Retrieval In The LLM Era

Soumen Chakrabarti

+

Many Collaborators





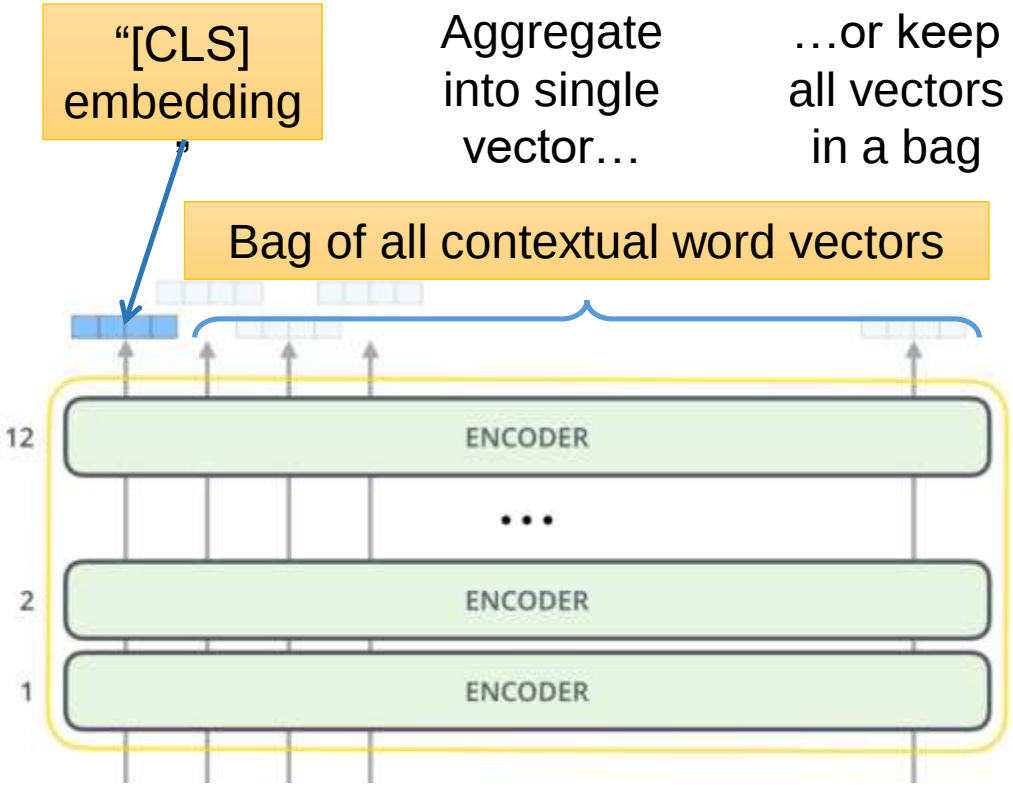

Brief history of search and retrieval

Discrete word-based 'lexical' indices based on inverted lists, super optimized for billions of documents, but suffering from "lexical gap"

'Semantic' indices built upon continuous embeddings of words (and audio, image, video, etc.), based on locality sensitive hashing, vector databases, hierarchical search networks

Text encoder for dense retrieval

- Dense encoder outputs a "[CLS] vector" in $\mathbb{R}^D$ representing the whole passage
- And also outputs one contextual vector per token/word
  - Bank, current
- Early dense retrieval used single vector per passage
- Recent systems use multi-vector representations

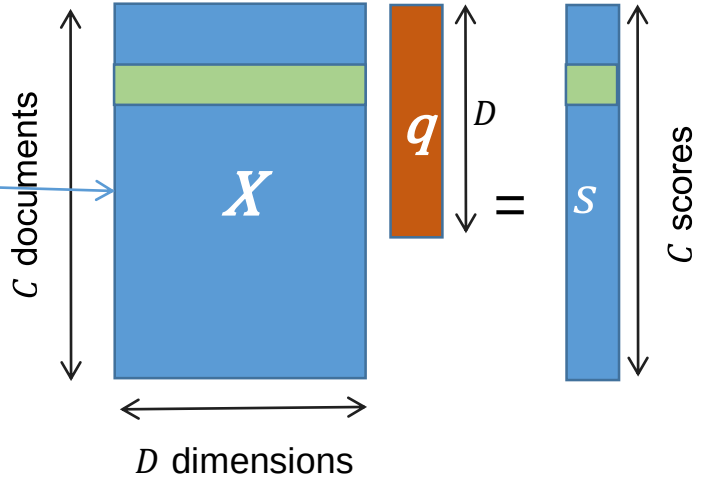


[CLS] He swam near the **bank**$_1$ because the **current**$_1$ was swift.

[CLS] His **bank**$_2$ offered a low interest on **current**$_2$ accounts.

Single-vector dense retrieval

- C docs in corpus, each condensed into a single "CLS vector" in $\mathbb{R}^D$
- Dense $C \times D$ corpus matrix
- Matrix-vector multiply with query vector to get scores $\boldsymbol{q} \cdot \boldsymbol{x}_c$
- Pick K rows (docs) with max dot
- Takes $O(CD)$ time
- Challenge: spend time that scales very slowly (sub-linearly) with C
- Some scaling with D, K tolerable



Pick K documents (row indexes) with top scores

Assume all vectors have L2 norm = 1, so dot = cosine similarity

Passage retrieval for question answering

Query: Birthday of composer of the film Sruthilayalu

Top documents when scored independently (K=3)

1. "In my opinion, Sruthilayalu is a breakout film by K Viswanath…"
2. "Sruthilayalu is a 1987 Telugu film…soundtrack composed by K V Mahadevan…"
3. "Sruthilayalu is unable to bring out the emotion in the audience…"

Gold item set

- "Sruthilayalu is a 1987 Telugu film…soundtrack composed by K V Mahadevan…"
- "K V Mahadevan (14 March 1918 – 21 June 2001) is a…"

Multiple documents need to **collaborate** to return correct response

Traditional top-K retrieval

- Existing implementations try **increasing** K and hoping superset of gold item set is retrieved, which they then filter.
- This may not work if the top-K items are redundant (e.g. different reviews of the same movie "Sruthilayalu").
- The top-K items may also fail to cover all aspects of the query (e.g. composer of the film is not mentioned, or the document matches partially but is irrelevant).
- Top-K paradigm with single document scoring inherently limited.

