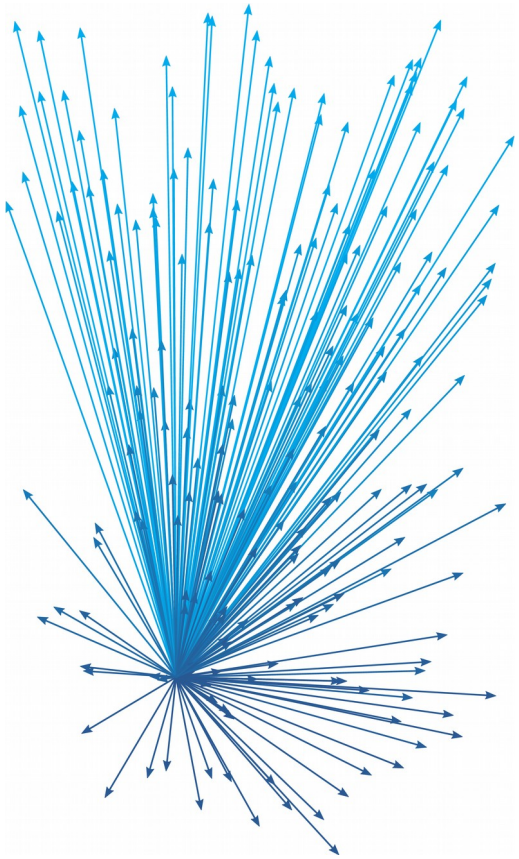
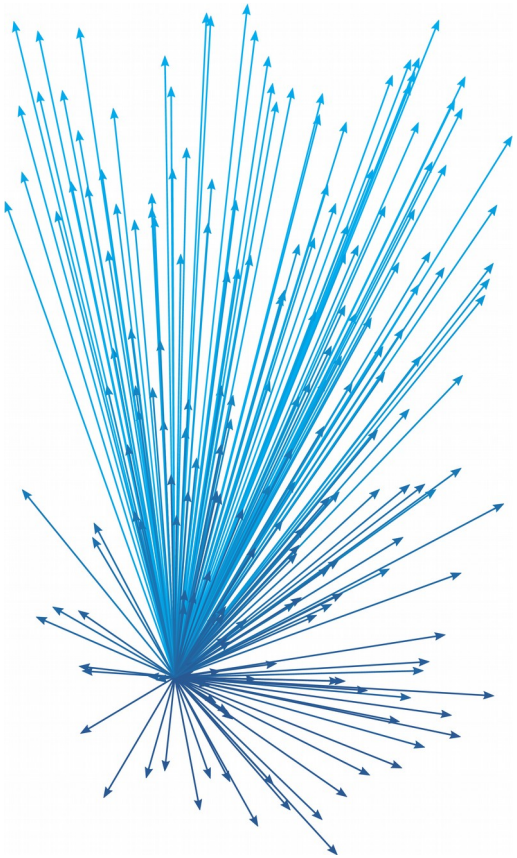
High-Dimensional Nearest Neighbor Search



High-Dimensional Nearest Neighbor Search

- **Who?**
  - About Cliqz and me
- **What?**
  - Problem statement
- **Why?**
  - Applications
- **How?**
  - Exact solutions in low dimensions
  - Approximate solutions in high dimensions

