Approximate Nearest Neighbor Search in Recommender Systems

Yury Malkov

About me

PhD in laser physics (2015)

Interests:

- CV/NLP
- Similarity search
- Recommender systems (Twitter, Verse)

Rise of commercial vector search solutions

Finally ANN becomes mainstream!

Pinecone

zilliz

aws

pgvector

Weaviate

redis

ORACLE

qdrant

elastic

VectorDB

vespa cloud

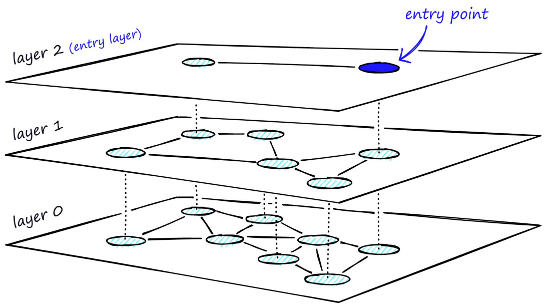
Astra DB

Approximate Nearest Neighbor Search Solutions

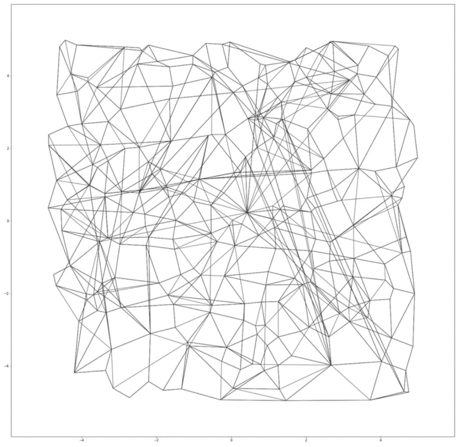
Many matured methods and libraries

- HNSW
- DiskANN and modifications
- IVFPQ, IVF-HNSW
- Scann, ngt, nvidia RAFT, song, cuhnsw…

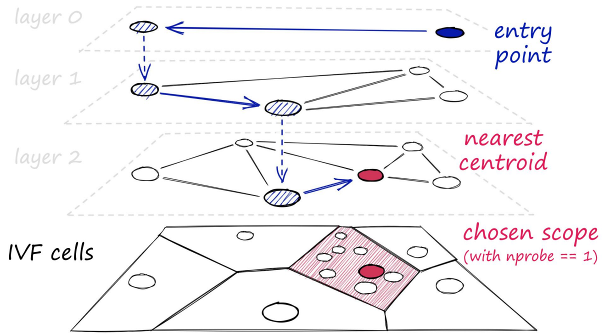
HNSW



DiskANN



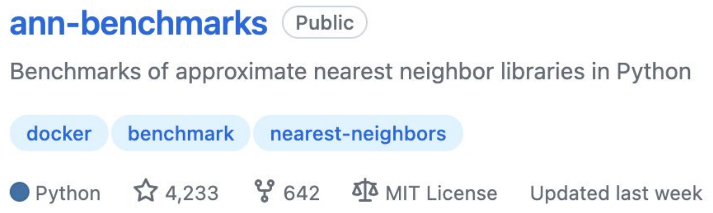
IVF-HNSW



Benchmarks

ann-benchmarks is very useful, but limited to dense L2 and cosine, single thread search vs ANN recall at million-scale.

Better, more diverse benchmarks are evolving (OOD, billion-scale, sparse)