Top Documents when Scored Independently (K=5)

1. "In my opinion, Sruthilayalu is a breakout film by K Viswanath…"
2. "Sruthilayalu is a 1987 Telugu film…soundtrack composed by K V Mahadevan…"
3. "Sruthilayalu is unable to bring out the emotion in the audience…"
4. "The soundtrack to the film…1990…by Joe Hisaishi, who composed for the film."
5. "Sruthi is a 1987 Malayalam film directed by Mohan…"

Table retrieval for question answering

($ in millions)	As of December	
	2016	2015
Performing loans and long-term receivables		
Aggregate contractual principal in excess of fair value	$ 478	$ 1,330
Loans on nonaccrual status and/or more than 90 days past due		
Aggregate contractual principal in excess of fair value	8,101	9,600
Aggregate fair value on loans on nonaccrual status and/or more than 90 days past due	2,138	2,391

The table below presents information about our funding sources.

	As of December			
$ in millions	**2018**		**2017**	
Deposits	$158,257	25%	$138,604	23%
Collateralized financings:				
Repurchase aggrements	78,723	13%	84,718	14%
Securities loaned	11,808	2%	14,793	2%
Other secured financings	21,433	3%	24,788	2%
Total collateralized financings	111,964	18%	124,299	20%
Unsecured short-term borrowings	40,502	7%	46,922	8%
Unsecured long-term borrowings	224,149	36%	217,687	36%
Total shareholders' equity	90,185	14%	82,243	13%
Total funding sources	$625,057	100%	$609,755	100%

Question: What is the sum of securities loaned in 2017 and aggregate contractual principal in excess of fair value in 2015 (in millions)?

- Query not highly similar to whole table
- No single table element covers whole query
- Top-K table element retrieval performs badly
- Need small subset of table elements which collectively cover large (colored) spans of question
- Pooled single vector representation unlikely to work

Discontent with single vectors

"You can't cram the meaning of a whole %&!$# sentence into a single $&!#* vector!" — Ray Mooney (2014)

"The pooling operation used in convolutional neural networks is a big mistake and the fact that it works so well is a disaster." —Geoffrey Hinton (2014)

Single-vector

Google DeepMind

On the Theoretical Limitations of Embedding-Based Retrieval

Orion Weller[*,1,2], Michael Boratko[1], Iftekhar Naim[1] and Jinhyuk Lee[1]
[1]Google DeepMind, [2]Johns Hopkins University

ColBERT: Multi-vector text matching

- Pooled single vector representation unlikely to work
- Represent text as bags of vectors
  $X_c = \{\boldsymbol{x}_{c,\ell} : \ell \in [L]\}, Q = \{\boldsymbol{q}_n : n \in [N]\}$
  - (For simplicity assume all queries have N vectors, all docs have L vectors each)
- Each query vector $\boldsymbol{q}_n$ gets the support of its best partner vector from doc:
  $\max_{\ell \in [L]} \boldsymbol{q}_n \cdot \boldsymbol{x}_{c,\ell}$
- Now add/avg over query vectors to get utility score of doc $s(X_c|Q) = \frac{1}{N} \sum_{n \in [N]} \max_{\ell \in [L]} \boldsymbol{q}_n \cdot \boldsymbol{x}_{c,\ell}$
- Still, each doc is scored to compete against other docs

Documents: From gladiators to coalitions

- C docs in corpus
  - Each X_c is a bag of vec
- Query Q can be satisfied by one doc
- Utility score of a doc $s(X_c|Q)$
- Find docs with K largest scores, as competing alternatives

- Fine grained 'doc'
  - Sentence, table cell, graph node
  - Fragmented text compared to full doc
- No single doc can satisfy query
- Find small $S \in [C]$ that collectively achieves large **utility** for Q
  - S is a subset of doc IDs

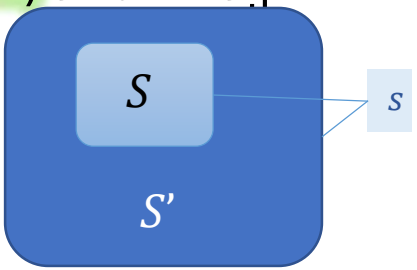
Coalition utility definition

- Recall, in gladiator mode,

$$s(X_c|Q) = \frac{1}{N}\sum_{\boldsymbol{q}\in Q}\max_{\ell\in[L]} \boldsymbol{q}_n \cdot \boldsymbol{x}_{c,\ell}$$

- Coalition is a subset S of doc indices
- Query vec $\boldsymbol{q} \in Q$ seeks best support from any vector in the (union of) all doc bags indexed by S

$$F(S|Q) = \frac{1}{N}\Sigma_{\boldsymbol{q}\in Q}\max_{\boldsymbol{x}\in\cup_{s\in S} X_s} \boldsymbol{q}\cdot\boldsymbol{x}$$



- Monotone: $S \subseteq S' \Rightarrow F(S|Q) \leq F(S'|Q)$
- Submodular (diminishing returns, greedy works):

$$s \notin S' \supseteq S' \Rightarrow F(S \cup s|Q) \geq F(S' \cup s|Q)$$

Marginal gain Δ from adding $c \notin S$ to S

- Query Q, doc set S already included, candidate (doc index) c
- $\Delta(c|S,Q) = F(S \cup c, Q) - F(S,Q)$

$$= \sum_{\boldsymbol{q} \in Q} F(S \cup c, \boldsymbol{q}) - F(S, \boldsymbol{q}) = \sum_{\boldsymbol{q} \in Q} \max\{F(c, \boldsymbol{q}), F(S, \boldsymbol{q})\} - F(S, \boldsymbol{q})$$

- Note: $\max\{a,b\} - b = [a-b]_+ = \max\{0, a-b\}$
- $\Delta(c|S,Q) = \sum_{\boldsymbol{q} \in Q} [F(c, \boldsymbol{q}) - F(S, \boldsymbol{q})]_+$
  - $F(c, \boldsymbol{q})$ is support from next candidate doc
  - $-F(S, \boldsymbol{q})$: does S already provide better support than c?