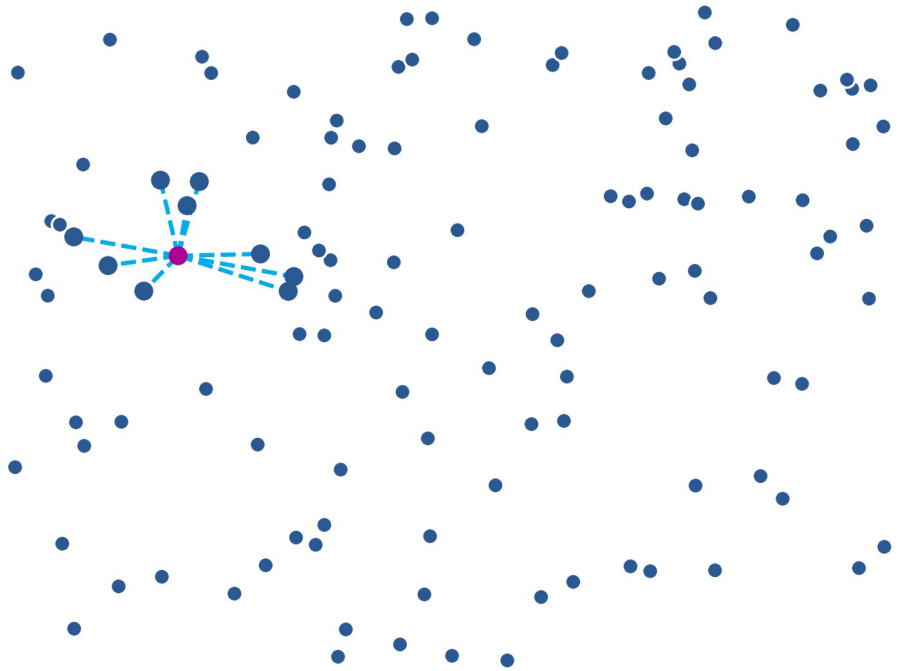


Who? – Cliqz and Me

- **Cliqz**
  - Builds privacy-focused browsers
  - Manages its own search index
- **Me**
  - Erik Larsson
  - Software engineer
  - Search backend
  - Almost 2 years at Cliqz

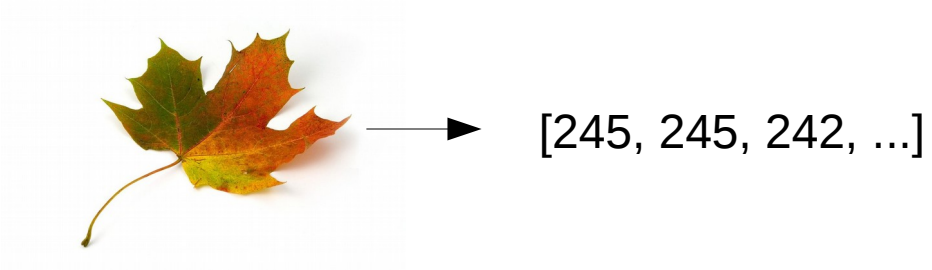
What? – Problem Statement

- **Data (D):**
  - Many vectors (millions or billions)
- **Input (Q):**
  - One query vector (not necessarily from **D**)
- **Output:**
  - The *k* vectors from **D** that are closest to **Q**