ann-benchmarks Public

Benchmarks of approximate nearest neighbor libraries in Python

docker benchmark nearest-neighbors

Python ☆ 4,233 642 MIT License Updated last week

NeurIPS'23 Competition Track: Big-ANN

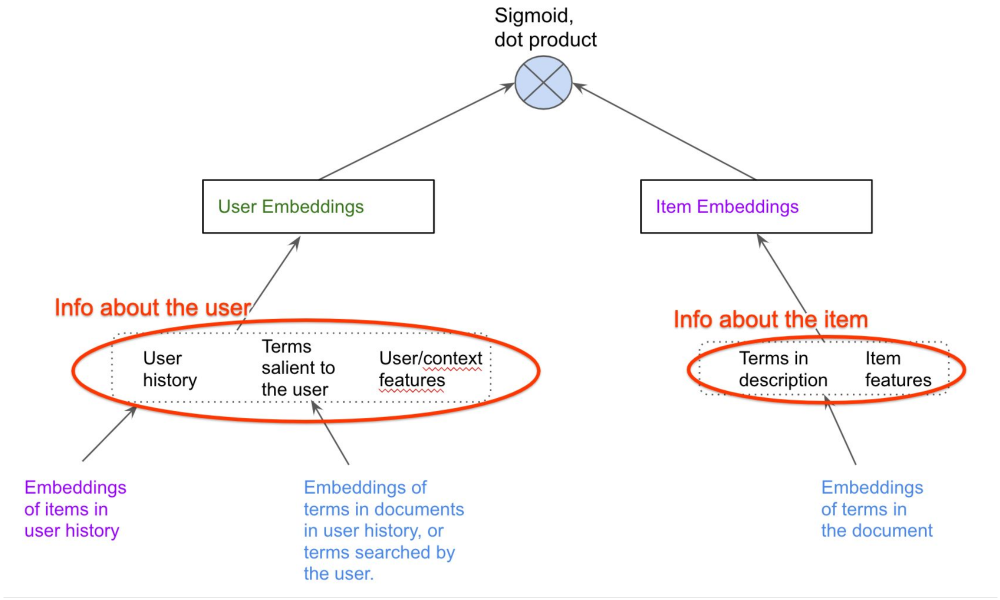
Supported by Microsoft Pinecone aws zilliz

Approximate nearest neighbor search applications

1. Recommender Systems

- Typical goal is recommending an item for a user (search, home timeline, notifications).
- A lot of $$$, but not generalizable beyond a company due to reliance on counting/graph/collaborative data.
- Huge corpus (millions to trillions), inference cost is very important => multistage funnel with ANN only a part.
- Usually homegrown solutions/infra for CPU (e.g. Meta, Google, Snap, Pinterest, Verse, Twitter, Amazon, Microsoft).
- Have issues with reproducibility.

Two tower architecture for candidate generation:


Sigmoid, dot product
User Embeddings
Item Embeddings
Info about the user
User history
Terms salient to the user
User/context features
Info about the item
Terms in description
Item features
Embeddings of items in user history
Embeddings of terms in documents in user history, or terms searched by the user.
Embeddings of terms in the document

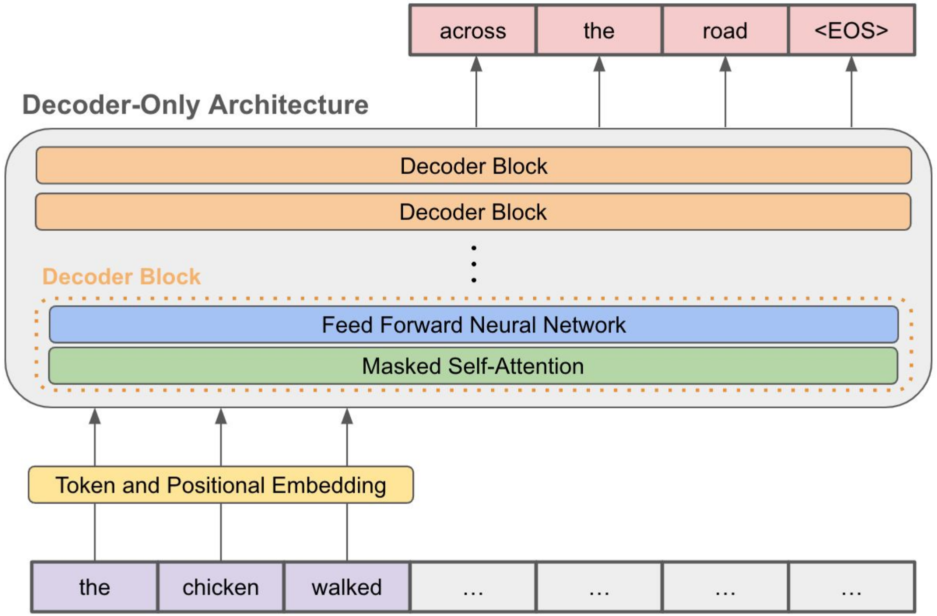
2. LLMs

LLMs had a spike of popularity with an introduction of ChatGPT.

Extremely generalizable. Likely will revolutionize RecSys with deep context understanding [1].

