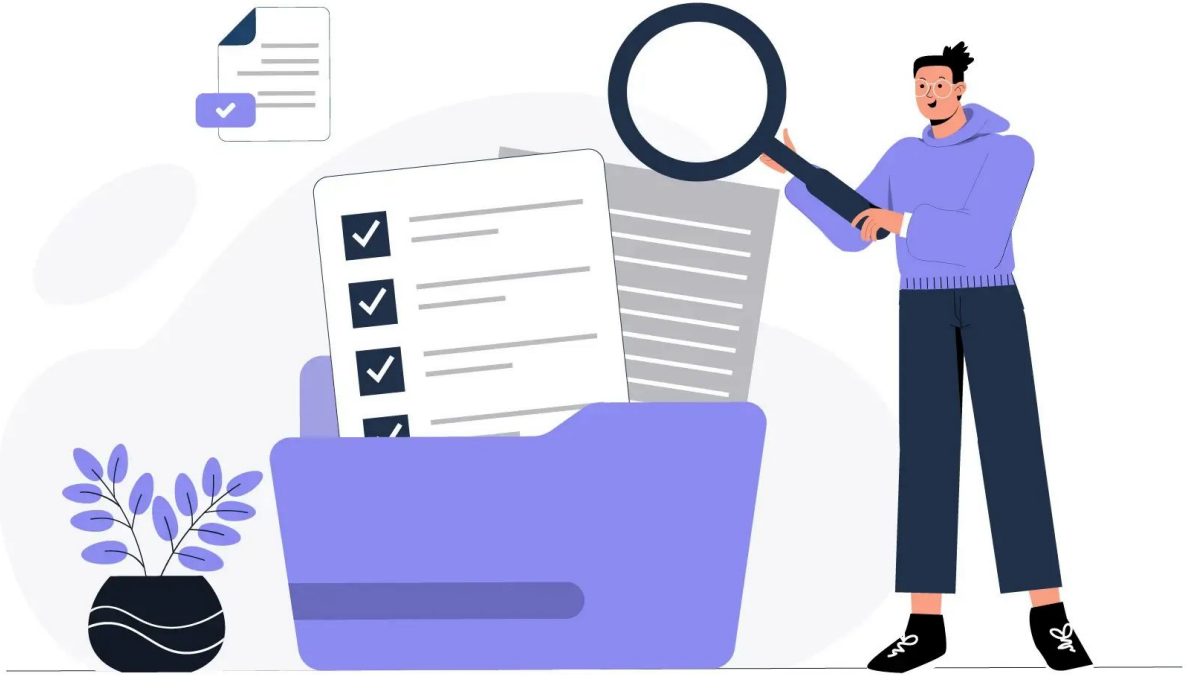
Векторный поиск

Елена Кантонистова

Далее в программе

1. Какие задачи будем решать?
2. Точный поиск по словам и по эмбеддингам
3. HNSW
4. FAISS

Какие задачи будем решать?



Виды поиска

1. Точный поиск по словам
2. Точный поиск по эмбеддингам
3. Приближенный поиск по эмбеддингам

1. Точный поиск по словам: BM25 - способ посчитать релевантность запроса q документу D

Для каждого токена запроса qi мы можем посчитать степень его релевантности документу D - это BM25(qi):

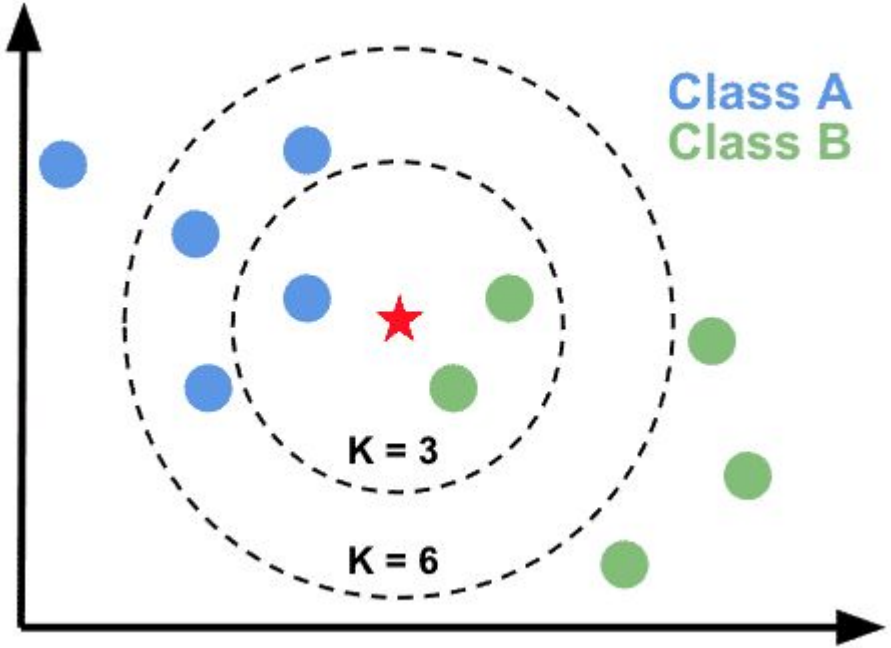
$$\sum_{i}^{n} IDF(q_i) \frac{f(q_i, D) * (k1 + 1)}{f(q_i, D) + k1 * (1 - b + b * \frac{fieldLen}{avgFieldLen})}$$

where:

