Agenda:

HNSW

ACORN

RAG (intro)

HNSW = "hierarchical navigable" small world

1. Build index / proximity graph

   - vectors = nodes

   - edges should connect "nearby" vectors

   core q: how do we pick edges?

2. [For inference / search] Traverse index to find NN

   - use greedy search from pre-defined entry point

   * most of work is in 1 - build a good graph ahead of time! Then search is relatively simple.

INTUITION for HNSW

- goal: fast search

- for this - any two nodes should be connected by a SHORT path

* we need to quickly get to "neighborhood

of query" - this path will be short if:

any two nodes in graph is connected by

a short path

INTUITION FOR "SMALL WORLDS"

General graph properties:

N = # of vertices (# of base vectors, or dataset size)

L = "characteristic path length" (global property

→ avg path length btwn **ANY** 2 vertices

we want this to be small!

C = "clustering coefficient" (local property)

→ cliquishness of any neighborhood

→ consider v

→ $N(v)$ is v's neighborhood

↳ v, and all nodes connected to it

$$C_v = \frac{\text{ratio of \# of edges in } N(v)}{\text{\# of edges } N(v) \text{ could have}}$$

→ numerator: $N(v)$ has some edges

(minimum would be $|N(v)| - 1$, because

at minimum v connects to all other

nodes, by our definition of $N(v)$

→ denom: if $N(v)$ is fully connected

it has $\frac{1}{2}(|N(v)|-1)(|N(v)|)$ edges -

each node connects to all other

nodes: 1/2 is for how we have

an undirected graph

→ so: $C_v = \dfrac{\#\text{ of edges in } N(v)}{1/2(|N(v)|)(|N(v)|-1)}$

→ C is sum of this over all nodes:

$$C = \frac{1}{|V|} \sum_{v \in V} C_v$$

→ WHAT IS A SMALL WORLD? Watts and Strogatz, 1998

SMALL: → L (diameter of graph) grows logarithmically

with N = SMALL world

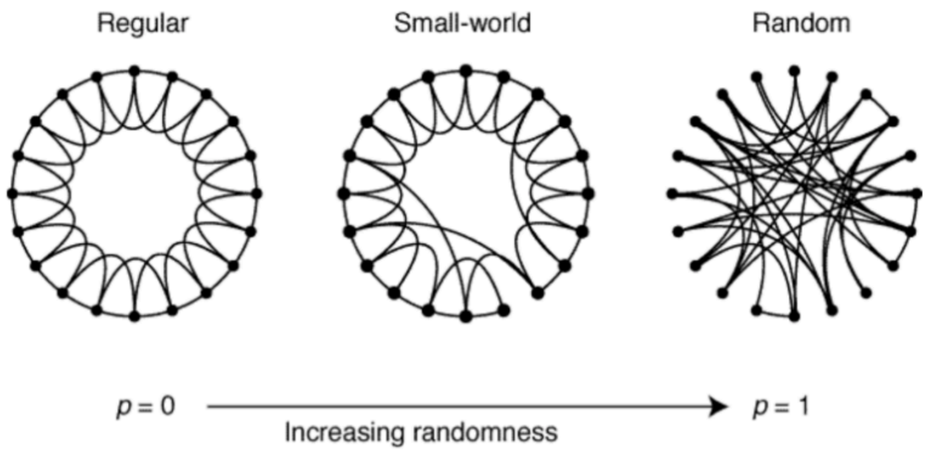
- social network: many (billions) of people
- avg distance btwn any 2 random

people ≈ 6

LARGE: L (diameter of graph) grows LINEARLY

w/ N

- consider 2 extremes:
  - regular: highly clustered, large world
  - random: poorly clustered, small world