Known issues:

- Limited vocabulary
- Bounded context length
- Hallucinations (lack of grounding)

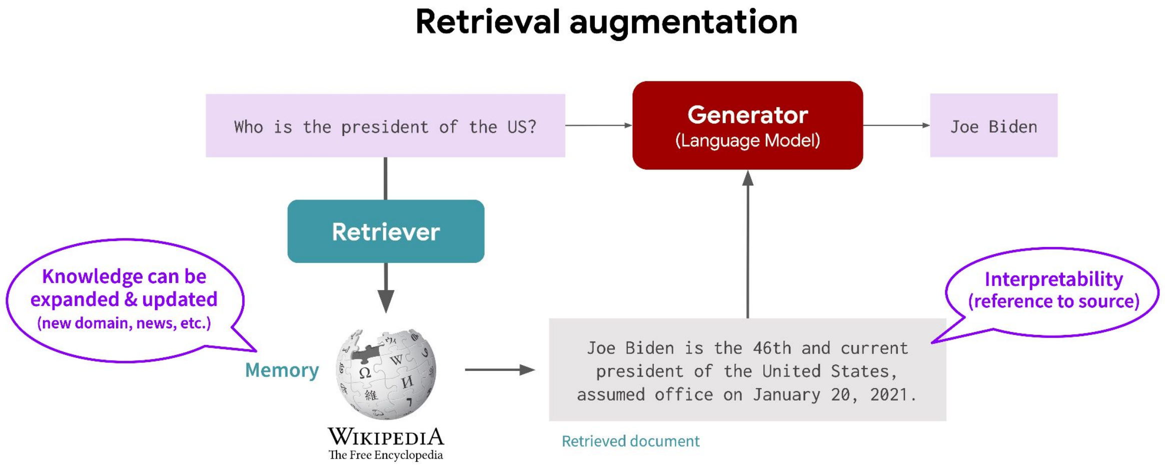


[1] Chen, Zheng. "PALR: Personalization Aware LLMs for Recommendation." *arXiv preprint arXiv:2305.07622* (2023).

2. LLM RAG (Retrieval Augmented Generation)

The most popular current application with LLM.

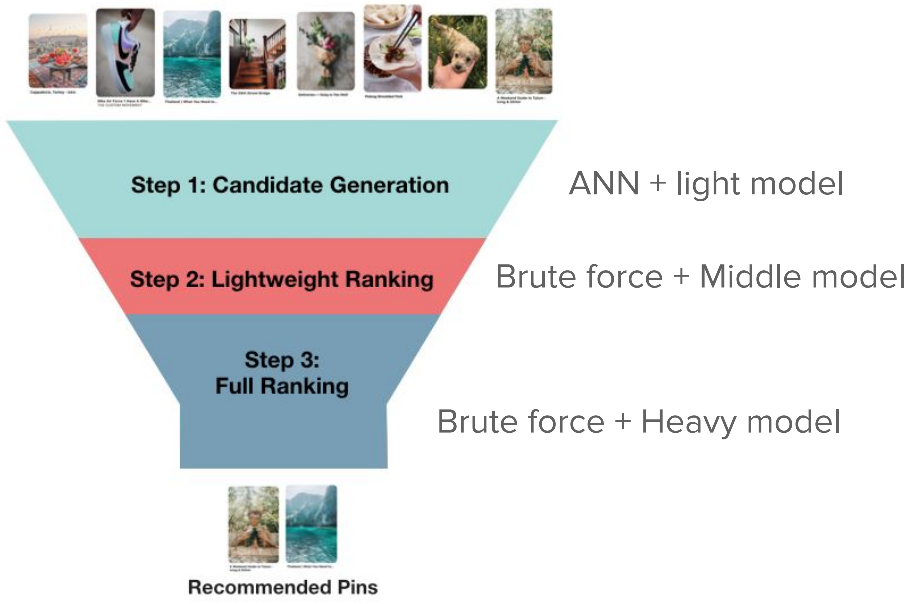
- Simple and human-interpretable alternative to finetuning in LLMs.
- Adds a level of grounding for generated results.
- For some applications (e.g. user data), scale is less important compared to consistency and availability at streaming updates.



Tighter ANN <> ranking funnel integration

Typical Large-scale Recommender Systems

- ANN is the **first part** of multi-stage funnel with cascaded models. Later cascades use **bruteforce** on the previous stage output.
- Queries and items by design have usually have different nature and features and live in different **incompatible** spaces. Those are distilled to L2/cosine for ANN thus losing information.
- Huge corpus for videos (1M-1T), high cost. Needs dedicated engineers to maintain.