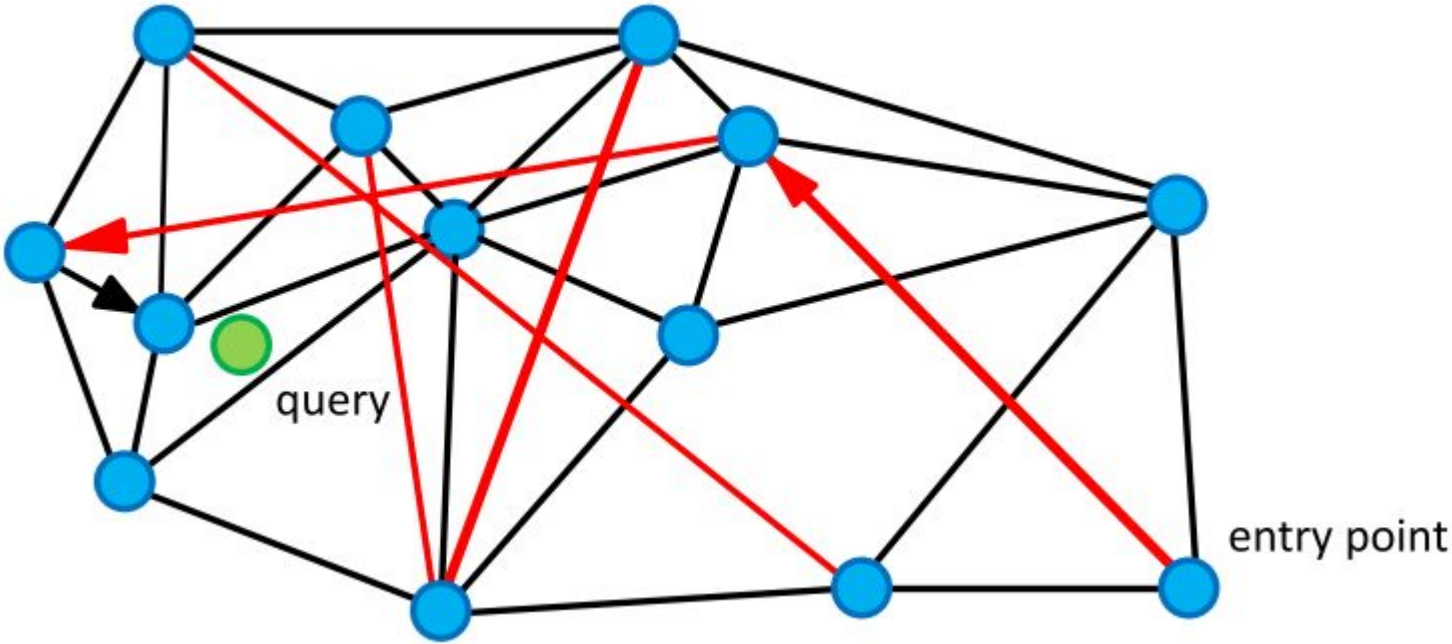
- qi is a query term,
- $f(qi,D)$ is qi's term frequency in the document D,
- D is the length of the document,
- avgdl is the average document length in the text collection,
- $k1$ and b are free parameters, usually chosen empirically (common values are $k1$=2.0 and b=0.75),
- IDF(qi) is the IDF for qi.

Далее суммируем эти релевантности по всем токенам qi запроса q.

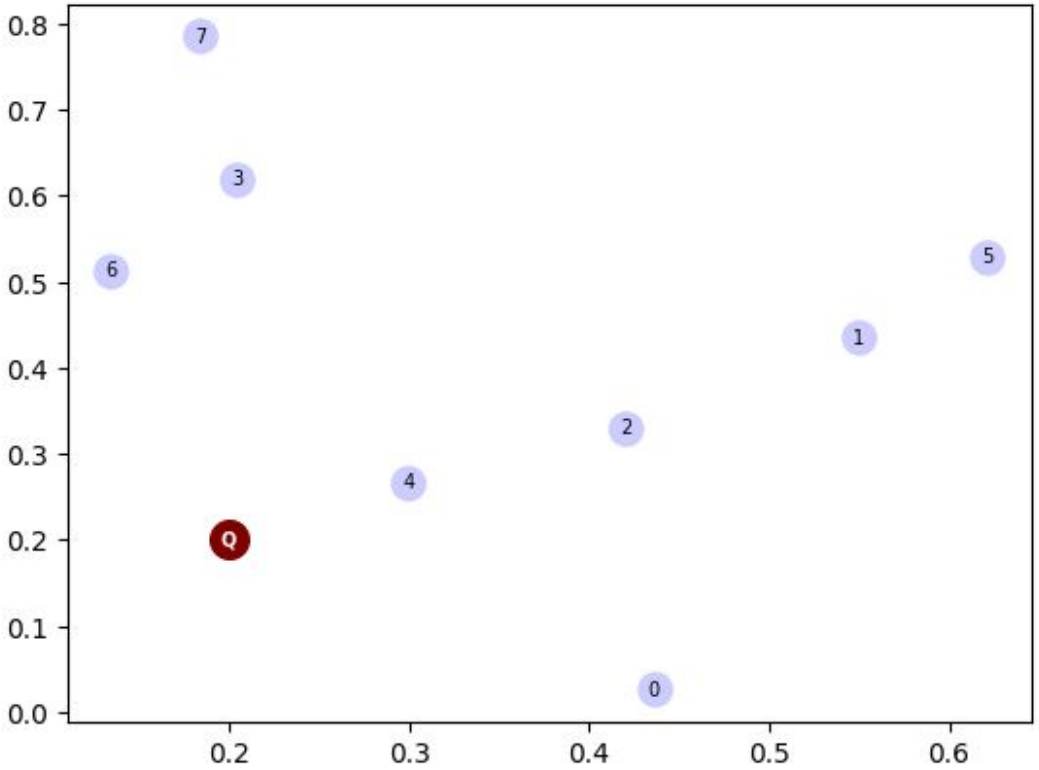
2. Точный поиск соседей (на векторах слов): KNN