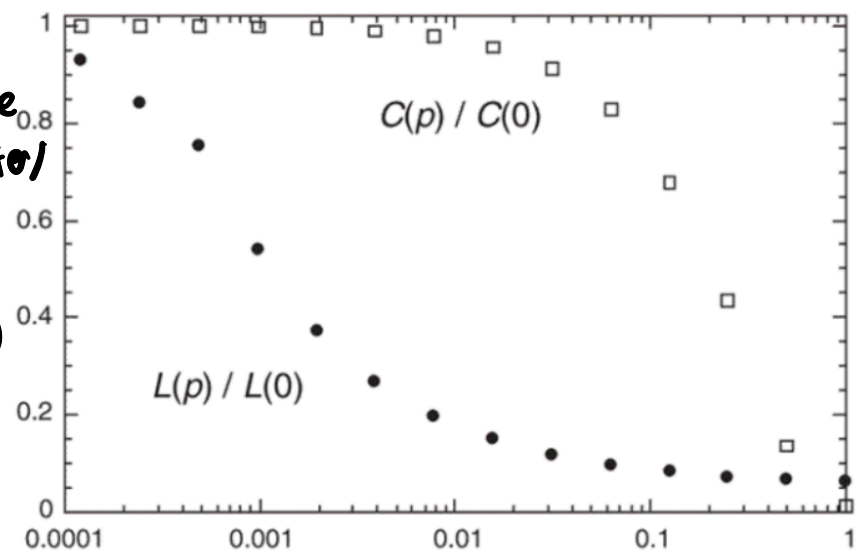
(prob of forming random edge)

* there's something in the middle of regular and random:
  - you can get "small world" by adding a few random edges
  - and more clustered too

Watts and Strogatz, 1998

Figure 2: Characteristic path length $L(p)$ and clustering coefficient $C(p)$ for the family of randomly rewired graphs described in Fig. 1.

measure diameter / L (path length)



(increase prob. of random edges)

→ measure L and C as we add random edges

→ as we move to right, path length quickly drops off (even at low probabilities!)

→ but clustering coefficient remains high

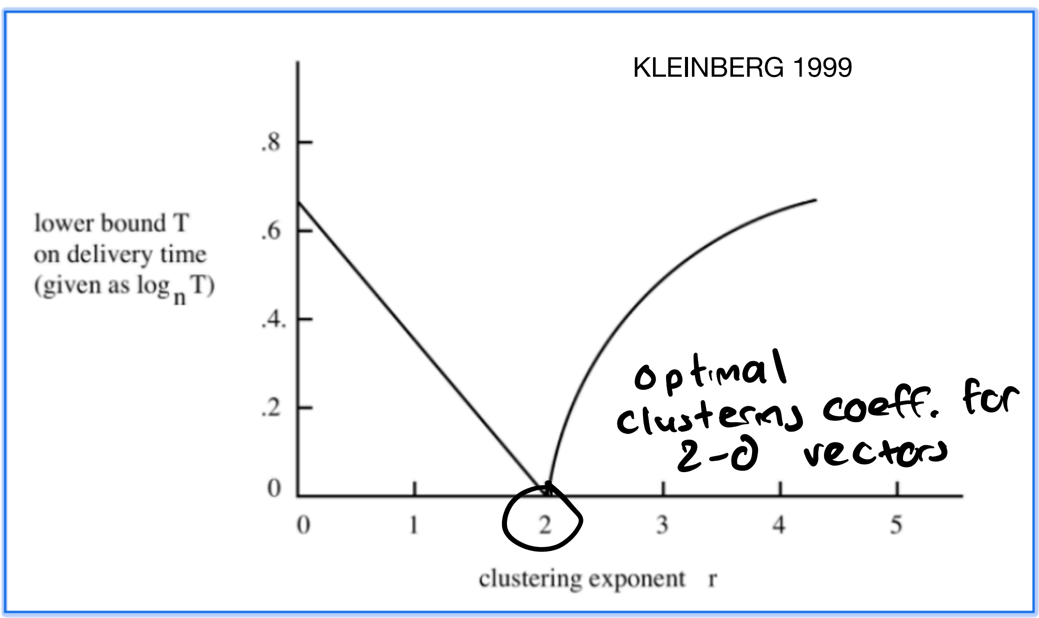
BACK TO HNSW - how do we build the graph index??

- we want small L / small world - any two nodes should be connected by a short path

* BUT: short path should be easily found - as we're doing GREEDY search (may not search all neighbors)

How do we create NAVIGABLE SMALL WORLDS?