Indexing for successive max marginal gain

- Grow $\emptyset = S_0 \subset S_1 \subset \cdots \subset S_K$ in K rounds
- Add one doc c_k in each round: $S_k = S_{k-1} \cup c_k$
- Choose c_k to maximize marginal gain

$$c_k = \underset{c \notin S_{k-1}}{\operatorname{argmax}} \Delta(c | S_{k-1}, Q)$$

- Using ColBERT style, probe index for each query vector $\boldsymbol{q}$:

$$\underset{c \notin S}{\operatorname{argmax}} \Delta(c | S, \boldsymbol{q}) = \underset{c \notin S}{\operatorname{argmax}} [F(c, \boldsymbol{q}) - F(S, \boldsymbol{q})]_+$$

- Unlike ColBERT, this depends on already-collected doc set S
- But index cannot change as S changes — first hurdle
- Changing S must be folded into a lifted query $\tilde{\boldsymbol{q}}_S$

Lifted query representation

- Extend word embedding with one extra element

$$\Delta(c|S,\boldsymbol{q}) = \left[\max_{\boldsymbol{x}\in X_c} \begin{bmatrix} \boldsymbol{q} \\ F(S,\boldsymbol{q}) \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{x} \\ -1 \end{bmatrix}\right]_+$$

- Because $[\cdots]_+$ is monotone,

$$= \max_{\boldsymbol{x}\in X_c} \left[\begin{bmatrix} \boldsymbol{q} \\ F(S,\boldsymbol{q}) \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{x} \\ -1 \end{bmatrix}\right]_+ = \max_{\tilde{\boldsymbol{x}}\in X_c} [\tilde{\boldsymbol{q}}_S \cdot \tilde{\boldsymbol{x}}]_+$$

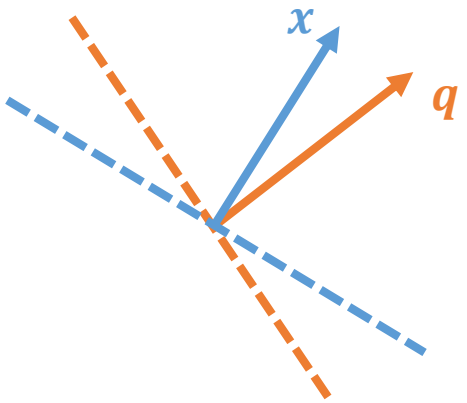
- $\tilde{\boldsymbol{q}}_S$ updated cheaply on the fly after each probe
  - A dense version of max marginal relevance
- $\tilde{\boldsymbol{x}}$ precomputed and indexed independent of queries
- Next hurdle: $[\cdots]_+$

From $\boldsymbol{q} \cdot \boldsymbol{x}_{c,\ell}$ to $\left[\boldsymbol{q} \cdot \boldsymbol{x}_{c,\ell}\right]_+$

- For each query vector $\boldsymbol{q}$, ColBERT probes index
- Index returns doc IDs $\{c\}$ such that doc c has some $\boldsymbol{x}_{c,\ell}$ to maximize $\boldsymbol{q} \cdot \boldsymbol{x}_{c,\ell}$
  - (Inverted list with vector key)
- These doc ID sets are merged and filtered
- But now we need to maximize $\left[\boldsymbol{q} \cdot \boldsymbol{x}_{c,\ell}\right]_+$
  - A different kind of index is needed
- Detour into foundation of single-vector-DB

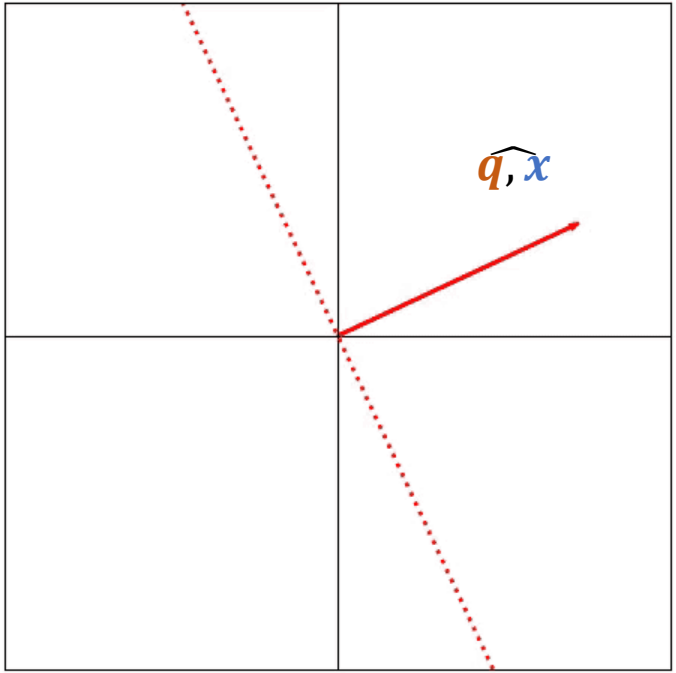
Projecting q and x on a random vector w

- Let $\widehat{q, x}$ be the angle between q and x
- Unit vectors q, x and their 'perpendicular hyperplanes' (dotted lines)
- A third unit vector w, with its perpendicular hyperplane, will rotate through 360° (equivalent to sampling)
- $\widehat{w, x}$ is the angle between w and x, in $(0, 180^\circ)$
- For unit length, $\cos(\widehat{w, x}) = w \cdot x$
  - If $\widehat{w, x} < \pi/2$, $w \cdot x > 0$ and $\text{sign}(w \cdot x) = 1$
  - If $\widehat{w, x} > \pi/2$, $w \cdot x < 0$ and $\text{sign}(w \cdot x) = -1$
- Same goes for $\widehat{w, q}$
- What fraction of w's give $\text{sign}(w \cdot x) \neq \text{sign}(w \cdot q)$?