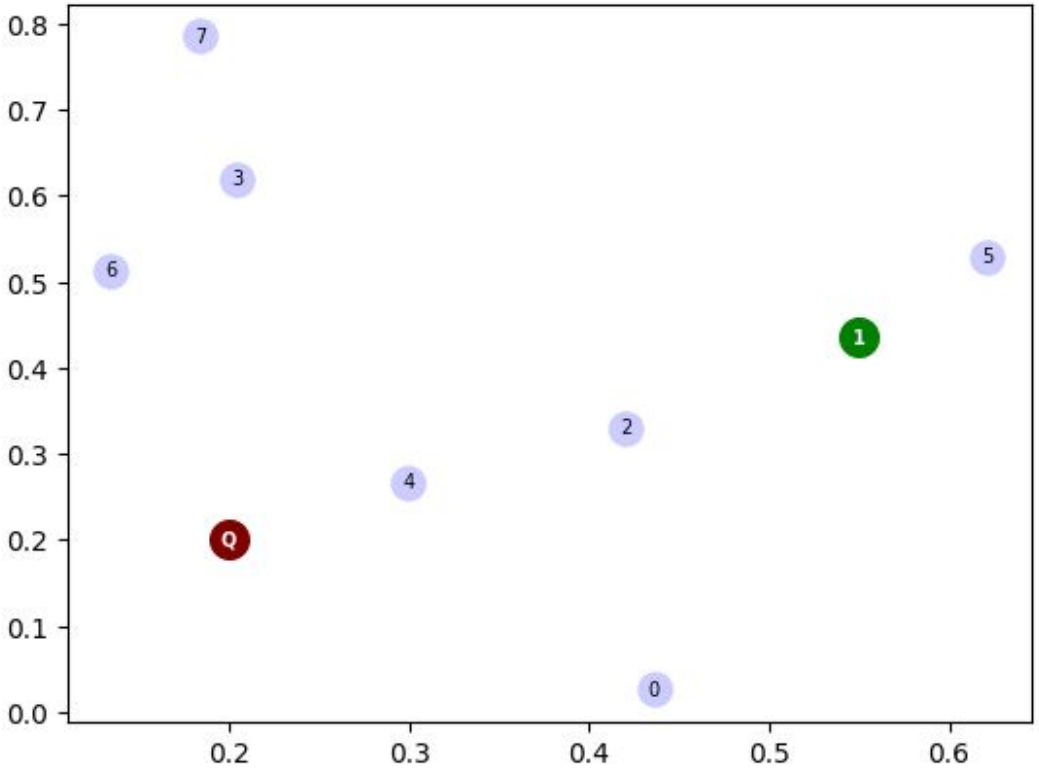
3. Approximate NN: NSW -> HNSW



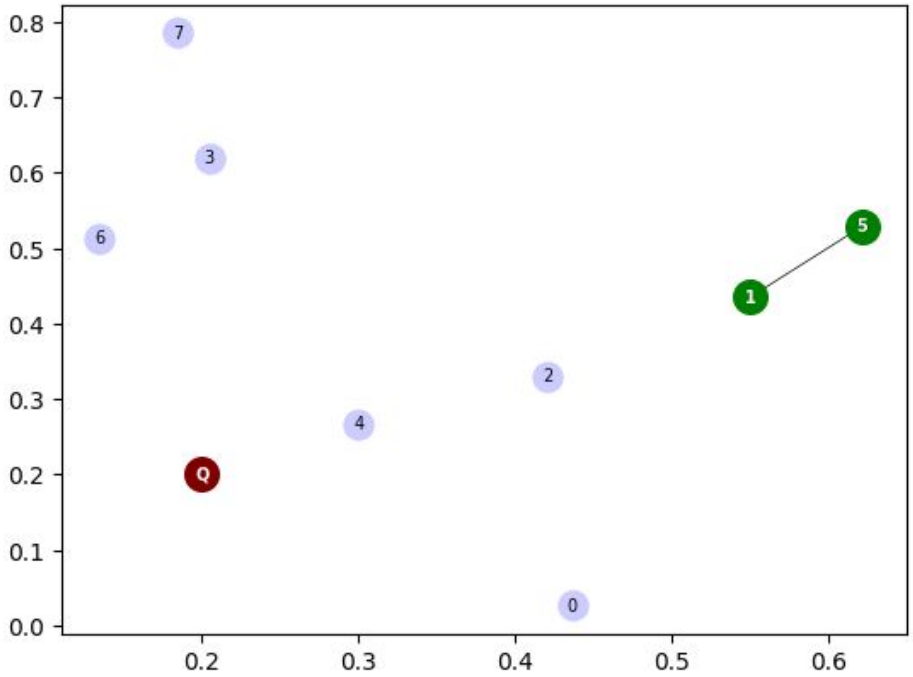
Navigable Small World (NSW): строим граф



