RE-THINKING RE-RANKING

Sean MacAvaney
University of Glasgow

Presented at:
Search Solutions 2025

Terrier

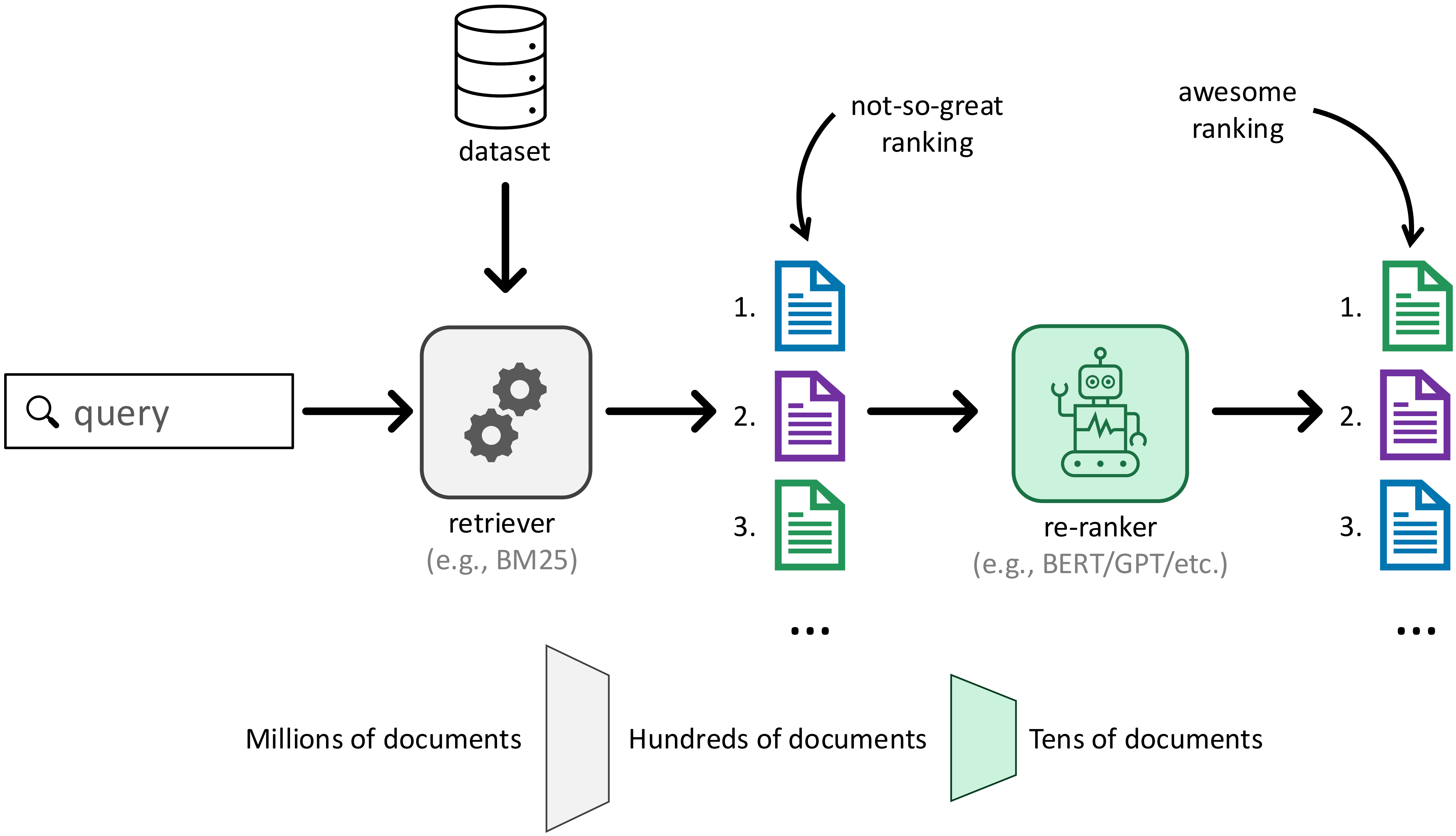
University of Glasgow