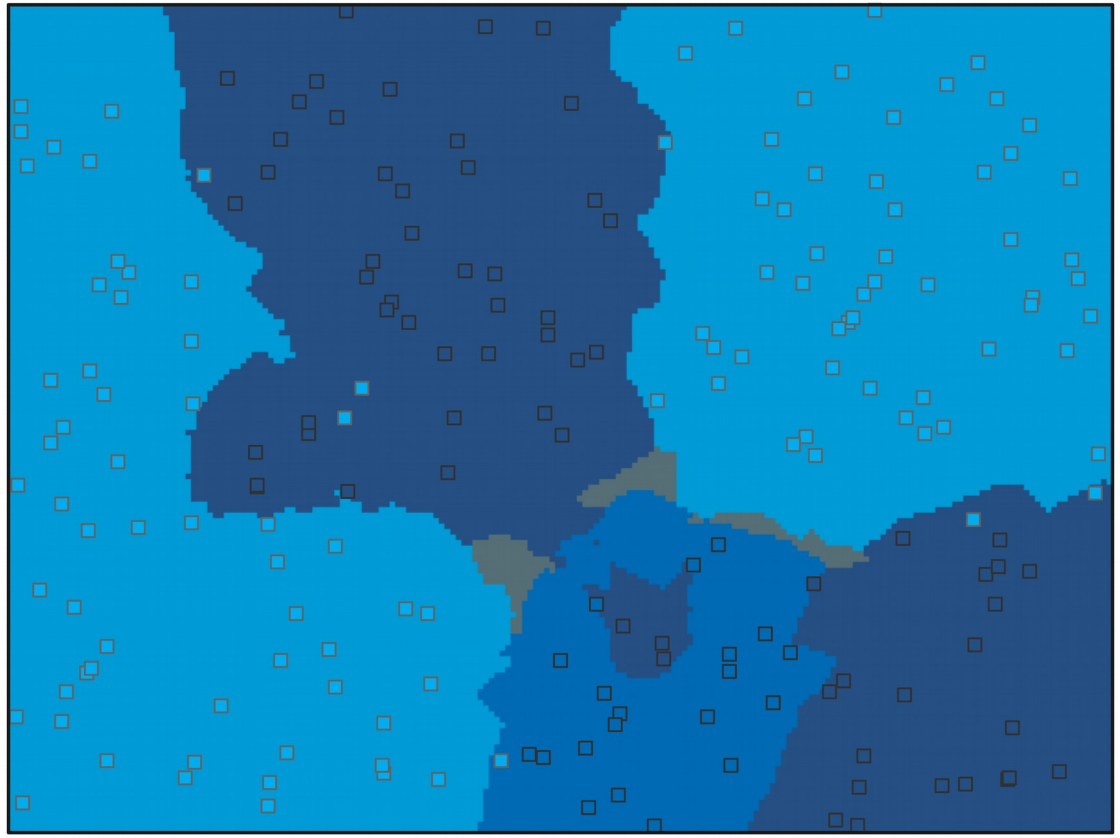
Why? – Applications

- **Reverse image search**
  - Represent image by a vector
  - Pixel values arranged in a vector
  - More advanced features (SIFT, SURF, ORB)
  - Similar vectors ↔ similar images



Why? – Applications

- **kNN classification**
  - Input data with known labels
  - Represent input objects by vectors
  - Assign new unseen object the label of its *k* nearest neighbors
  - Regression
- **Fast and simple baseline**

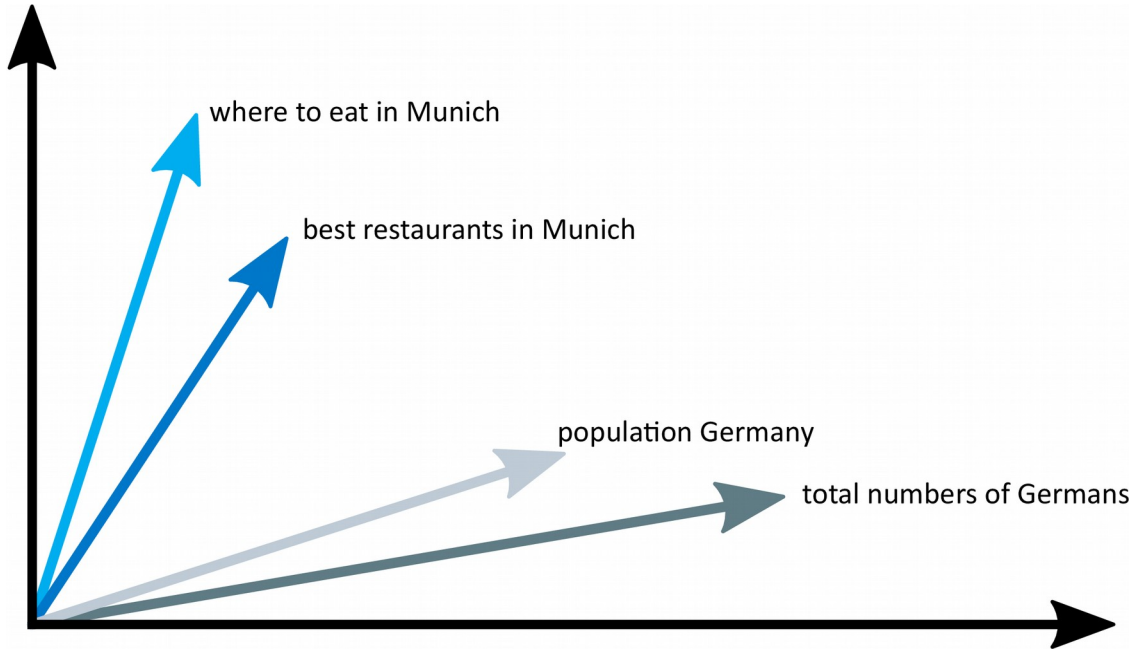


Why? – Applications

- **Plant classifier**
  - Map images of plants to vectors
  - Do a NN lookup with an unknown query image
  - Assign label of closest vector(s)