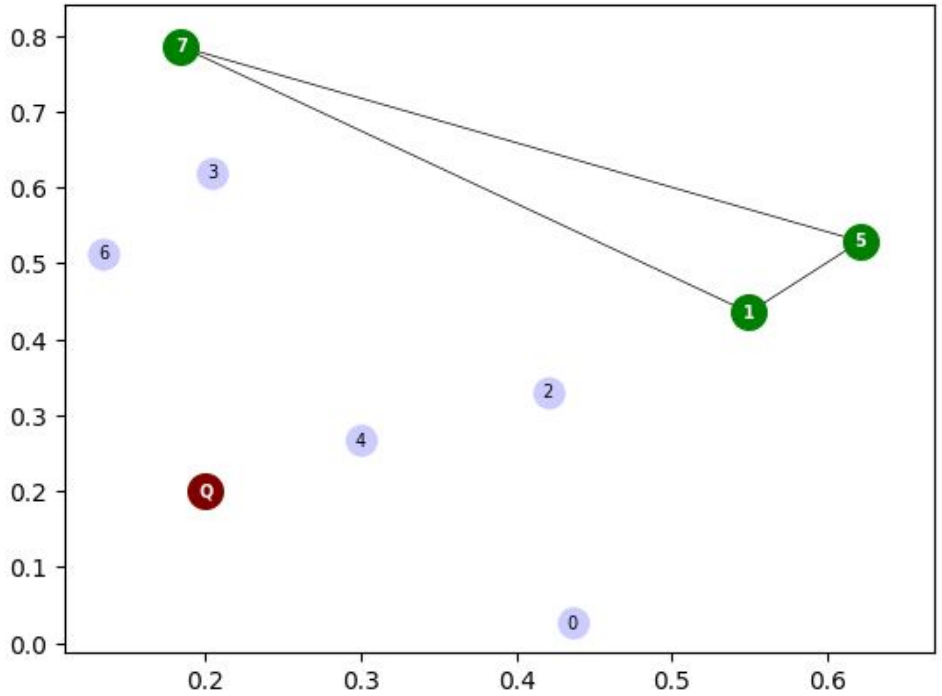
Берем случайную вершину (эмбеддинг) и добавляем в граф



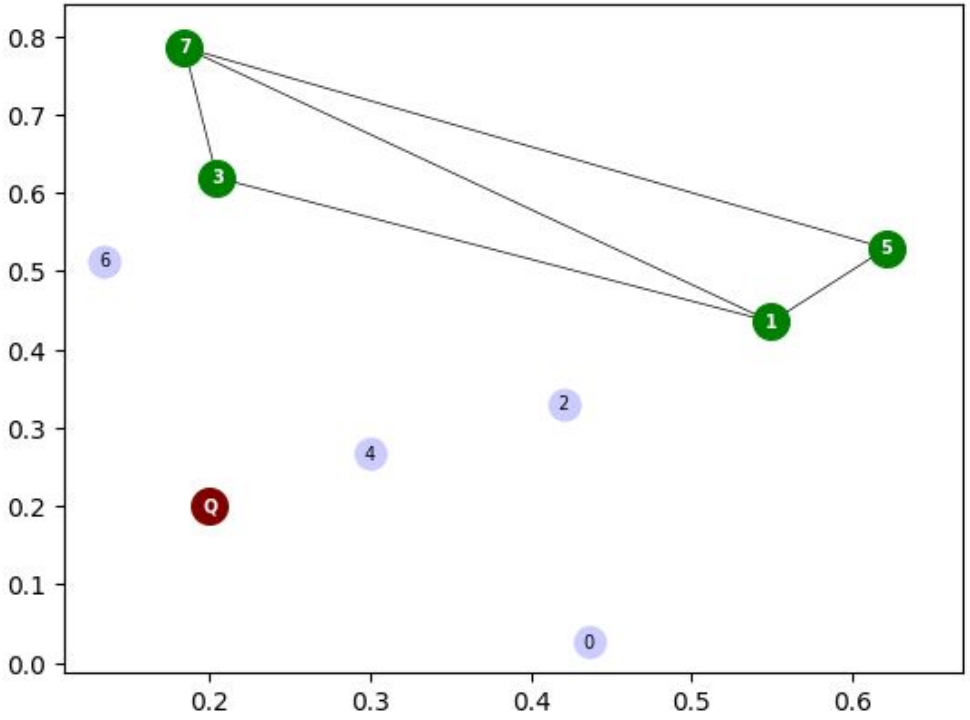
Каждую следующую случайную вершину соединяем с двумя ближайшими, которые уже есть в графе