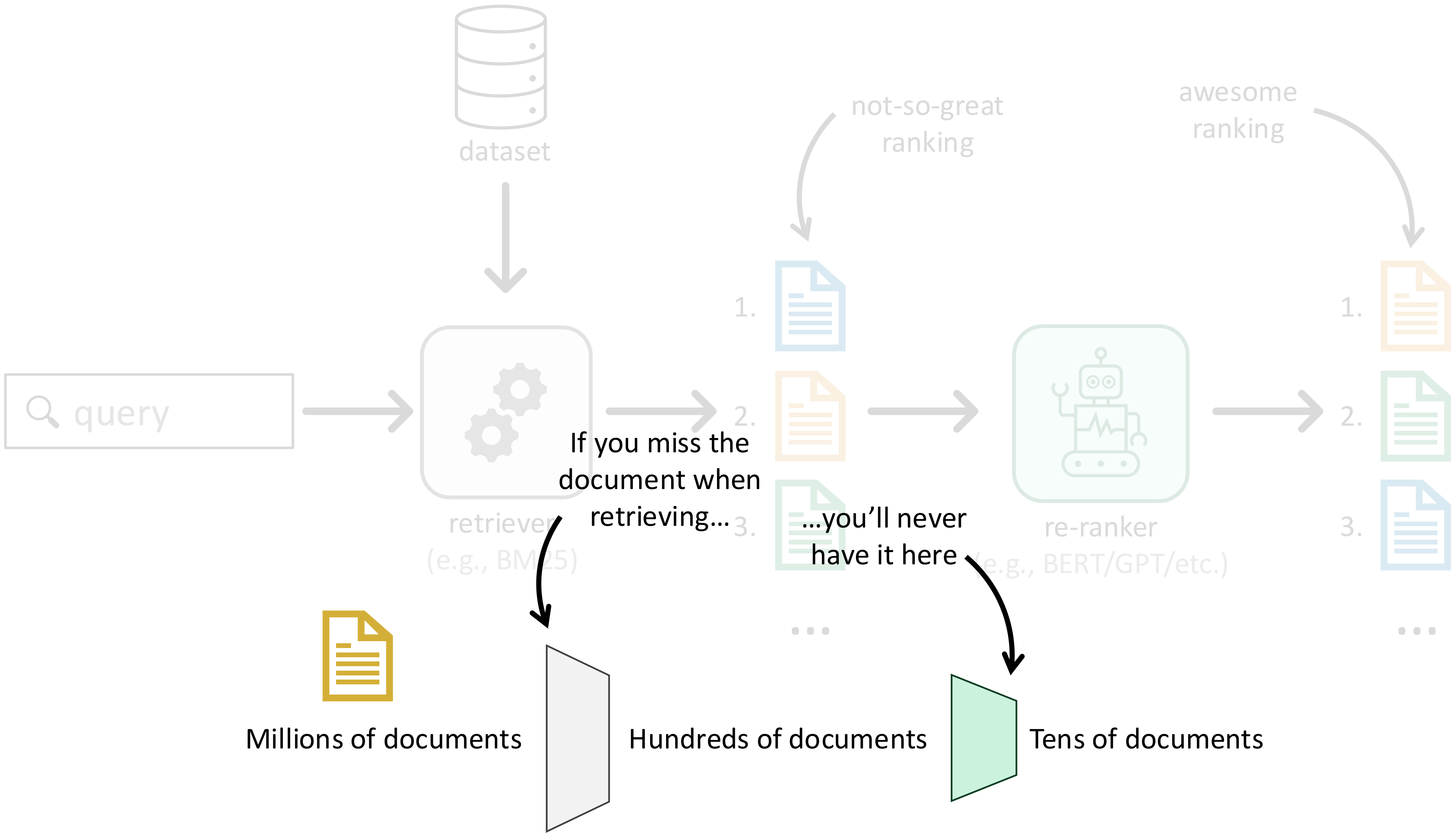
Sean MacAvaney
@macavaney

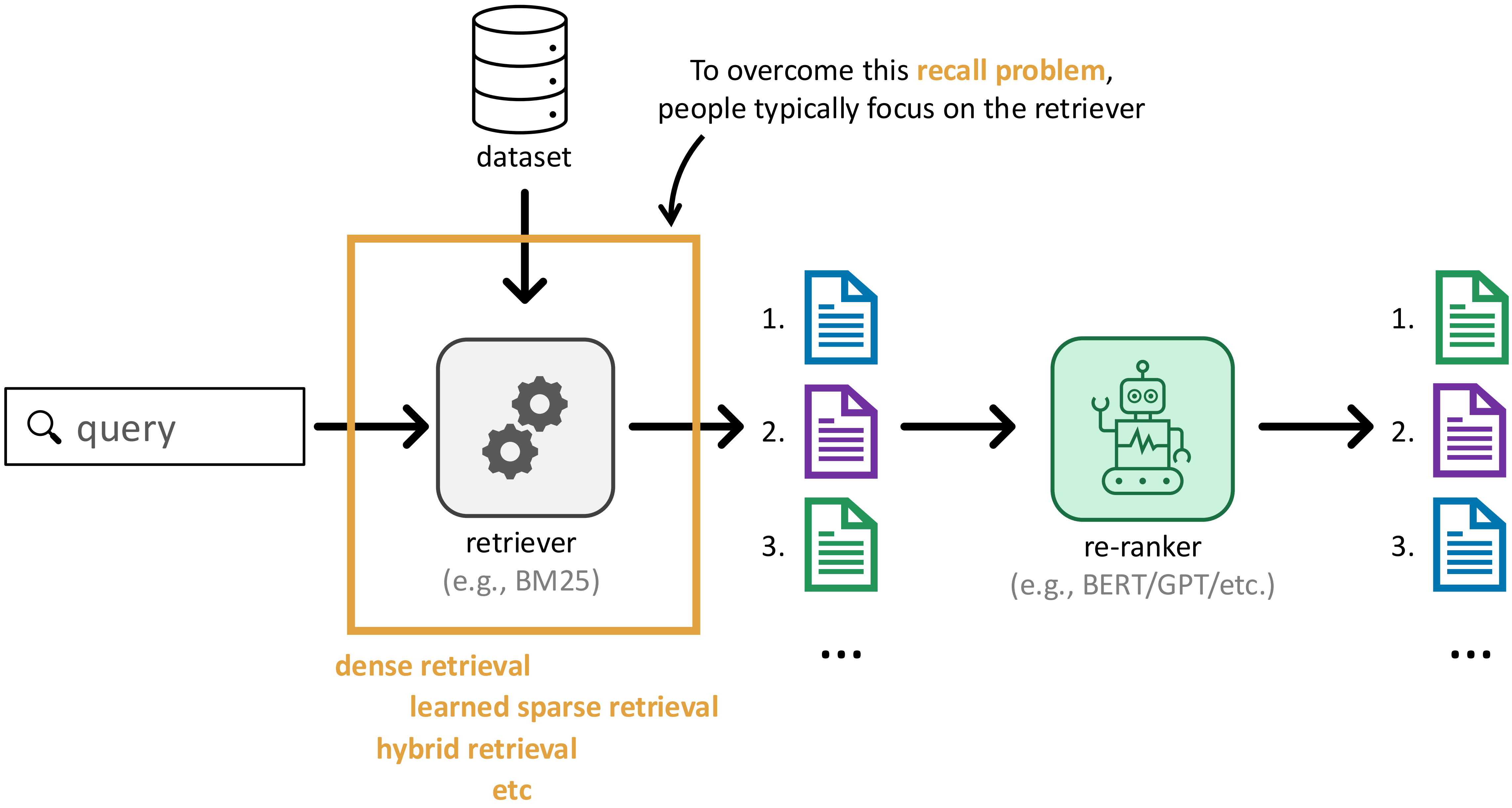
University of Glasgow · Senior Lecturer

Conduct practical research in information retrieval:
- Learned Sparse Expansion
- Search Result Evaluation using LLMs
- Document Quality Prediction
- **Adaptive Re-Ranking**