Step 1: Candidate Generation
ANN + light model
Step 2: Lightweight Ranking
Brute force + Middle model
Step 3: Full Ranking
Brute force + Heavy model
Recommended Pins

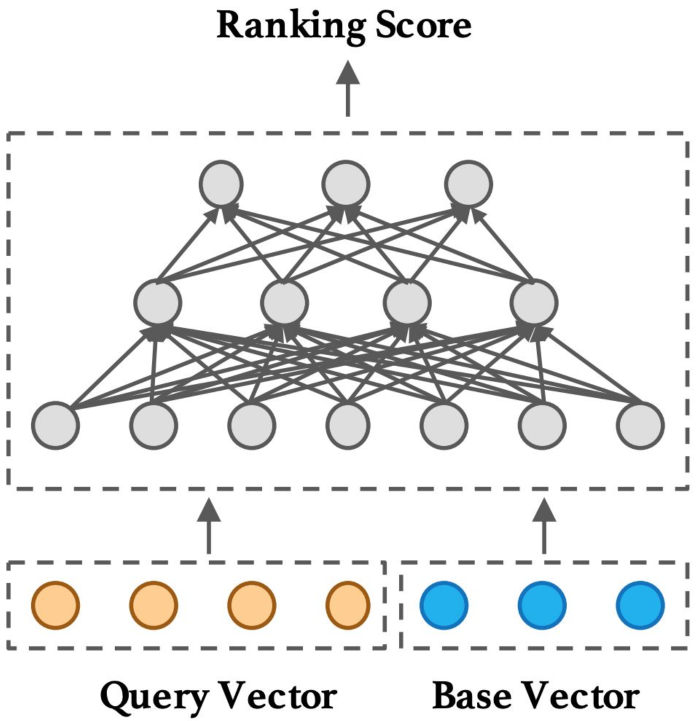
Using learnable (neural) distances

In a properly designed ML system, ranker is arguably the main component.

- Why not apply ANN to the ranker?

The problem is that queries and documents live in different spaces.

HNSW (and some other graph algorithms) do not restrict the items to be in the same space only distances matter. **But:** how to construct the index if there is no item-item distance?



Using L2 distance for data vectors (SL2G)

Solution - Ignores queries. Just use L2 distance for index construction on raw vectors.

Use supplied neural distance during search.

Gives a speedup, though baselines are questionable.

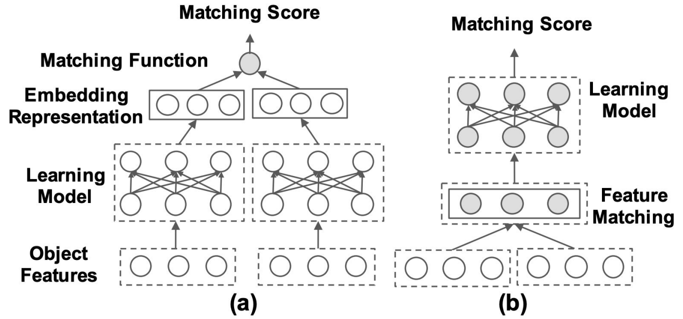
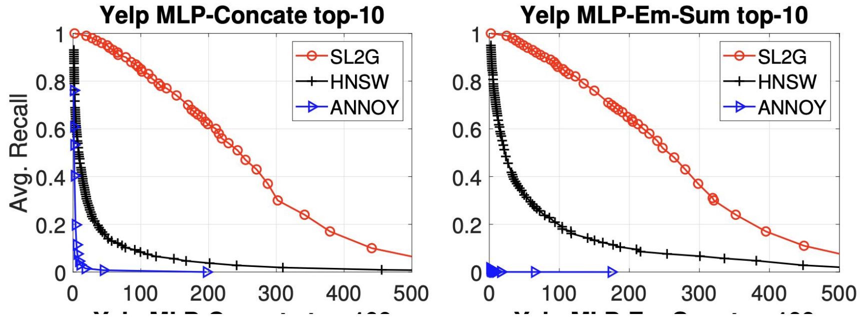


Figure 1: Neural network based matching models: (a) representation learning; (b) matching function learning.