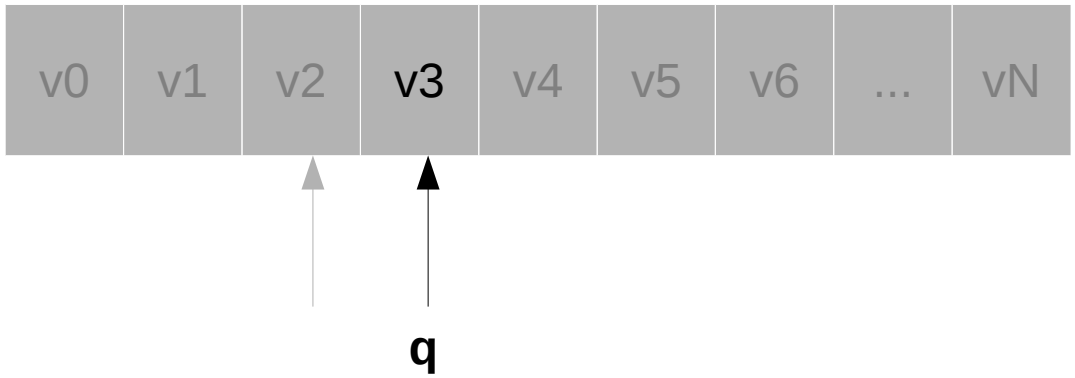
Why? – Applications

- **Similar queries at Cliqz**
  - Answer new, unknown queries by considering similar, known queries
  - Queries with different phrasing but similar meaning
  - Map query to vector (word2vec, tf-idf vectors)
  - NN-lookup
  - Map back to queries



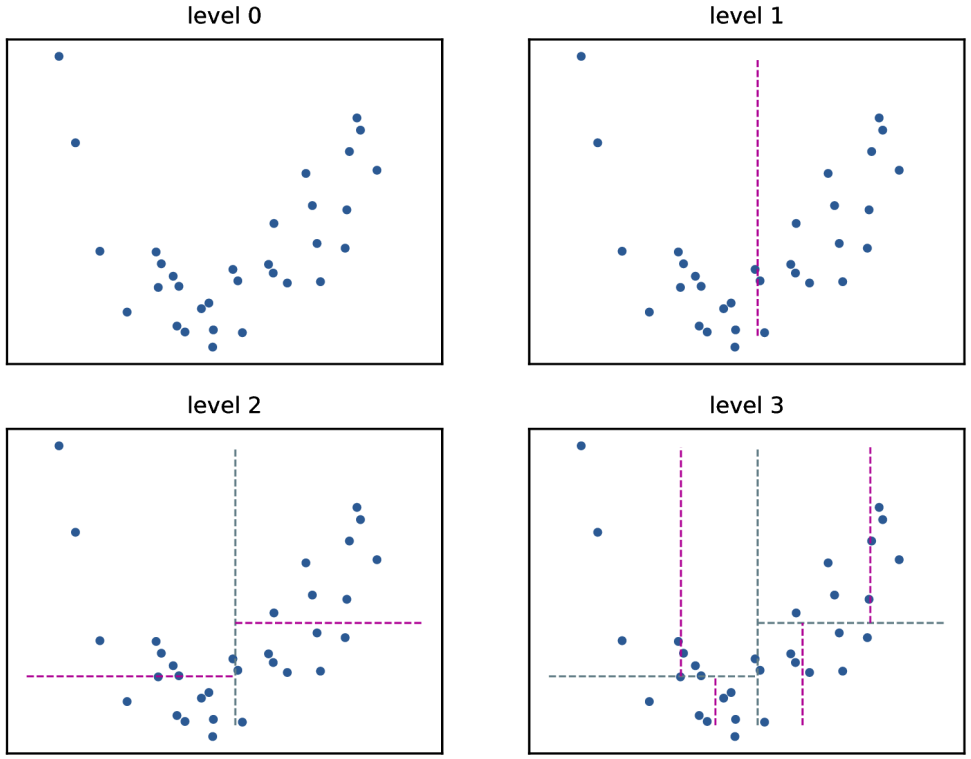
How? – Exact Solutions

- **Linear scan**
  - Conceptually easy
  - No extra space for index
  - Slow
- **Spatial partitioning**
  - Divide space into disjoint subsets
  - Divide and conquer