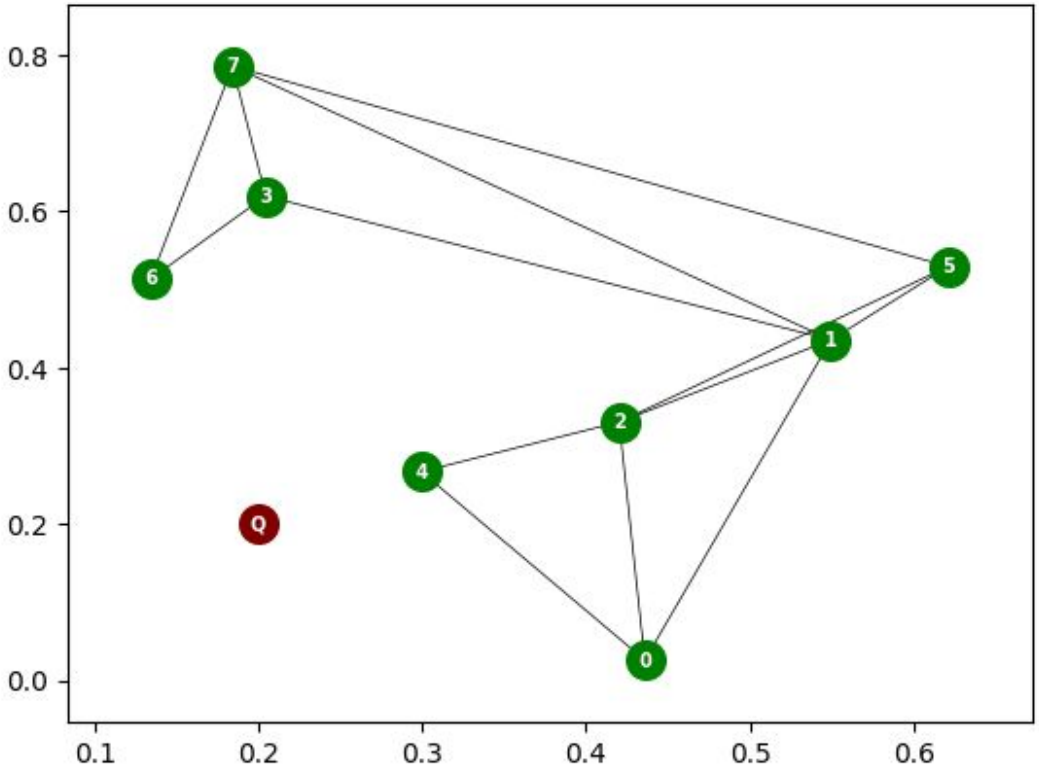
Каждую следующую случайную вершину соединяем с двумя ближайшими, которые уже есть в графе



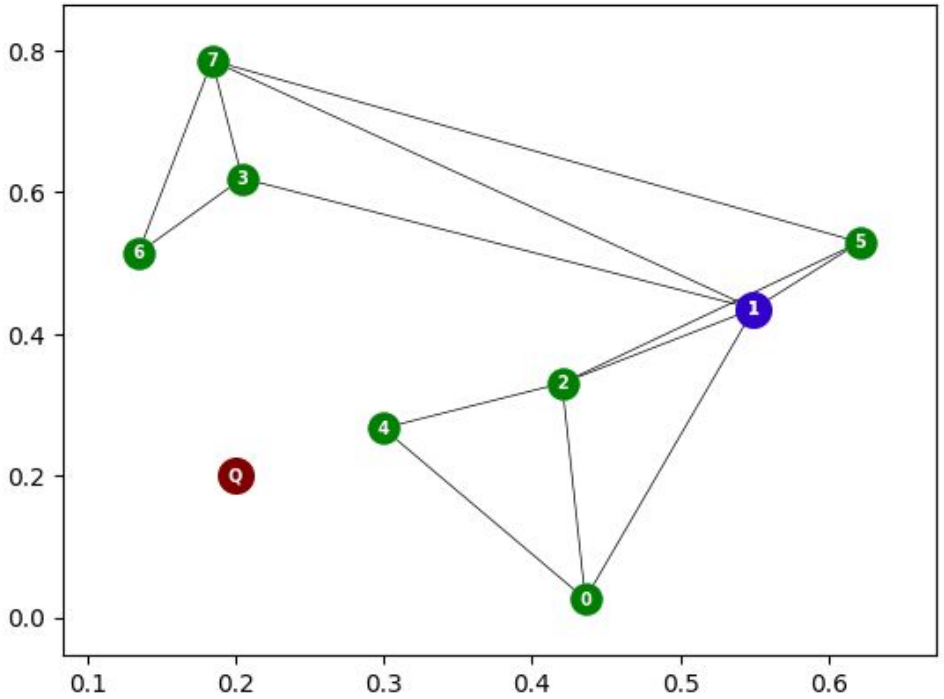
Каждую следующую случайную вершину соединяем с двумя ближайшими, которые уже есть в графе



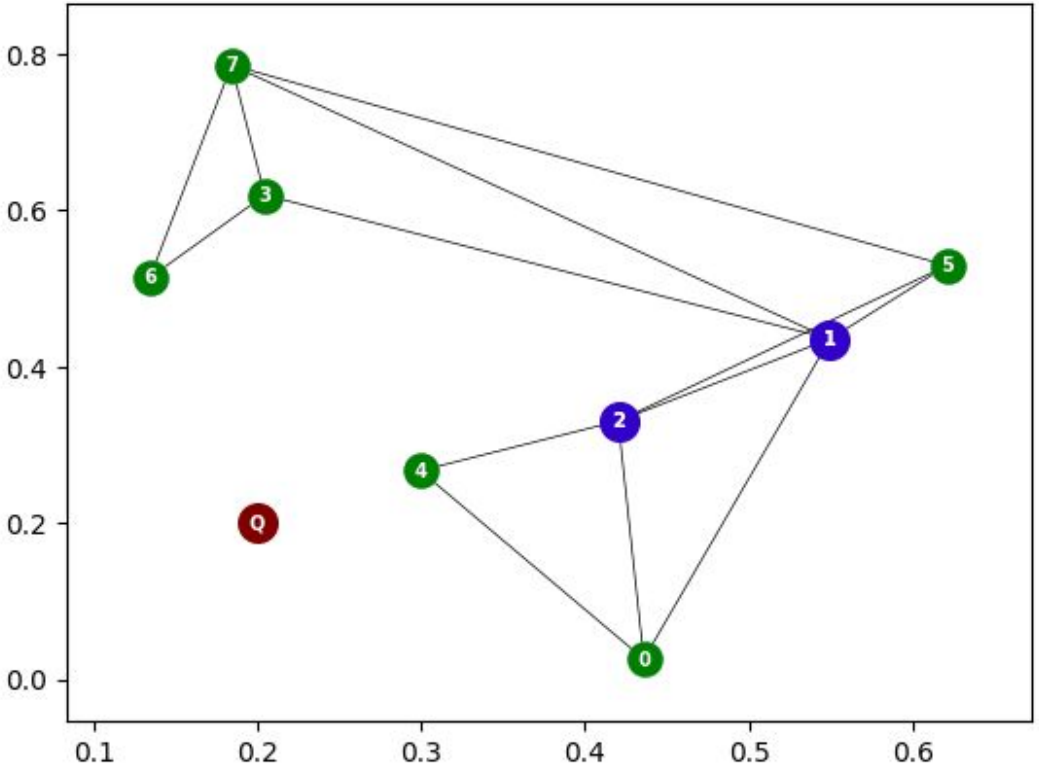
Navigable Small World (NSW): получаем граф