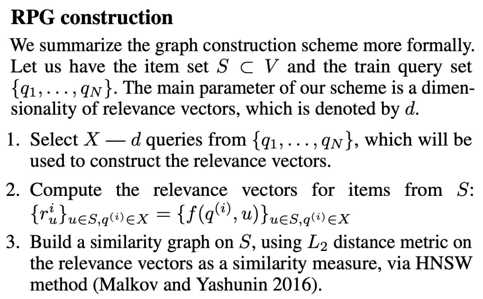


Tan, Shulong, et al. "Fast item ranking under neural network based measures." *Proceedings of the 13th International Conference on Web Search and Data Mining*. 2020.

Using projections from the query space

Solution - Using distances from queries as a metric. Creates a simple surrogate metric that can be used for graph construction.

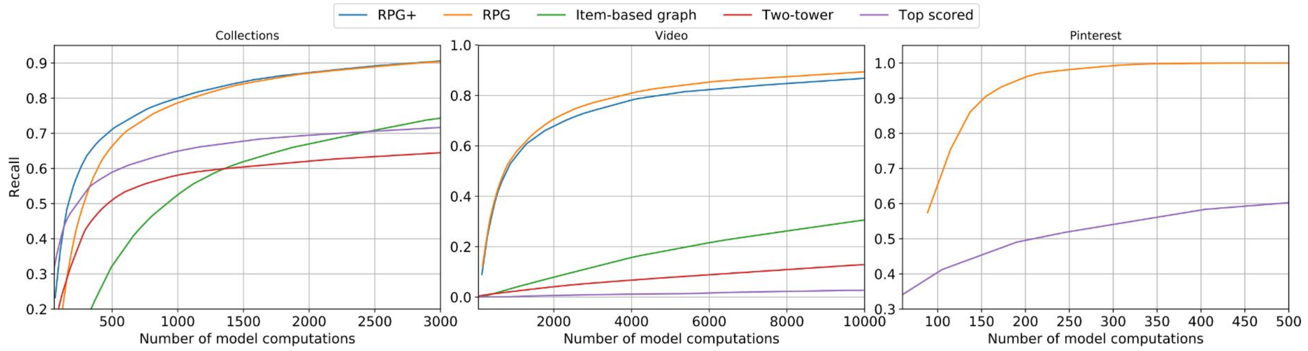
Works better than SL2G and two-tower.



RPG construction

We summarize the graph construction scheme more formally. Let us have the item set $S \subset V$ and the train query set $\{q_1, \dots, q_N\}$. The main parameter of our scheme is a dimensionality of relevance vectors, which is denoted by d.

1. Select X — d queries from $\{q_1, \dots, q_N\}$, which will be used to construct the relevance vectors.
2. Compute the relevance vectors for items from S: $\{r_u^i\}_{u \in S, q^{(i)} \in X} = \{f(q^{(i)}, u)\}_{u \in S, q^{(i)} \in X}$
3. Build a similarity graph on S, using L_2 distance metric on the relevance vectors as a similarity measure, via HNSW method (Malkov and Yashunin 2016).



1. Morozov, Stanislav, and Artem Babenko. Relevance Proximity Graphs for Fast Relevance Retrieval. *arXiv preprint arXiv:1908.06887*
2. Бритвина, Е. В., Крылов, В. В., & Мальков, Ю. А. (2013). Алгоритм максимизации релевантности, использующий графовые модели данных. *Труды НГТУ им. РЕ Алексеева*, (2 (99)), 75-83.

Using Item-Query bipartite graph ranking

Solution - Encoding the links through item-query ordering (connecting through queries that are closest to items).

Interesting approach, better than SL2G, but is not compared to projections (more direct vs discrete, needs a ton of queries).