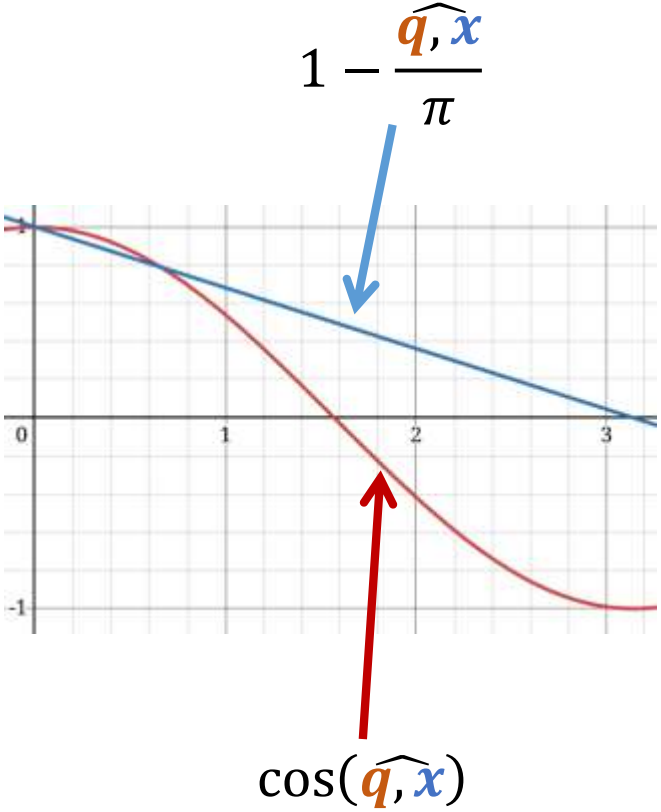
Probability of sign disagreement

- Signs disagree if $w \cdot q > 0$ and $w \cdot x < 0$, or vice versa
- I.e., q and x fall on different sides of gray hyperplane
- Gray vector w must be in orange cones for $\text{sign}(w \cdot q) \neq \text{sign}(w \cdot x)$ to hold, elsewhere signs agree
- $\Pr\left(\text{sign}(w \cdot q) \neq \text{sign}(w \cdot x)\right) = \frac{\widehat{q,x}}{\pi}$, conversely,
- $\Pr\left(\text{sign}(w \cdot q) = \text{sign}(w \cdot x)\right) = 1 - \frac{\widehat{q,x}}{\pi}$



Hyperplane hash for cosine ranking

- $\Pr\big(\text{sign}(\boldsymbol{w}\cdot\boldsymbol{q}) = \text{sign}(\boldsymbol{w}\cdot\boldsymbol{x})\big)$
  $= 1 - \frac{\widehat{\boldsymbol{q},\boldsymbol{x}}}{\pi} \overset{\text{rank}}{\longleftrightarrow} \cos(\widehat{\boldsymbol{q},\boldsymbol{x}})$
- In $[0,\pi]$ both these functions are monotone decreasing
- Ranking using one is equivalent to ranking using the other
- Works in any number of dimensions
- Amplify accuracy by sampling many $\boldsymbol{w}$



Return to subgoal: $\operatorname{argmax}_c [\boldsymbol{q} \cdot \boldsymbol{x}_c]_+$

- $[\boldsymbol{q} \cdot \boldsymbol{x}]_+$ is either $\boldsymbol{q} \cdot \boldsymbol{x}$ or 0
- Choose random unit vector $\boldsymbol{w}$
- Define $\Phi_{\boldsymbol{w}}(\blacksquare) = \frac{1}{\sqrt{2}}[\blacksquare, \operatorname{sign}(\boldsymbol{w} \cdot \blacksquare)\blacksquare] \in \frac{1}{\sqrt{2}}[\blacksquare, \pm\blacksquare]$
  - Here $\blacksquare$ is a placeholder for $\boldsymbol{q}$ or $\boldsymbol{x}$
- $\Phi_{\boldsymbol{w}}(\boldsymbol{q}) \cdot \Phi_{\boldsymbol{w}}(\boldsymbol{x}) = \frac{1}{2}[\boldsymbol{q} \cdot \boldsymbol{x} + \operatorname{sign}(\boldsymbol{w} \cdot \boldsymbol{q})\operatorname{sign}(\boldsymbol{w} \cdot \boldsymbol{x})\boldsymbol{q} \cdot \boldsymbol{x}]$
- $\Phi_{\boldsymbol{w}}(\boldsymbol{q}) \cdot \Phi_{\boldsymbol{w}}(\boldsymbol{x}) = \begin{cases} \boldsymbol{q} \cdot \boldsymbol{x}, & \text{if } \operatorname{sign}(\boldsymbol{w} \cdot \boldsymbol{q}) = \operatorname{sign}(\boldsymbol{w} \cdot \boldsymbol{x}) \\ 0, & \text{otherwise} \end{cases}$
- $\Pr\big(\operatorname{sign}(\boldsymbol{w} \cdot \boldsymbol{q}) = \operatorname{sign}(\boldsymbol{w} \cdot \boldsymbol{x})\big) = 1 - \frac{\widehat{\boldsymbol{q},\boldsymbol{x}}}{\pi}$ (as before)

Two cases: angle $\widehat{\boldsymbol{q}, \boldsymbol{x}}$ acute vs obtuse

- Acute $\widehat{\boldsymbol{q}, \boldsymbol{x}} \in [0, \frac{\pi}{2})$
  - $\boldsymbol{q} \cdot \boldsymbol{x} > 0$ and $[\boldsymbol{q} \cdot \boldsymbol{x}]_+ = \boldsymbol{q} \cdot \boldsymbol{x}$
  - $\Pr[\text{sign}(\boldsymbol{w} \cdot \boldsymbol{q}) = \text{sign}(\boldsymbol{w} \cdot \boldsymbol{x})] = 1 - \frac{\widehat{\boldsymbol{q}, \boldsymbol{x}}}{\pi} > \frac{1}{2}$
  - $\Pr[\Phi_{\boldsymbol{w}}(\boldsymbol{q}) \cdot \Phi_{\boldsymbol{w}}(\boldsymbol{x}) = [\boldsymbol{q} \cdot \boldsymbol{x}]_+ = \boldsymbol{q} \cdot \boldsymbol{x}] > \frac{1}{2}$