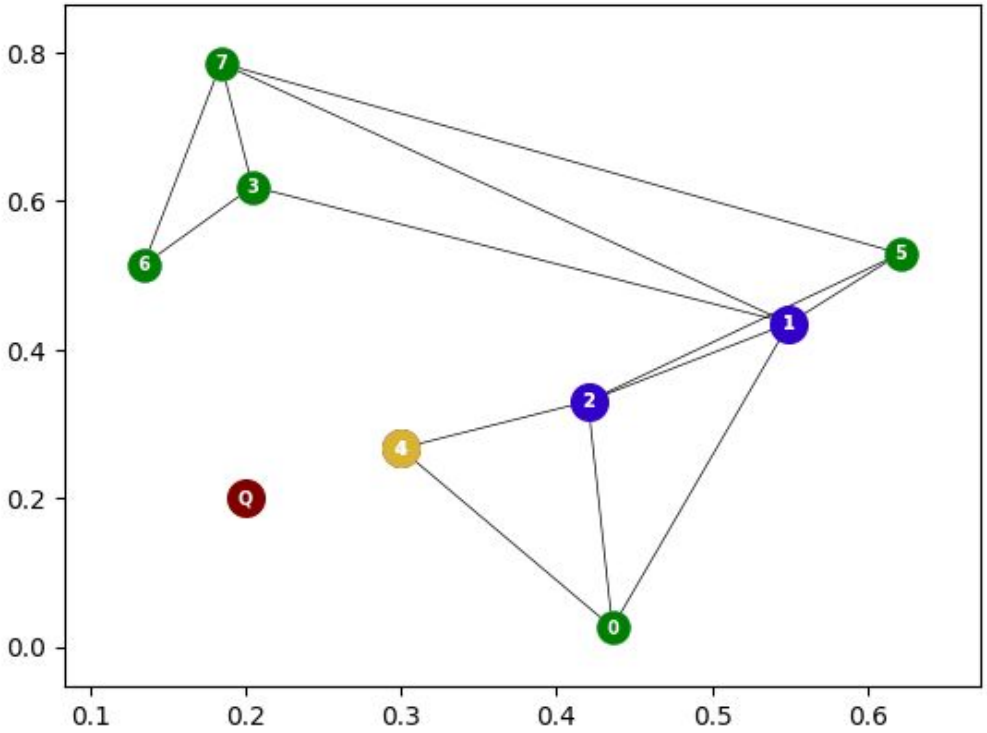
NSW: поиск по графу - берем случайную вершину и считаем расстояние от q до нее и ее соседей



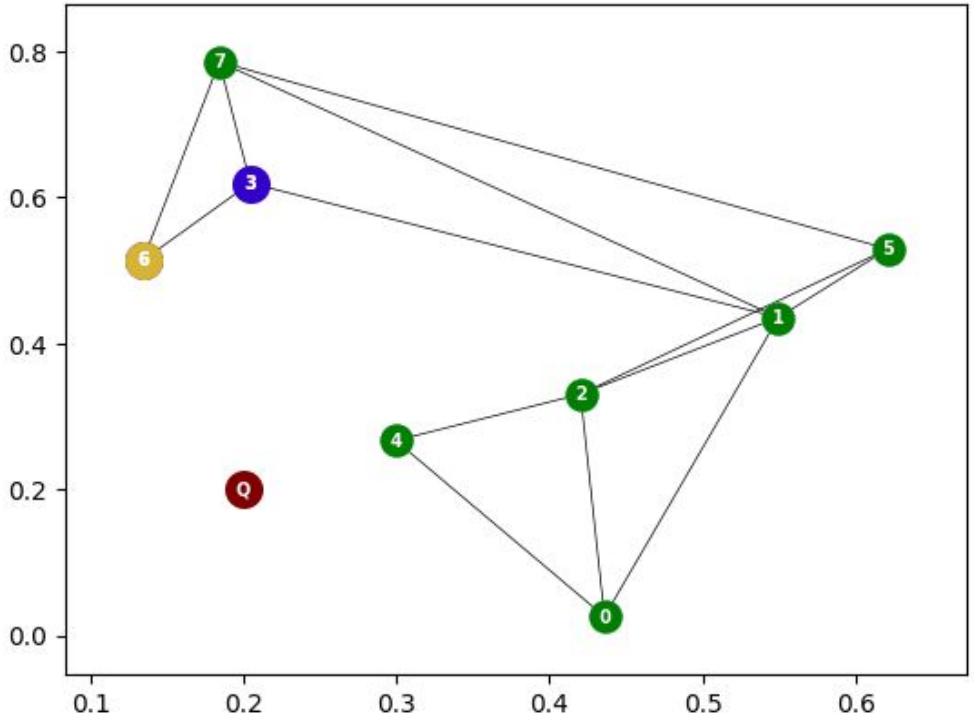
NSW: поиск по графу - идем в ближайшую к q из соседей