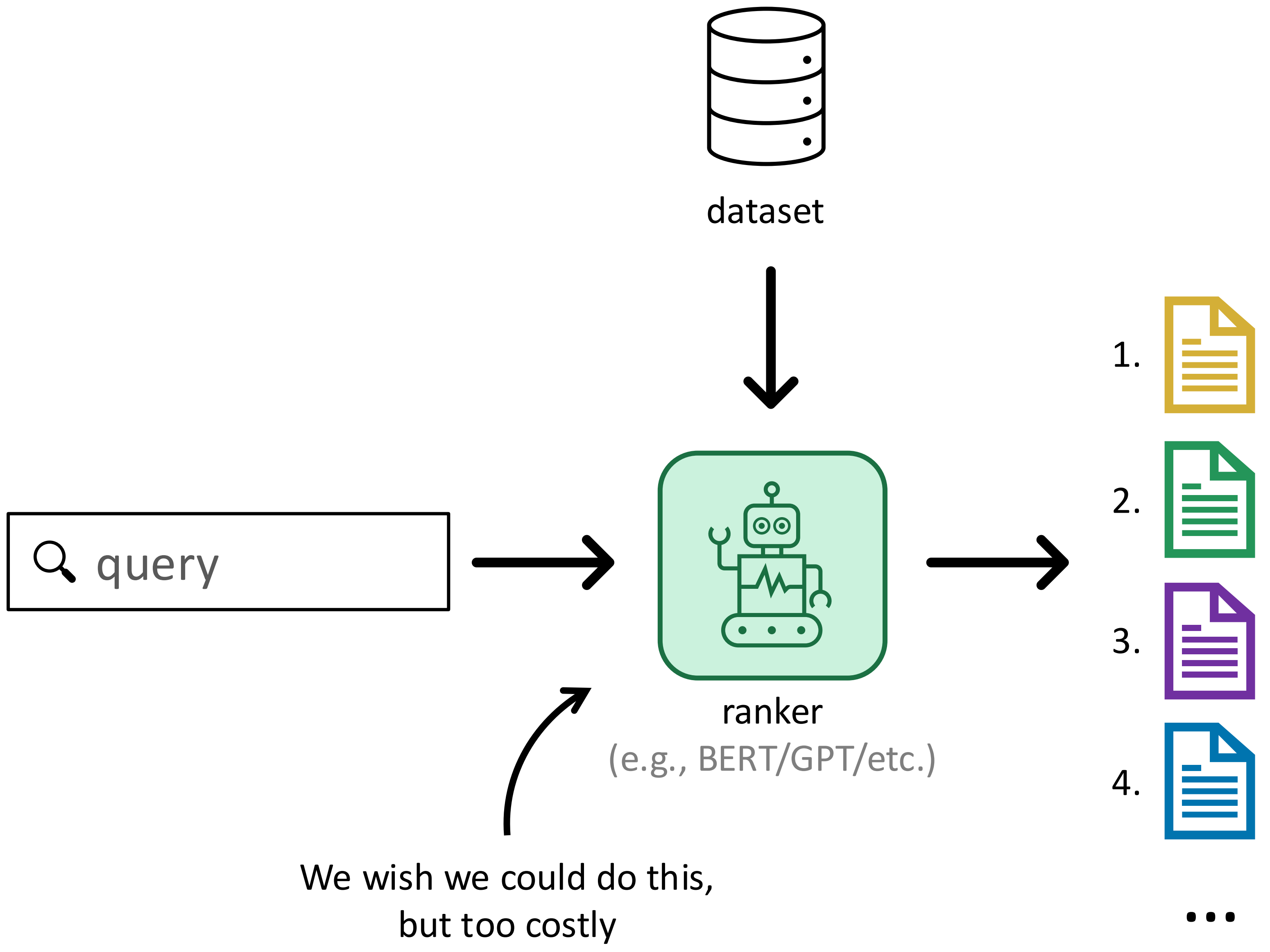
What’s Re-Ranking?