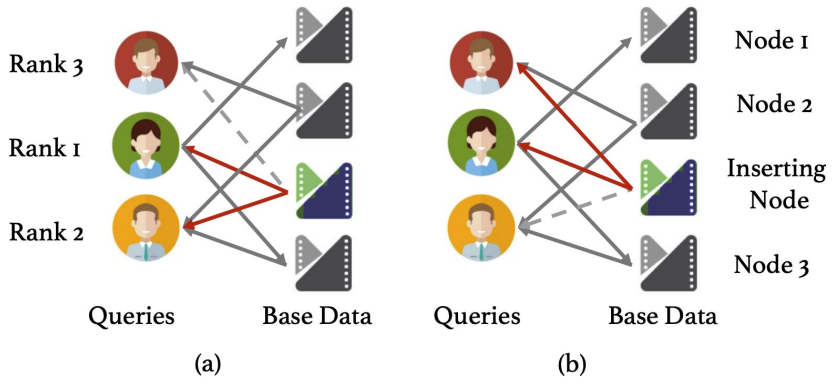
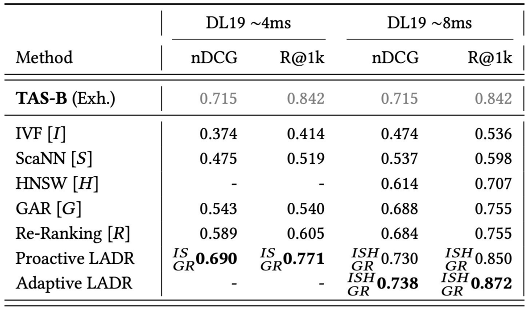


Figure 3: Illustrations for edge selection methods: (a) connect to top M candidates. (b) connect to diverse candidates by two-hop edge selection.

Tan, Shulong, Weijie Zhao, and Ping Li. "Fast neural ranking on bipartite graph indices." *Proceedings of the VLDB Endowment* 15.4 (2021): 794-803.

Graph reranking solutions in text retrieval

Solution - Use text-based cross encoders (heavy ranker) on embedding-based ANN graph and seeds to improve the accuracy/speed.



	DL19 ~4ms		DL19 ~8ms	
Method	nDCG	R@1k	nDCG	R@1k
TAS-B (Exh.)	0.715	0.842	0.715	0.842
IVF [*I*]	0.374	0.414	0.474	0.536
ScaNN [*S*]	0.475	0.519	0.537	0.598
HNSW [*H*]	-	-	0.614	0.707
GAR [*G*]	0.543	0.540	0.688	0.755
Re-Ranking [*R*]	0.589	0.605	0.684	0.755
Proactive LADR	${}^{IS}_{GR}$**0.690**	${}^{IS}_{GR}$**0.771**	${}^{ISH}_{GR}$0.730	${}^{ISH}_{GR}$0.850
Adaptive LADR	-	-	${}^{ISH}_{GR}$**0.738**	${}^{ISH}_{GR}$**0.872**

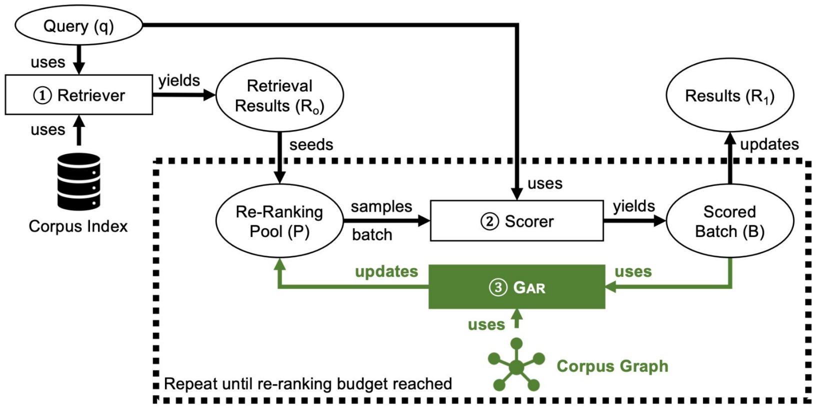


Figure 1: Overview of GAR. Traditional re-ranking exclusively scores results seeded by the retriever. GAR (in green) adapts the re-ranking pool after each batch based on the computed scores and a pre-computed graph of the corpus.