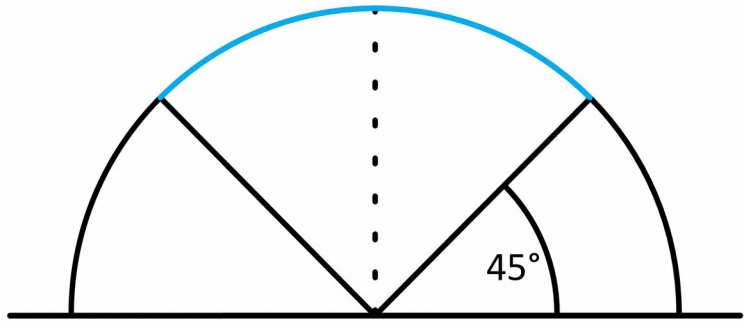
How? – Spatial Partitioning

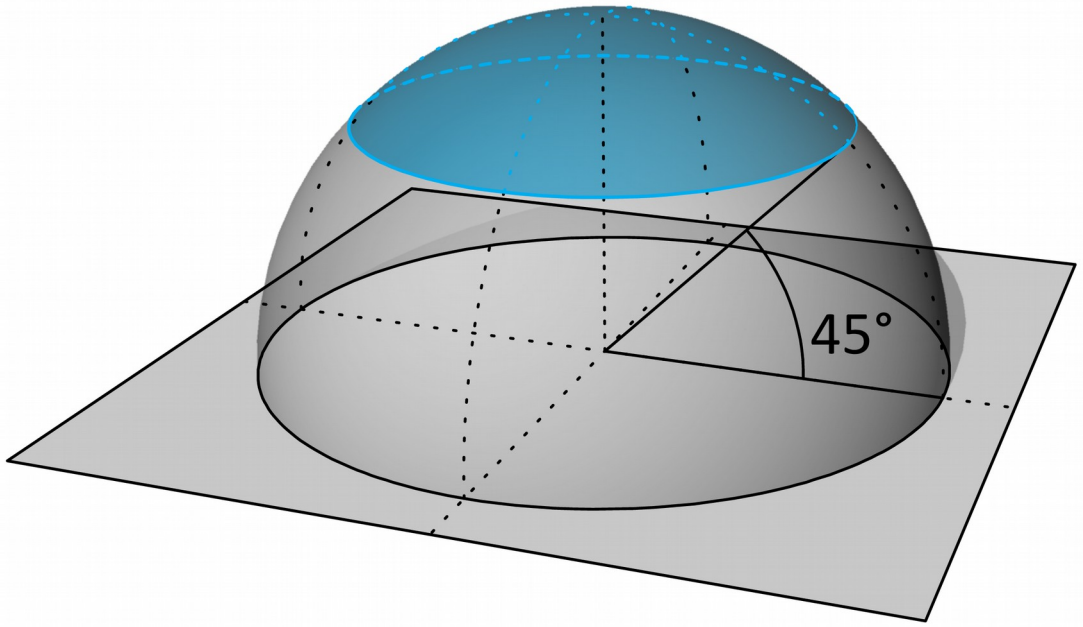
- **Kd-tree**
  - Binary tree
  - Each node splits the space with half of the vectors on each side
  - Search by traversing tree from root down to leaf
- **Ball tree**
  - Similar to Kd-tree
  - Cover space with “balls” containing all points within a specific radius



How? – High-Dimensional Vectors

- **100-1000 dimensions**
- **Curse of dimensionality**
  - Many methods scale poorly as the dimension increases
  - Considering one coordinate at a time is no longer enough
- **Splitting random data with a plane**
  - In 2d/3d most vectors end up reasonably far away from the plane
  - In 100d most vectors end up pretty close to the plane