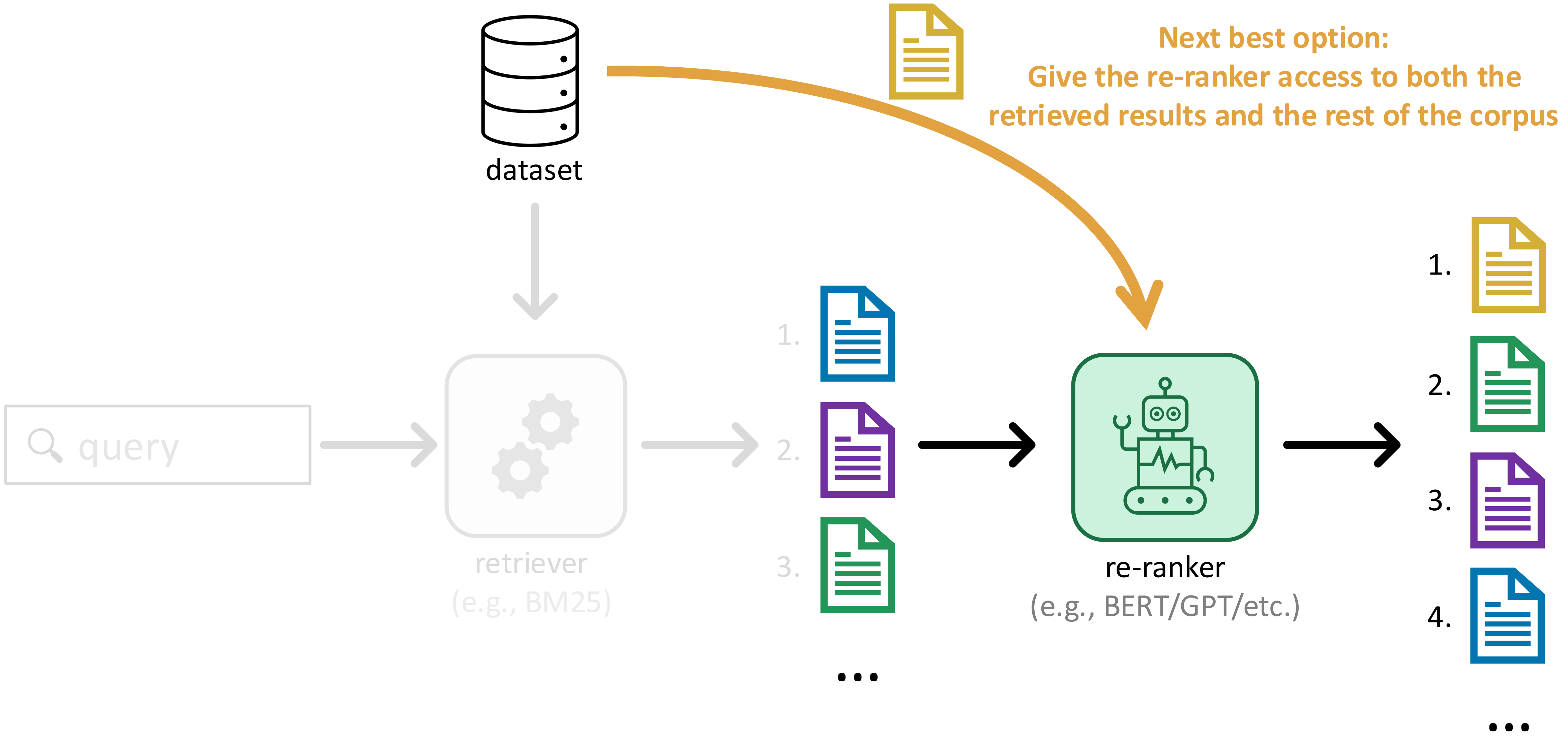

dataset
query
retriever
(e.g., BM25)
not-so-great ranking
1.
2.
3.
...
re-ranker
(e.g., BERT/GPT/etc.)
awesome ranking
1.
2.
3.
...
Millions of documents
Hundreds of documents
Tens of documents


dataset
not-so-great ranking
awesome ranking
query
1.
2.
3.
...
retriever (e.g., BM25)
re-ranker (e.g., BERT/GPT/etc.)
If you miss the document when retrieving…
…you'll never have it here
Millions of documents
Hundreds of documents
Tens of documents