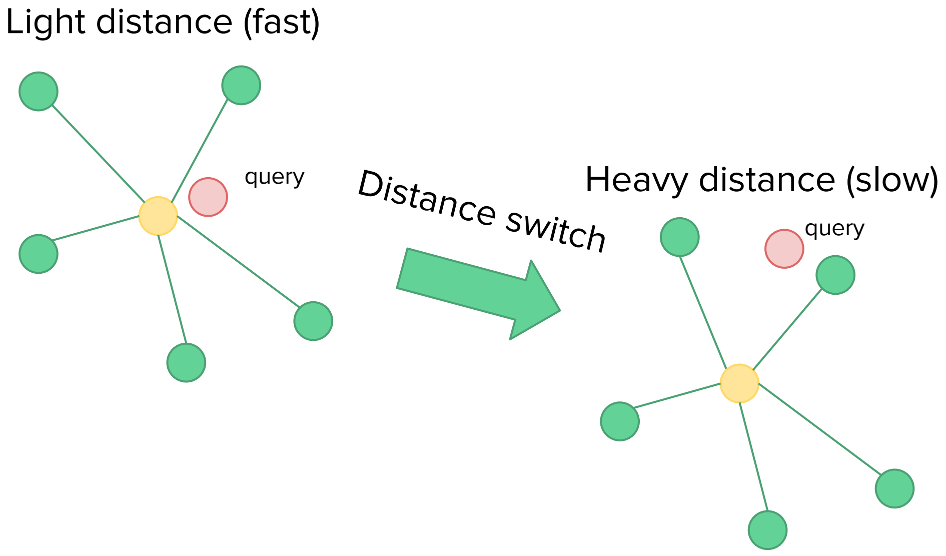
1. MacAvaney, Sean, Nicola Tonellotto, and Craig Macdonald. "Adaptive re-ranking with a corpus graph." *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 2022.
2. Kulkarni, Hrishikesh, et al. "Lexically-Accelerated Dense Retrieval." *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2023.

Funnel => Cascaded graph search

Instead of brute force search in later funnel stages we transfer the seeds, switching distance function during graph traversal.

The ranker can undo the recall errors in previous stages (which is impossible with the classical funnel).

Shown to be beneficial for retrieval with an embedding-based graph [3].



1. Chen, R., Liu, B., Zhu, H., Wang, Y., Li, Q., Ma, B., ... & Zheng, B. Approximate nearest neighbor search under neural similarity metric for large-scale recommendation. ACM 2022 (pp. 3013-3022).
2. Boytsov, L., Novak, D., Malkov, Yu., & Nyberg, E.. Off the Beaten Path: Let's Replace Term-Based Retrieval with k-NN Search. CIKM 2016
3. MacAvaney, Sean, Nicola Tonellotto, and Craig Macdonald. "Adaptive re-ranking with a corpus graph." Proceedings of the 31st ACM International Conference on Information & Knowledge Management. 2022.

Other optimizations for recommender systems

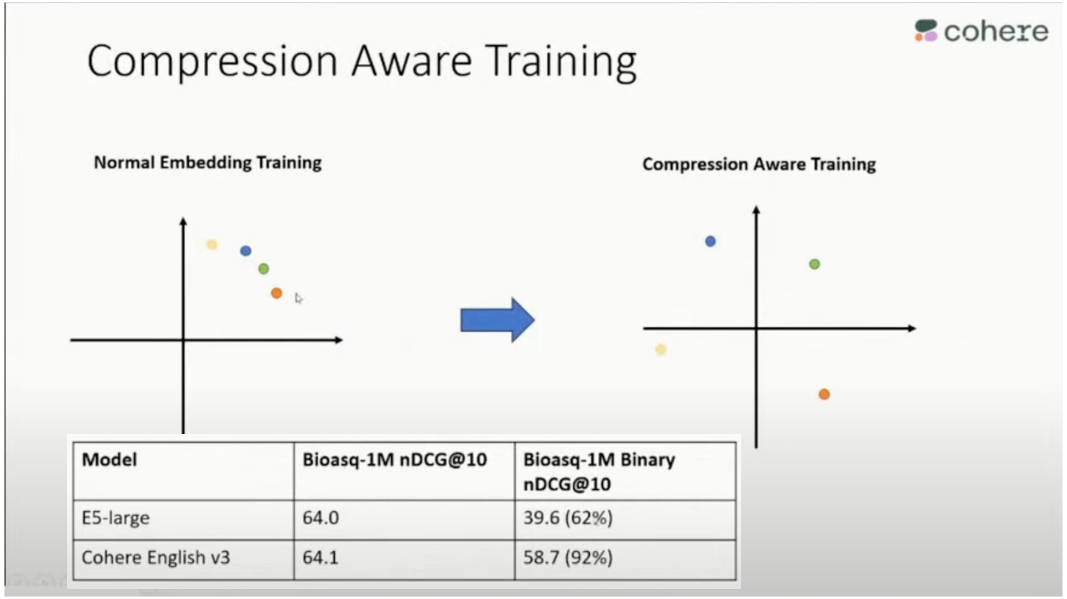
ANN-aware training, binarization

Large embeddings are costly.