How? – High-Dimensional Vectors

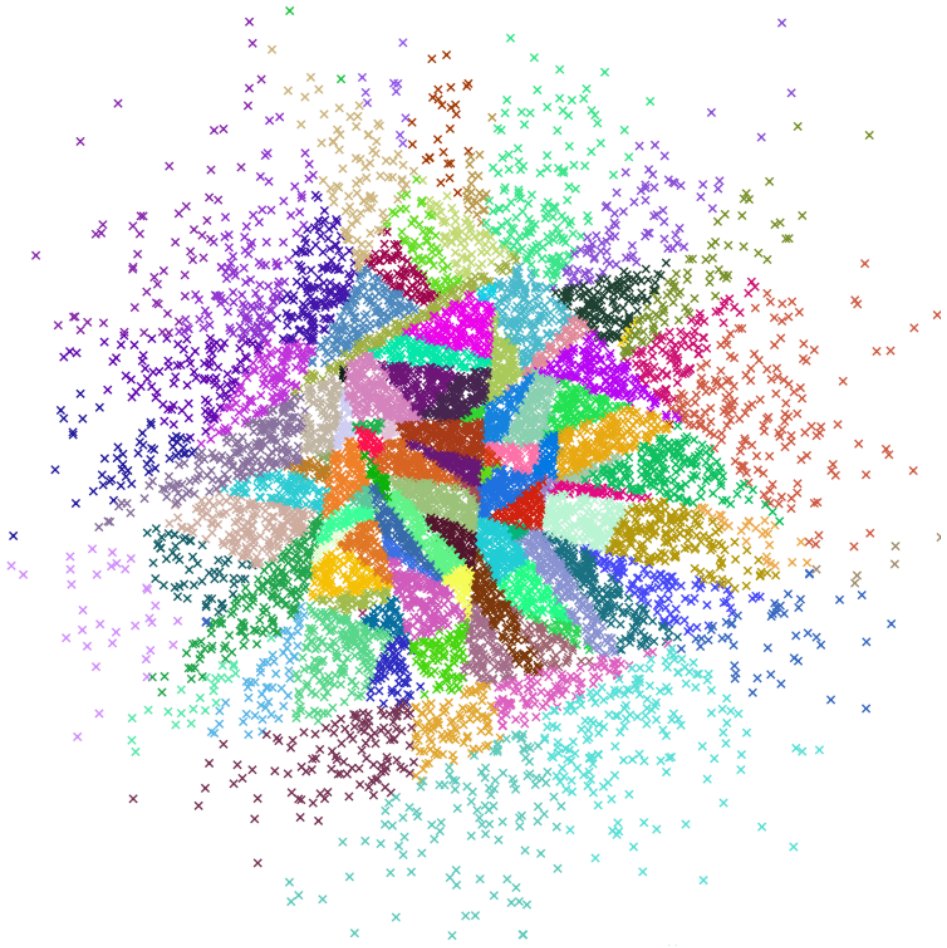
- **Ways forward**
  - Same algorithms, slower
  - Something more clever/complicated
  - Make the problem simpler

How? – High-Dimensional Vectors

- **Ways forward**
  - Same algorithms, slower
  - Something more clever/complicated
  - **Make the problem simpler**
- **Return vectors that are pretty close**

How? – Approximate Solutions

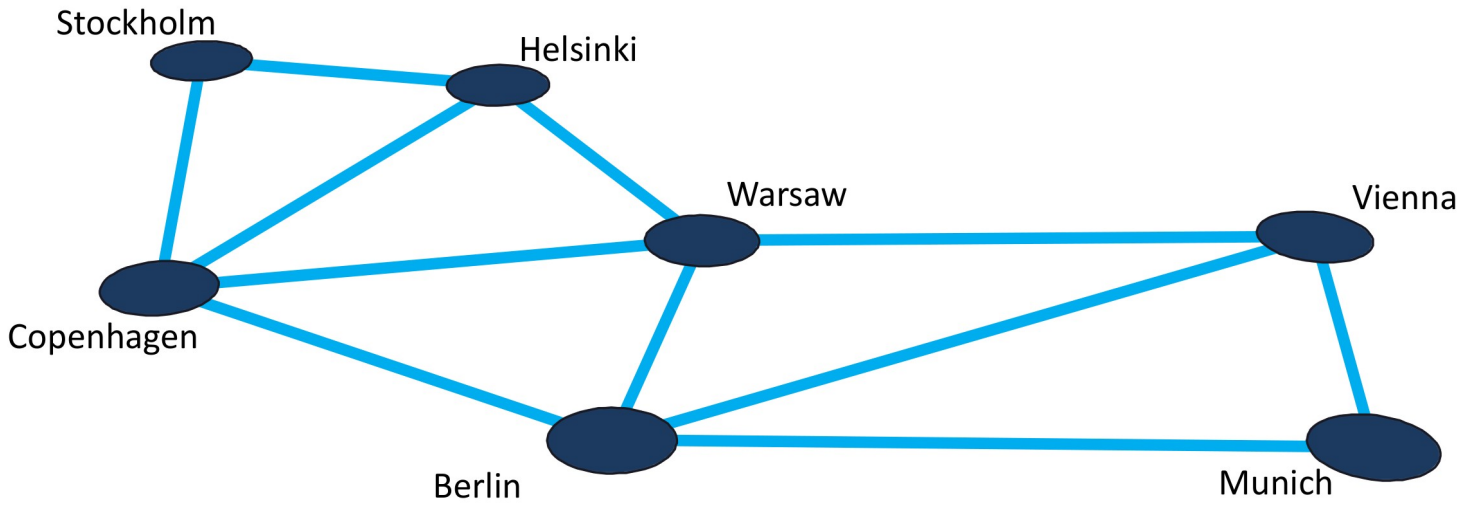
- **Annoy – A**pproximate **n**earest **n**eighbors **o**h **y**eah
  - A forest of kd-trees with non-axis-aligned splitting planes
  - Search in all trees simultaneously
  - Search parameter decides how many nodes are visited
  - Nice UI (C++ with python bindings)
  - Used by Spotify for music recommendations
  - Previously used at Cliqz for similar queries
  - https://github.com/spotify/annoy