- not all worlds are "navigable"
  - navigable = greedy (decentralized) approaches can FIND short path length
  - intuition: we need some degree of clustering



How do we SEARCH over a navigable small world: 2-phases:

1. Zoom-out
   - start from entry point (or periphery) - could have low degree, traverse to high-degree node ("hub")
2. Zoom-in:
   - start from hub (high degree), traverse to ans - low degree

* unfortunately, zoom-out is subject to a local minima

so instead
1) Any two nodes should be connected by short path (have random edges)
2) short paths should be "easily found" by greedy search (have clustering)
3) start search from "hub"
4) Fix degree of node - limits amt of search

OK FINALLY - HNSW

- key idea 1: separate edges according

to length scale

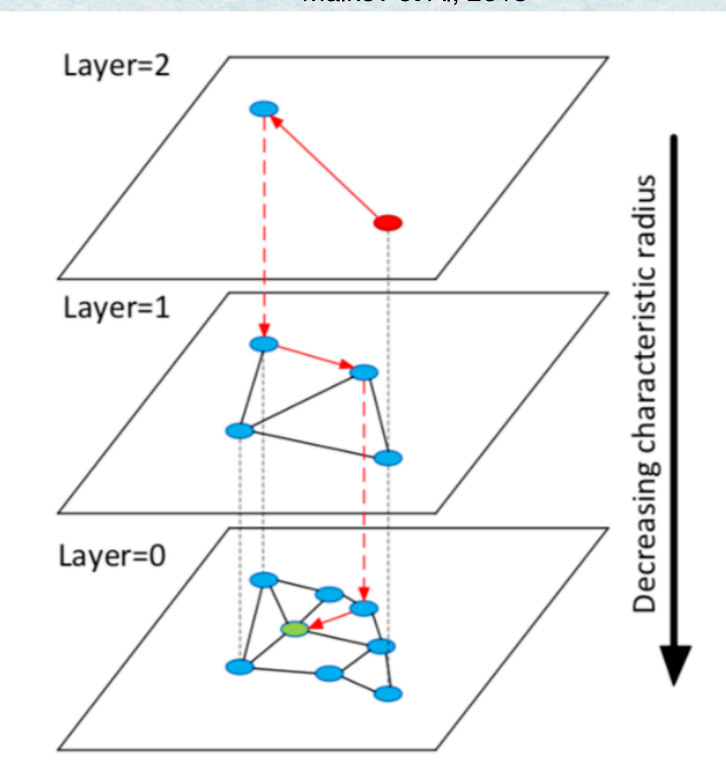
(consider "length") as # of hops btwn two nodes

- start search from longest range edge → long range jumps

- as we go down graph - we have more LOCAL structure - edges are "shorter"

- key idea 2: bound degree of each node - makes search tractable

Malkov et Al, 2018



Multilayer, hierarchical graph

- upper level = longest range links

- hierarchical: upper levels sparser

- lowest level: All nodes

- when building, as we go up, sample at fixed probability from below

- bounded degree: each node has at most M neighbors

* search time = roughly O(log N) where N = # leafs

-steps per level, etc = constant

How do we construct this graph?

- Index Construction
- Iteratively add vectors to partially constructed multilayer graph
- For each vector v:
  1) Choose an integer level l stochastically
     - Level probabilities decay exponentially
     - Leads to logarithmic scaling of # layers
  2) Iterate from the top layer to level l
     - Perform greedy search
     - Chosen node is entrypoint to level below
  3) Iterate from level l to level 0
     - Perform greedy search
     - Select M (constant) nearest neighbors to become edges of v

ANN search alg:

ANN Search Algorithm

1) Begin search from pre-defined entrypoint at top level
2) Iterate from the top layer to level l
   - Perform greedy search
   - Chose a single node as the entrypoint to level below
3) Search level 0
   - Perform greedy search
   - Choose K nodes to return

Sources: Liana Patel's cs229s (Stanford) Lecture on Knowledge Intensive LLM Systems: https://docs.google.com/presentation/d/16ZkmYEmiCKEbiAd3vrgsLBVpT4lQD6Aj0jx_FHZLWRQ/edit#slide=id.p