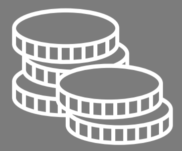


To overcome this recall problem,
people typically focus on the retriever
dataset
query
retriever
(e.g., BM25)
1.
2.
3.
...
re-ranker
(e.g., BERT/GPT/etc.)
1.
2.
3.
...
dense retrieval
learned sparse retrieval
hybrid retrieval
etc


dataset
query
ranker
(e.g., BERT/GPT/etc.)
1.
2.
3.
4.
...
We wish we could do this,
but too costly



Next best option:
Give the re-ranker access to both the
retrieved results and the rest of the corpus
dataset
query
retriever
(e.g., BM25)
1.
2.
3.
...
re-ranker
(e.g., BERT/GPT/etc.)
1.
2.
3.
4.
...



I’ll show this can be done with minimal cost.

I’ll show this can be done with minimal cost.

The idea can improve retrievers like ColBERT, too.

I’ll show this can be done with minimal cost.

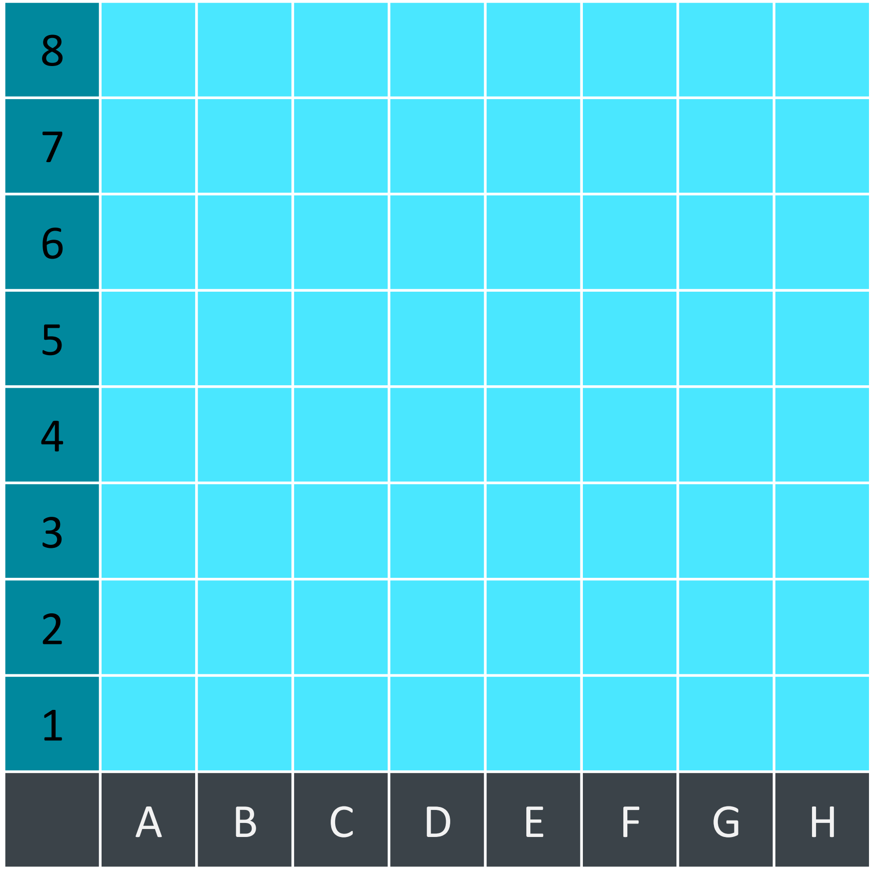
The idea can improve retrievers like ColBERT, too.