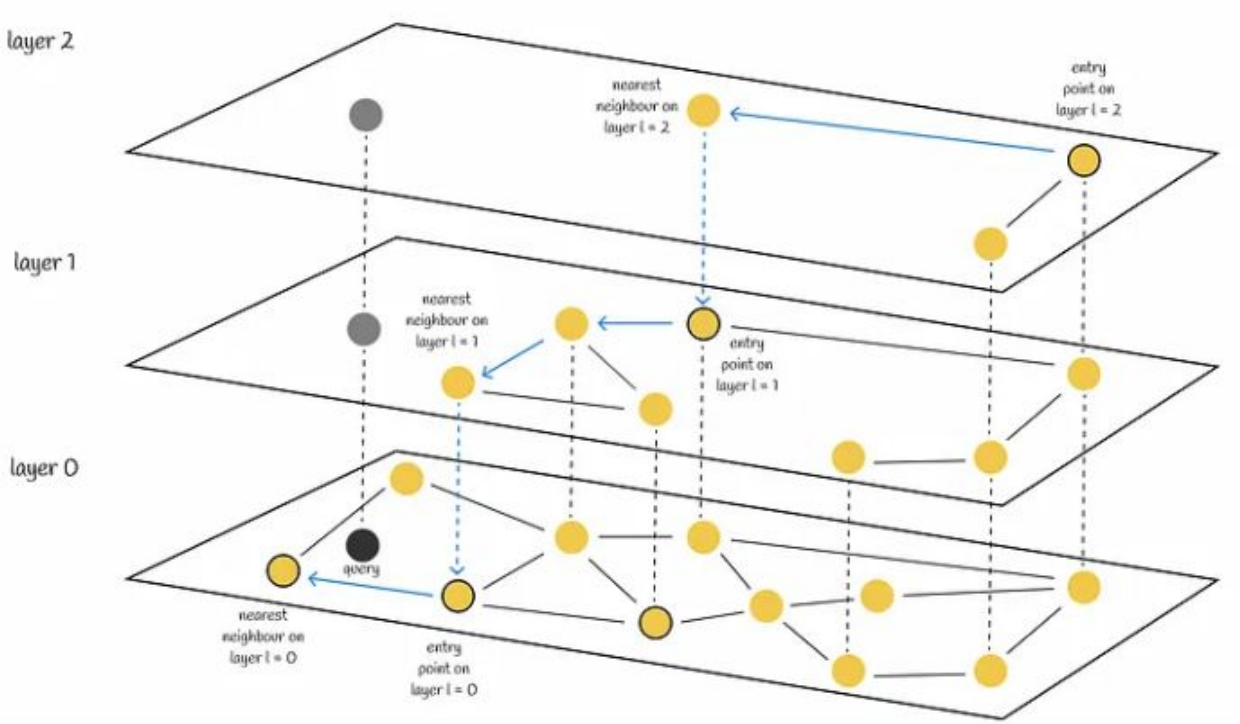
NSW: поиск по графу - идем в ближайшую к q из соседей и из нее повторяем процесс



NSW: итог - получаем приближенного ближайшего соседа. Почему? Зависит от стартовой точки.



HNSW - Hierarchical NSW - когда запросов много!



HNSW - Hierarchical NSW - когда запросов много!

Строим уровневые графы:

- Каждый вектор при добавлении получает случайный максимальный уровень L - то есть точка добавляется на уровни 1, 2, …, L
- На каждом уровне точка соединяется с `M` ближайшими соседями (по выбранной метрике)

Добавление точки (объекта) в граф:

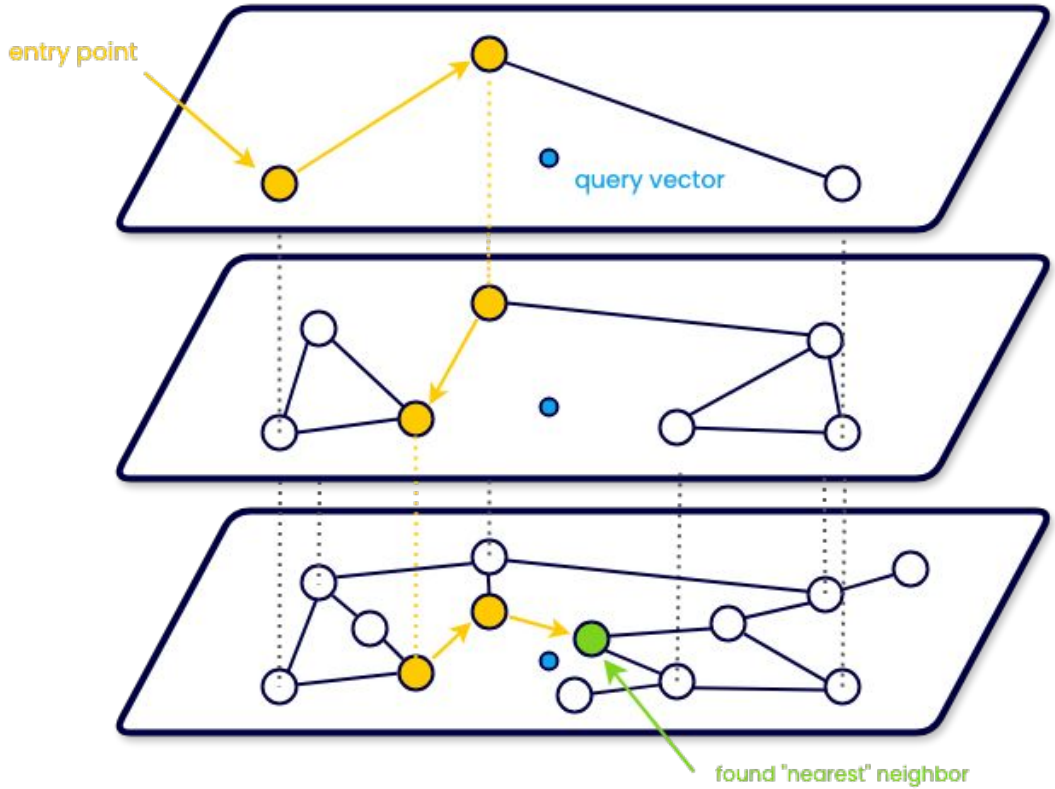
- Задаем точке случайный уровень L
- Добавляем точку на верхний уровень и соединяем с M ближайшими соседями
- Далее добавляем точку на следующий уровень и ищем там M ближайших соседей и так далее