https://github.com/spotify/annoy

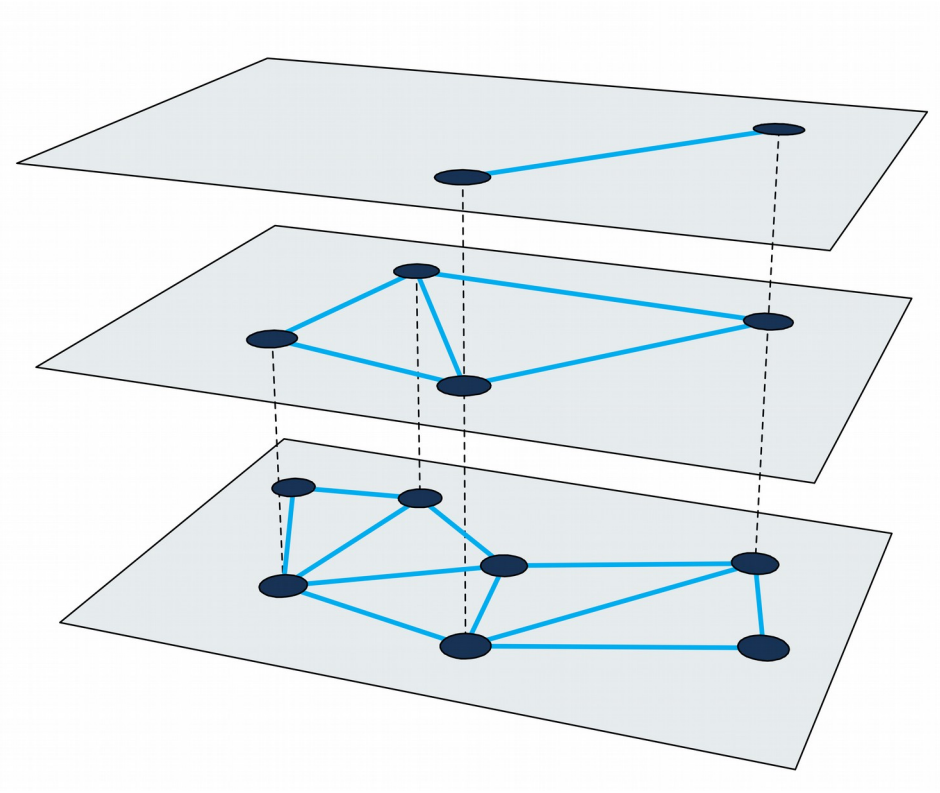
How? – Approximate Solutions

- Proximity graph



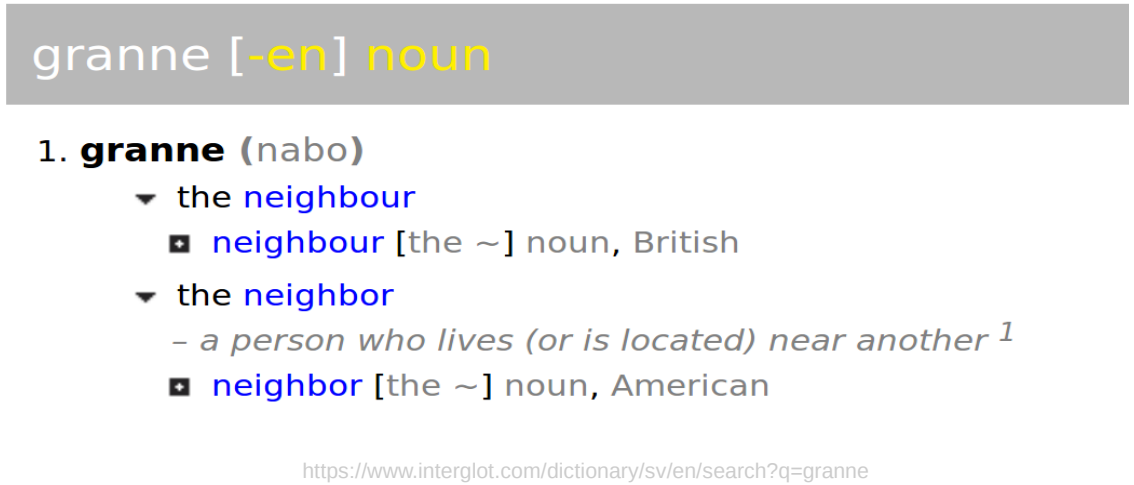
How? – Approximate Solutions

- **HNSW – H**ierarchical **N**avigable**-S**mall **W**orld
  - Graph-based: layers of proximity graphs (similar to skip list)
  - Greedy search in each layer
  - Elements inserted one by one by searching in so far constructed index
  - Yu. A. Malkov and D. A. Yashunin: *Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs*



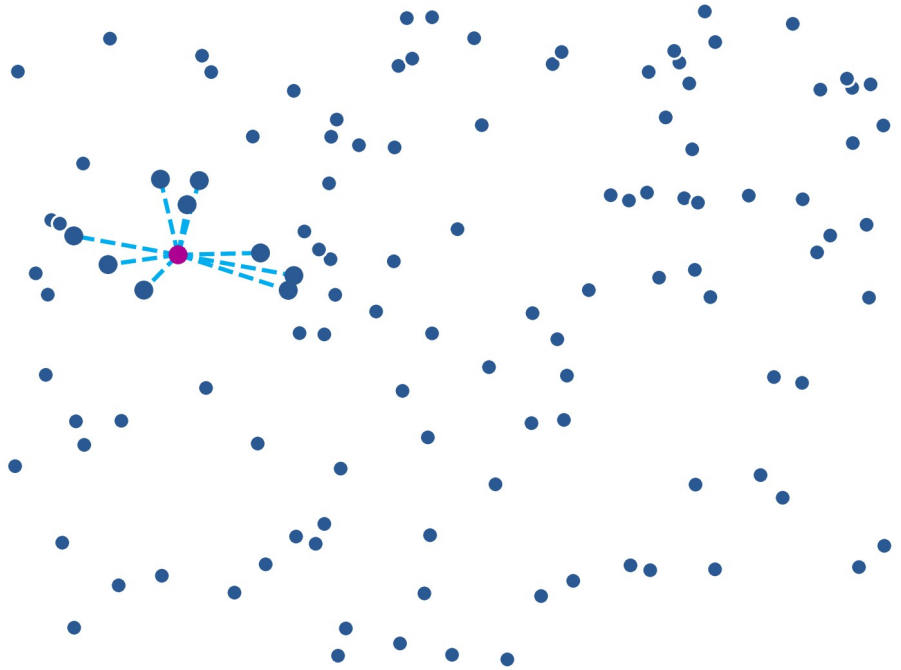
How? – Approximate Solutions

- **granne – g**raph-based **r**etrieval of **a**pproximate **n**earest **ne**ighbors
  - Based on HNSW
  - Optimized index construction
  - Hybrid RAM/disk usage
  - Index billions of vectors
  - Rust with python bindings
  - Used in the Cliqz search backend to serve similar queries
  - https://github.com/herrerik/granne



Recapitulation

- **The (Approximate) Nearest Neighbor Problem has many interesting applications.**
- **A few fundamentally different methods**
- **Best methods depends on dimensionality, data size and structure**



High-Dimensional Nearest Neighbor Search