Ready-to-use with Open-Source tools!

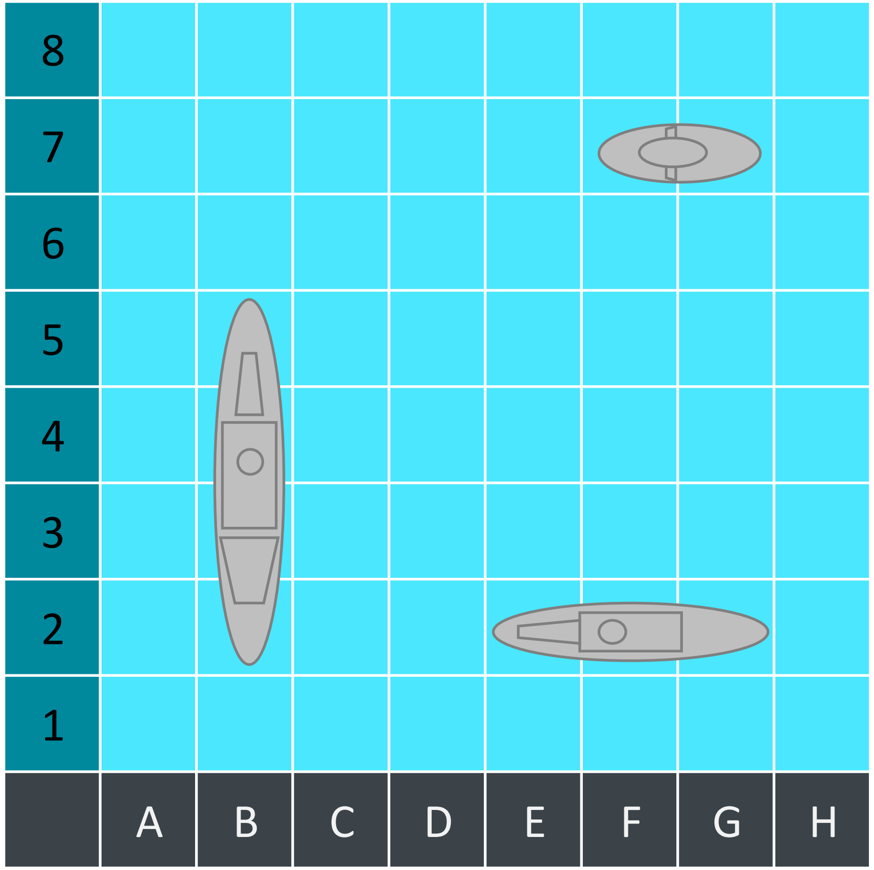
Battleship

Photograph By Pavel Ševela, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=18331432