HNSW: поиск по графу

Когда приходит эмбеддинг запроса:

- Начало поиска — стартуем с произвольной (или заранее выбранной) точки на самом верхнем уровне
- Грубая навигация — двигаемся по рёбрам, пока не найдём ближайшую точку на этом уровне
- Спуск вниз — переходим на уровень ниже, используя найденную точку как старт
- Точный поиск — на нижнем (нулевом) уровне ищем ближайших соседей до нужного k.

HNSW: поиск по графу



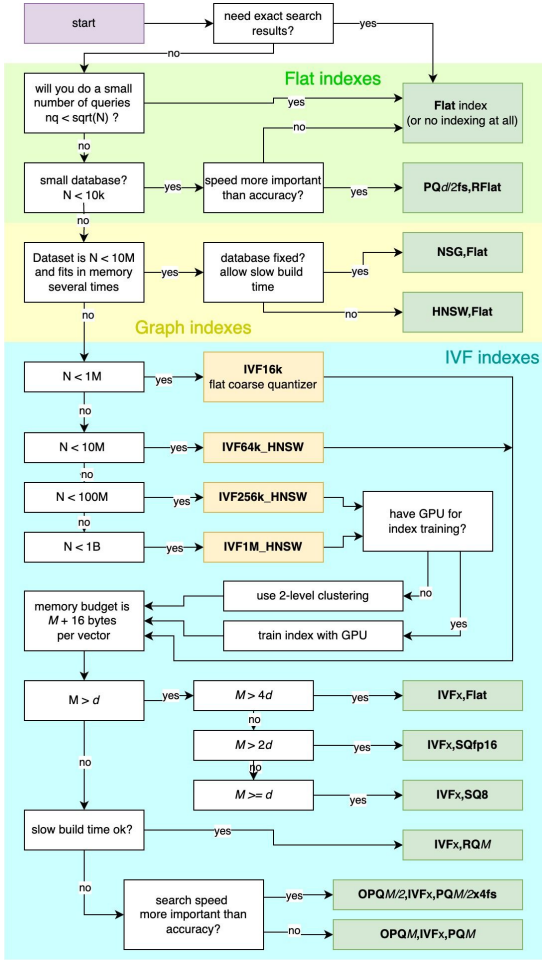
Библиотеки: FAISS

FAISS (Facebook AI Similarity Search) — это библиотека от Facebook для эффективного поиска ближайших соседей (ANN — Approximate Nearest Neighbors) в больших коллекциях векторов.

Индекс в FAISS

- ✔️ **Индекс - это структура, в которой физически лежат эмбеддинги**
- ✔️ **Вид структуры зависит от выбранного алгоритма**
  - ✔️ **Flat = массив**
  - ✔️ **HNSW = многоуровневый граф**
  - ✔️ **IVF = кластеры**
  - ✔️ **PQ = сжатые векторы в кодовых блоках**

FAISS


start
need exact search results?
yes
no
Flat indexes
will you do a small number of queries nq < sqrt(N) ?
yes
Flat index (or no indexing at all)
no
small database? N < 10k
yes
speed more important than accuracy?
no
yes
PQd/2fs,RFlat
no
Dataset is N < 10M and fits in memory several times
yes
database fixed? allow slow build time
yes
NSG,Flat
no
HNSW,Flat
Graph indexes
IVF indexes
N < 1M
yes
IVF16k flat coarse quantizer
no
N < 10M
yes
IVF64k_HNSW
no
N < 100M
yes
IVF256k_HNSW
have GPU for index training?
no
N < 1B
yes
IVF1M_HNSW
use 2-level clustering
no
memory budget is M + 16 bytes per vector
train index with GPU
yes
M > d
yes
M > 4d
yes
IVFx,Flat
no
M > 2d
yes
IVFx,SQfp16
no
M >= d
yes
IVFx,SQ8
no
slow build time ok?
yes
IVFx,RQM
no
search speed more important than accuracy?
yes
OPQM/2,IVFx,PQM/2x4fs
no
OPQM,IVFx,PQM

Практика

https://colab.research.google.com/drive/1o-zBXgLiPQ_b4rO2UgBu2F82EkR-8dWp?usp=sharing