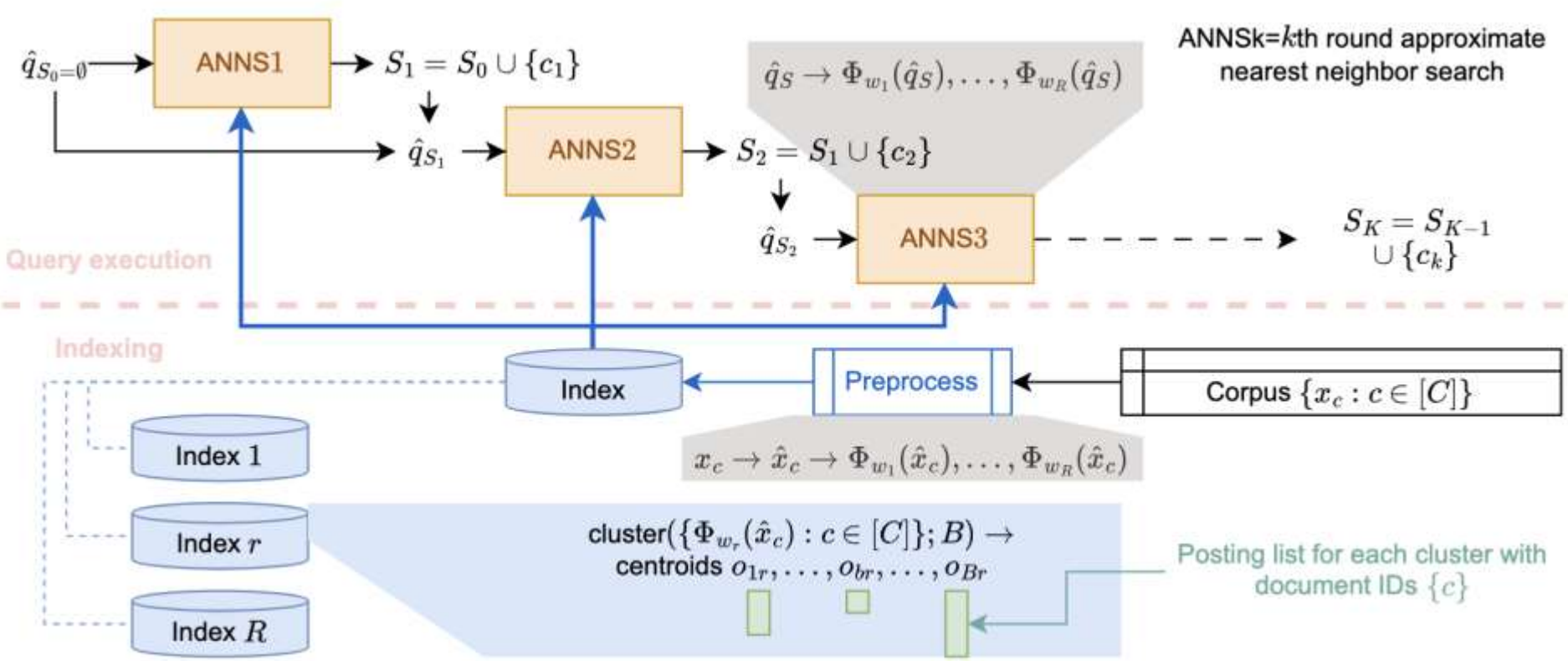
Two cases: angle $\widehat{q,x}$ acute vs obtuse

- Acute $\widehat{q,x} \in [0, \frac{\pi}{2})$
  - $q \cdot x > 0$ and $[q \cdot x]_+ = q \cdot x$
  - $\Pr[\text{sign}(w \cdot q) = \text{sign}(w \cdot x)] = 1 - \frac{\widehat{q,x}}{\pi} > \frac{1}{2}$
  - $\Pr[\Phi_w(q) \cdot \Phi_w(x) = [q \cdot x]_+ = q \cdot x] > \frac{1}{2}$
- Obtuse $\widehat{q,x} \in (\frac{\pi}{2}, \pi)$
  - $q \cdot x < 0$ and $[q \cdot x]_+ = 0$
  - $\Pr[\text{sign}(w \cdot q) = \text{sign}(w \cdot x)] = 1 - \frac{\widehat{q,x}}{\pi} < \frac{1}{2}$
  - $\Pr[\Phi_w(q) \cdot \Phi_w(x) = [q \cdot x]_+ = 0] > \frac{1}{2}$

Amplify by sampling R random vecs $\boldsymbol{w}_{1:R}$

- These two events are equivalent
  - $\exists r \in [R]$: $\Phi_{w_r}(\boldsymbol{q}) \cdot \Phi_{w_r}(\boldsymbol{x}) = [\boldsymbol{q} \cdot \boldsymbol{x}]_+$
  - $\max_{r \in [R]} \Phi_{w_r}(\boldsymbol{q}) \cdot \Phi_{w_r}(\boldsymbol{x}) = [\boldsymbol{q} \cdot \boldsymbol{x}]_+$
- Probability max over replicas fails to compute $[\boldsymbol{q} \cdot \boldsymbol{x}]_+$ is $< 2^{-R}$
- Convert each doc $\boldsymbol{x} \in X_c$ to $\widehat{\boldsymbol{x}}$; compute $\Phi_{\boldsymbol{w}_{1:R}}(\widehat{\boldsymbol{x}})$, build R replicas
- Given query Q, start with empty $S_0 = \emptyset$
- In each round $k \in [K]$
  - For each query word, compute $\widehat{\boldsymbol{q}}_{S_{k-1}}$, probe indexes, merge, prune
  - Choose next c_k from pruned doc set
  - $S_k \leftarrow S_{k-1} \cup c_k$ and go on to next round

Dense Index for Set Coverage (DISCo)