Training the model so that common compression methods do not degrade the performance that much.

Examples of compatible embeddings:

- Cohere [1]
- OpenAI AdaV2 [2]



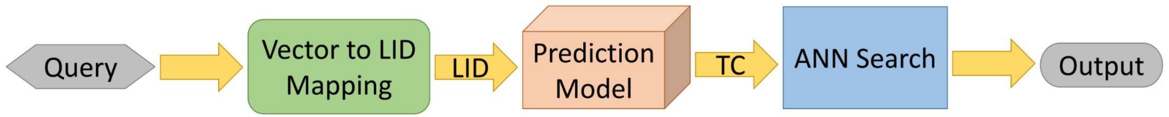
Compression Aware Training

Model	Bioasq-1M nDCG@10	Bioasq-1M Binary nDCG@10
E5-large	64.0	39.6 (62%)
Cohere English v3	64.1	58.7 (92%)

[1] Nils Reimers's presentation: https://www.youtube.com/watch?v=Abh3YCahyqU
[2] QDrant binary embeddings test for adaV2 https://qdrant.tech/articles/binary-quantization/

Dynamic early stopping criteria

Based on the local parameters (LID, graph structure), the stopping condition can be altered to save compute on simple samples and improve recall on complicated ones.



(b) Workflow of `Tao`

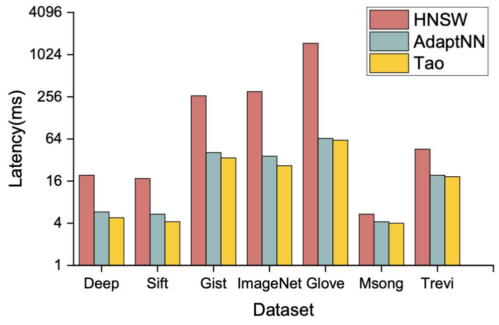


Figure 8: Latency for plain HNSW, `AdaptNN` and `Tao`.

1. Yang, Kaixiang, et al. "Tao: A Learning Framework for Adaptive Nearest Neighbor Search using Static Features Only." *arXiv preprint arXiv:2110.00696* (2021).
2. Li, Conglong, et al. "Improving approximate nearest neighbor search through learned adaptive early termination." 2020 ACM SIGMOD

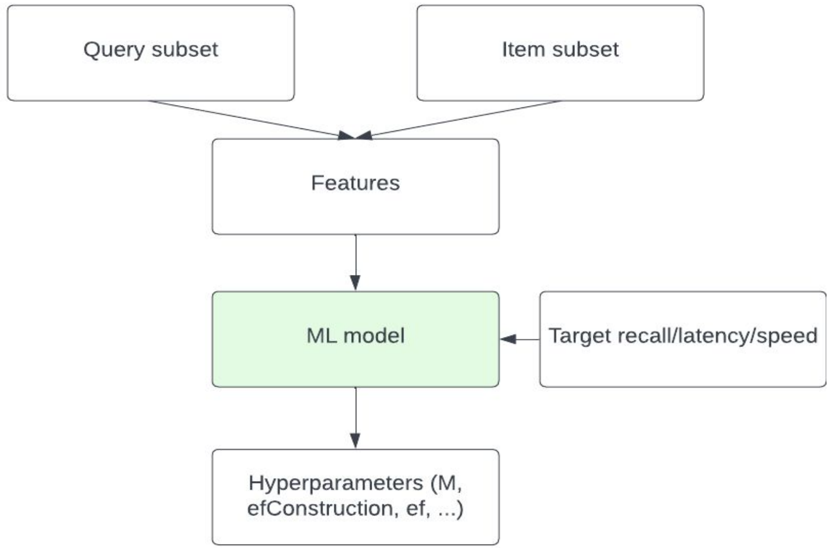
Estimating the ANN hyperparameters

Tuning the hyperparameters for algorithms might be hard, especially if the users don’t own the holistic ANN system, which is a real problem.

Optimal hyperparameters are a function of the dataset (e.g. LID, graph characteristics) and target characteristics.

In can be estimated from samples.

Very little published work.


Query subset
Item subset
Features
ML model
Target recall/latency/speed
Hyperparameters (M, efConstruction, ef, ...)

Thank you for the attention!