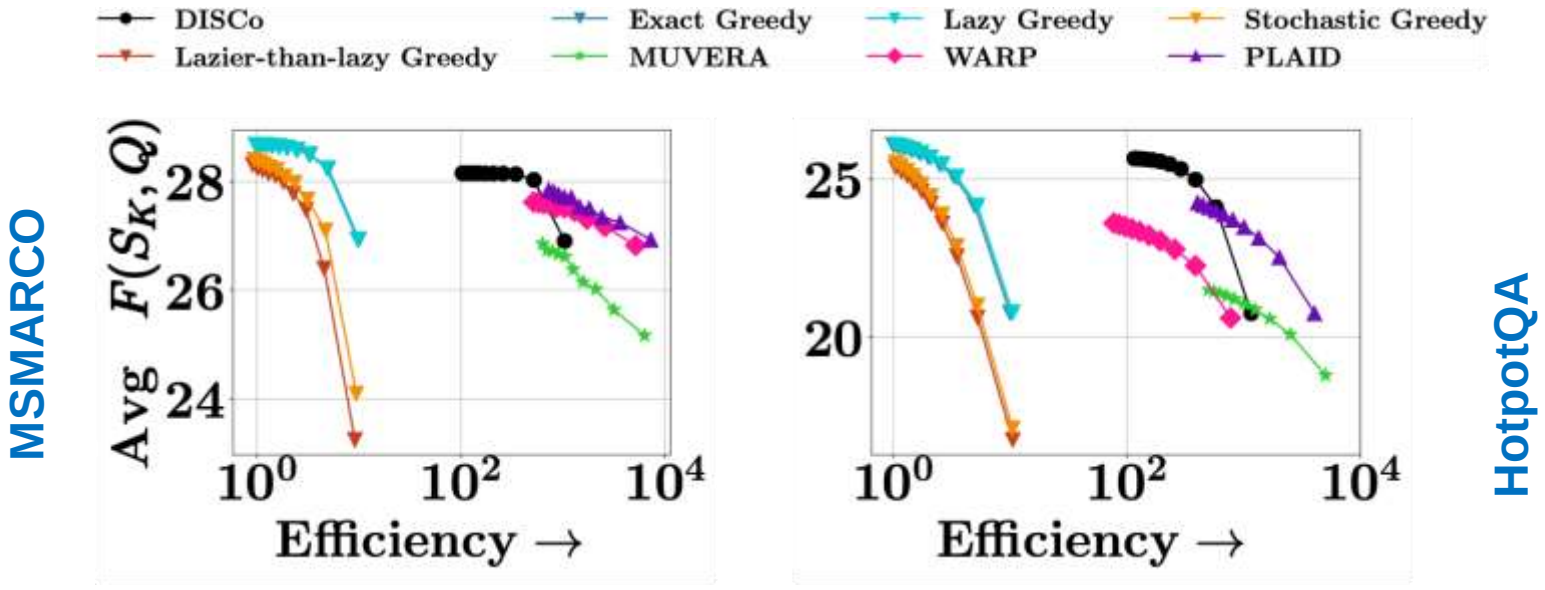


Baselines

- **Exact Greedy** – Exhaustively evaluates every candidate's marginal gain at each step for strong coverage, but is slow at scale.
- **Lazy Greedy** – Uses a priority queue to avoid re-scoring all candidates, speeding up greedy via diminishing returns.
- **Stochastic Greedy** – Samples a random subset of candidates per round to accelerate retrieval with a tunable accuracy trade-off.
- **Lazier-than-Lazy Greedy** – Heapifies a sampled subset to combine lazy and stochastic speed-ups, with implementation-dependent performance.
- **PLAID** – Fast independent MaxSim retrieval via multi-stage pruning and IVF-style indexing for large CPU/GPU latency reductions.
- **MUVERA** – Compresses each passage and query into a single vector that approximates MaxSim, enabling ANN-based top-K retrieval.
- **WARP** – Accelerates independent retrieval by pruning token pairs and using compressed residuals with highly optimized runtimes.

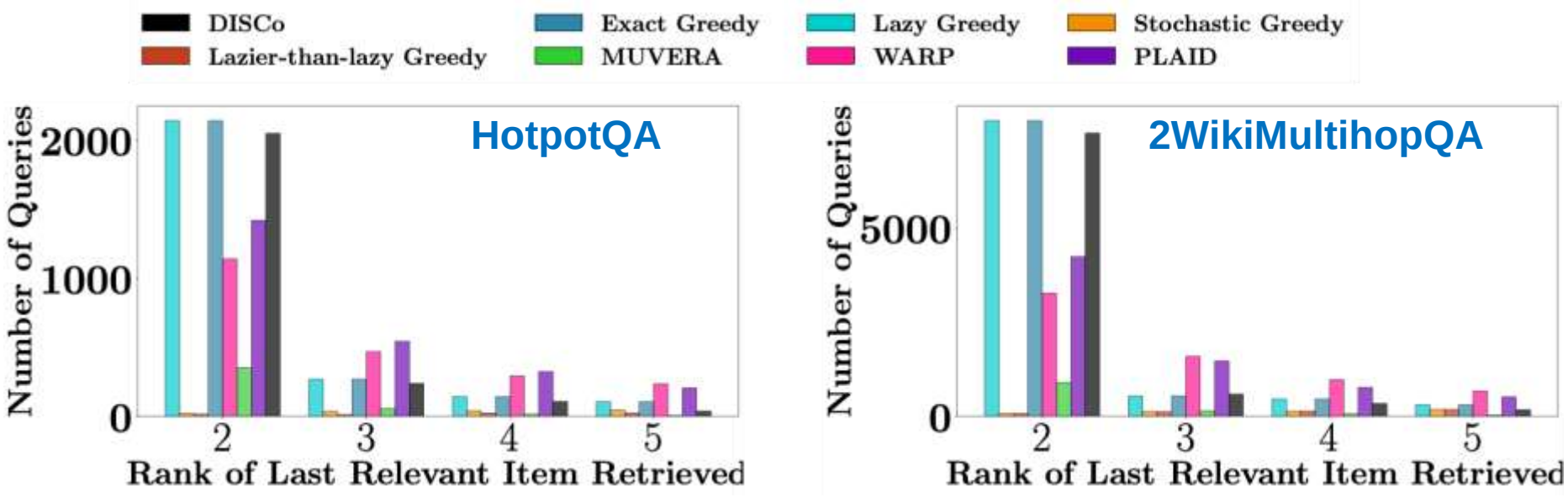
Set utility vs query throughput

- Exact Greedy gives optimal achievable solution, very slowly.
- Compare methods based on the coverage scores $F(S_K|Q)$ v/s efficiency = time taken by Exact Greedy divided by time taken by various other methods (larger = more efficient)

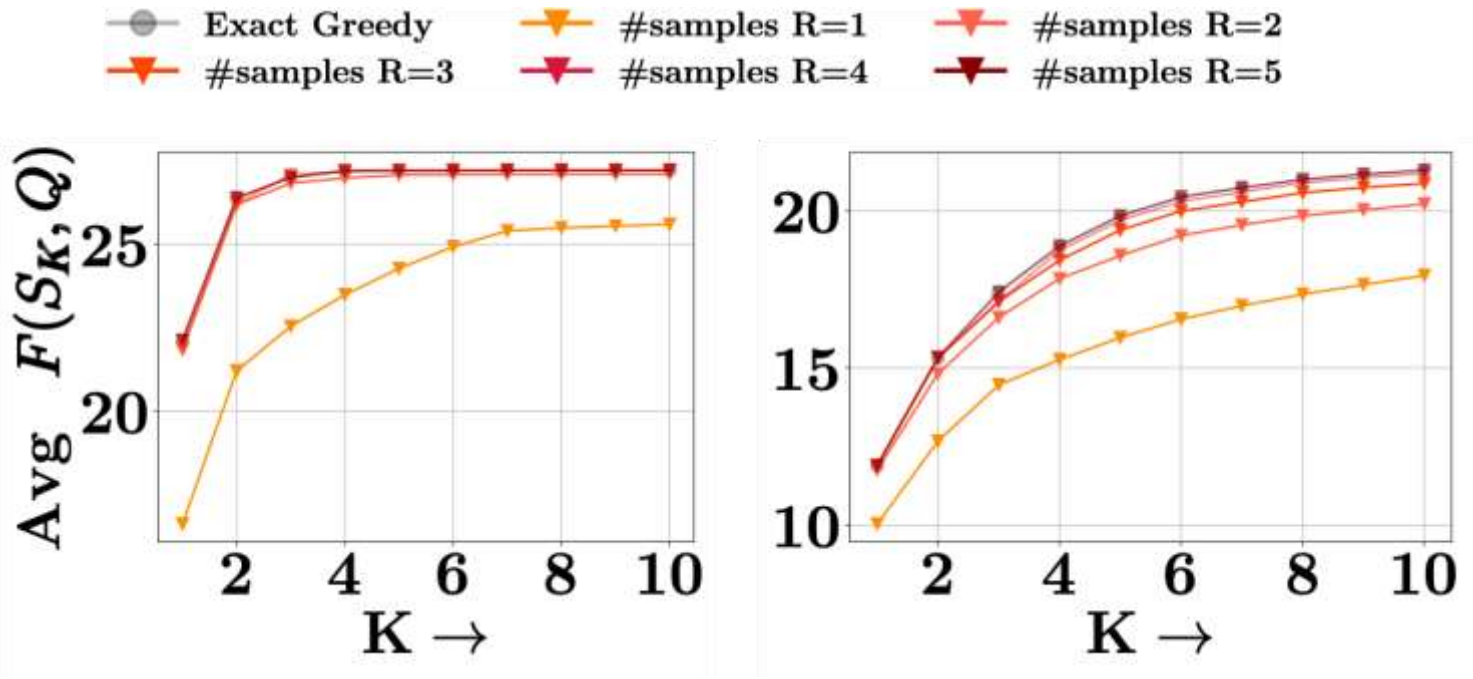


Multi-hop QA datasets (gold set utility)

- Nested output sets $\emptyset = S_0 \subset \cdots \subset S_K \subset \cdots$
- At what K are all doc items necessary to infer answer collected (by various systems)?



Impact of number of replicas, R

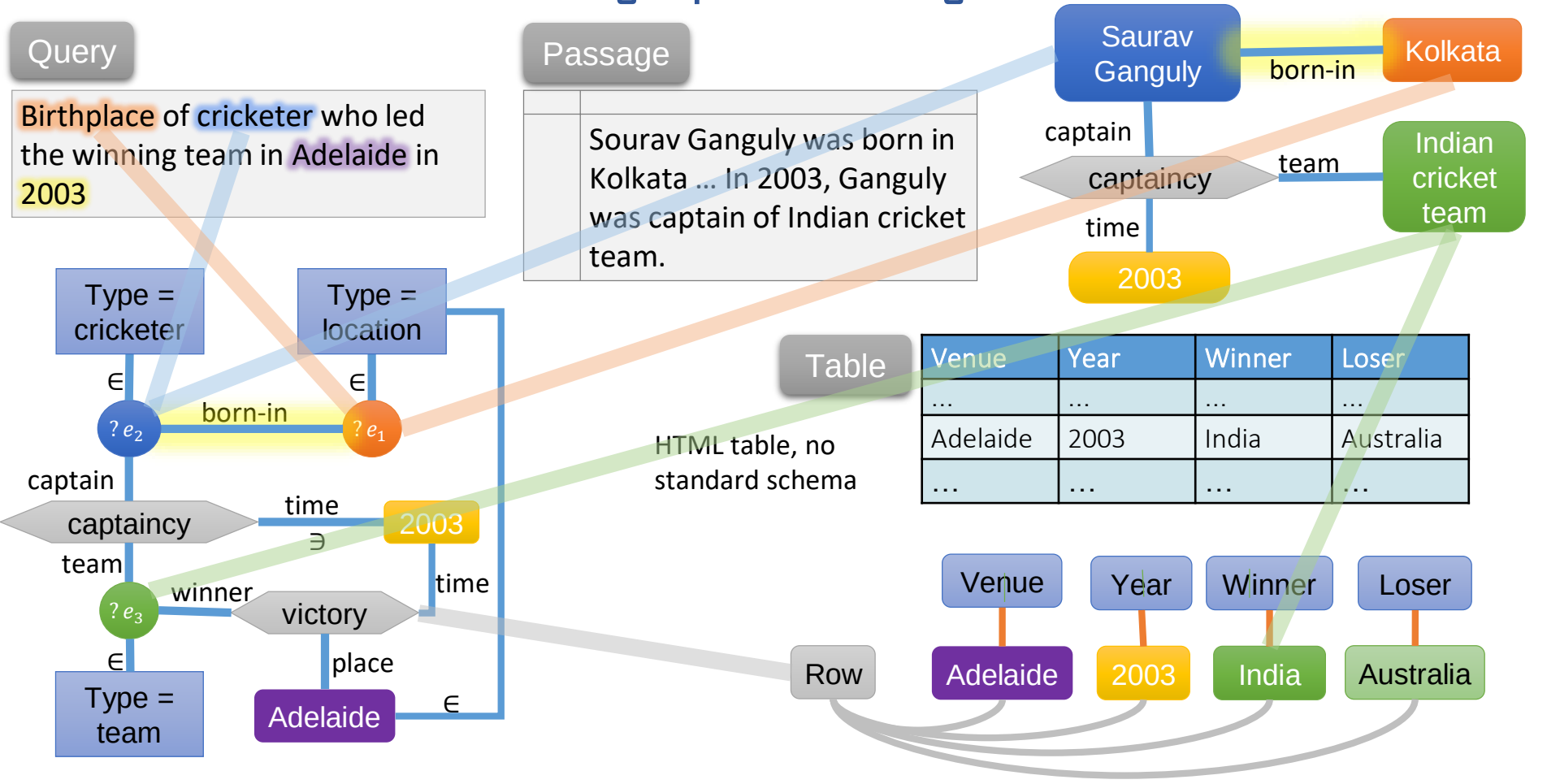


NFCorpus

Scifact

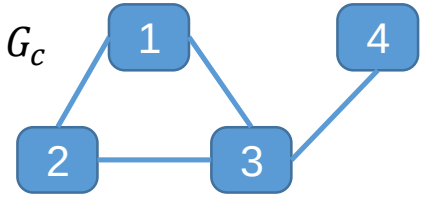
Subset utility with varying K and number of index replicas R

Collective search as (sub)graph matching

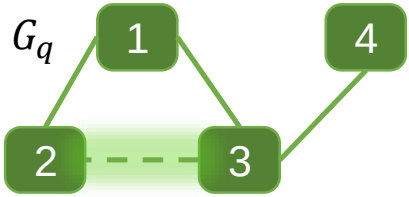


Subgraph isomorphism search

- Graphs G_q, G_c on same number of nodes N (via padding)
- If G_q is a subgraph of G_c, then ∃ node permutation π: $[N] \overset{1:1}{\leftrightarrow} [N]$ st $(i,j) \in E_q \Longrightarrow \big(\pi(i), \pi(j)\big) \in E_c$
- Adjacency matrices A_q, A_c
- π induces permutation matrix Π
- Subgraph isomorphic means $\Pi A_q \Pi^\intercal \leq A_c$ (elementwise)
- Searching for Π is computationally difficult (QAP)
- Can neural graph representation help?



A_c	1	2	3	4
1	0	1	1	0
2	1	0	1	0
3	1	1	0	1
4	0	0	1	0



A_q	1	2	3	4
1	0	1	1	0
2	1	0	0	0
3	1	0	0	1
4	0	0	1	0

Neural (sub)graph matching and search

- Is query graph a *subgraph* of corpus graph?
  [AAAI 2022, ICLR 2025]
- Is query graph isomorphic to a *connected* subgraph of corpus graph? [NeurIPS 2022]
- Estimating the *size of a maximal clique* in a graph
  [ICLR 2025]
- *Edit distance* between query and corpus graph
  [NeurIPS 2024]
- Embedding *dimensions are exchangeable* → fast search
  [ICLR 2026 ]

Inverted index for subgraph search

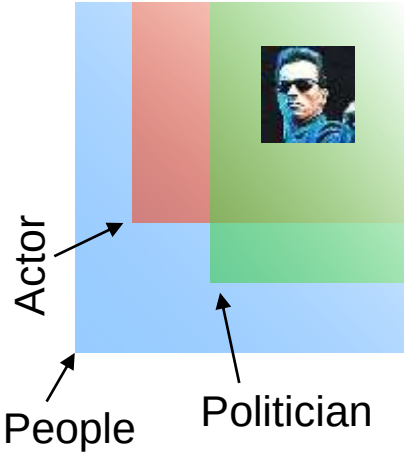
- (Sparse) inverted indices have string tokens as keys instead of vector
  - Space and query time intensively optimized over 20 years
  - Far more efficient than dense (vector) index
- How to create a fake document with 'words' from embeddings provided by a graph neural network?
- Contextual representation for graph inverted indices (CoRGII) [NeurIPS 2025]

Other intriguing retrieval problems

- Back to single vector corpus item (doc) $\boldsymbol{x} \in \mathbb{R}^D$
- Query is also a single vector $\boldsymbol{q} \in \mathbb{R}^D$
- But, asymmetric hinge 'distance'

$$d(q,x) = \sum_{i\in[D]}\left[q[i] - x[i]\right]_+ = \|[\boldsymbol{q} - \boldsymbol{x}]_+\|_1$$

- If $\boldsymbol{q} \leq \boldsymbol{x}$ (elementwise) then distance is zero
- Useful for set containment ($q \subseteq x$) search
  - Text entailment, partial order learning [SIGIR 2018]
- Find K of C corpus items with min hinge distance in $o(C)$ time → FourierHashNet [NeurIPS 2023 🔦]



Conclusion and outlook

- New age of dense retrieval / "vector DB"
- Compression to single vector easy but lossy
- Real applications benefit from composite abstractions: sets, sequences, graphs, . . .
- Distances and similarities that are faithful to applications are also algorithmically more difficult to index
- Challenges at the confluence of ML algorithms and practice

Ask not what LLMs can do for you; ask what you can do for LLMs