Your Board

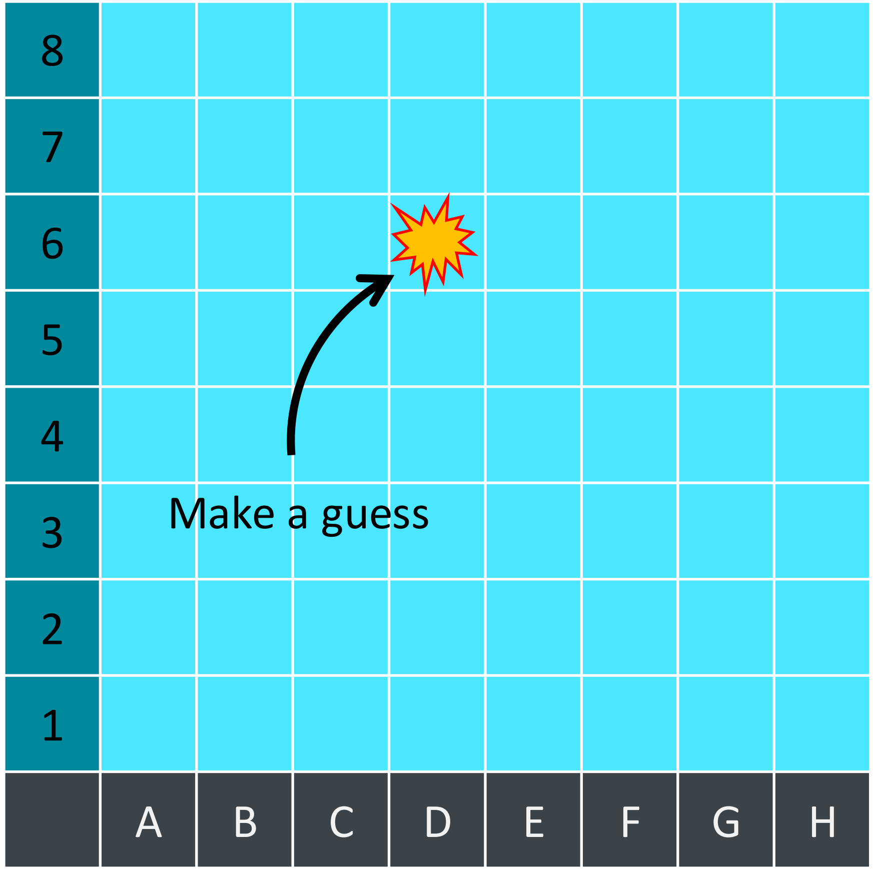


Opponent’s board (Secret)

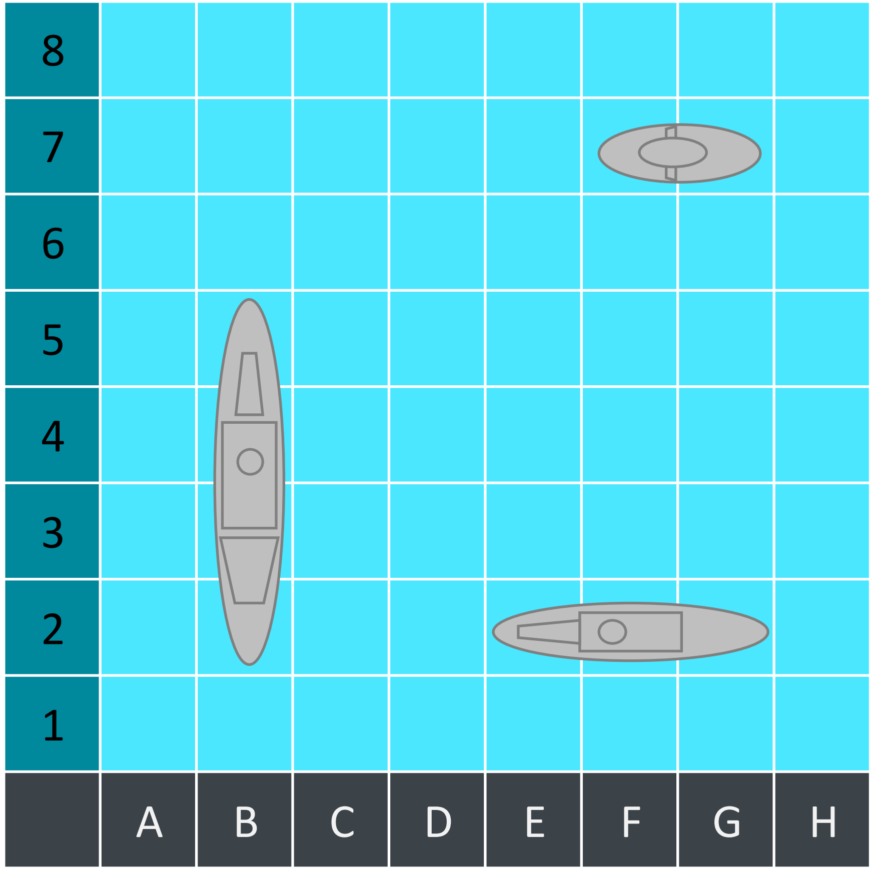


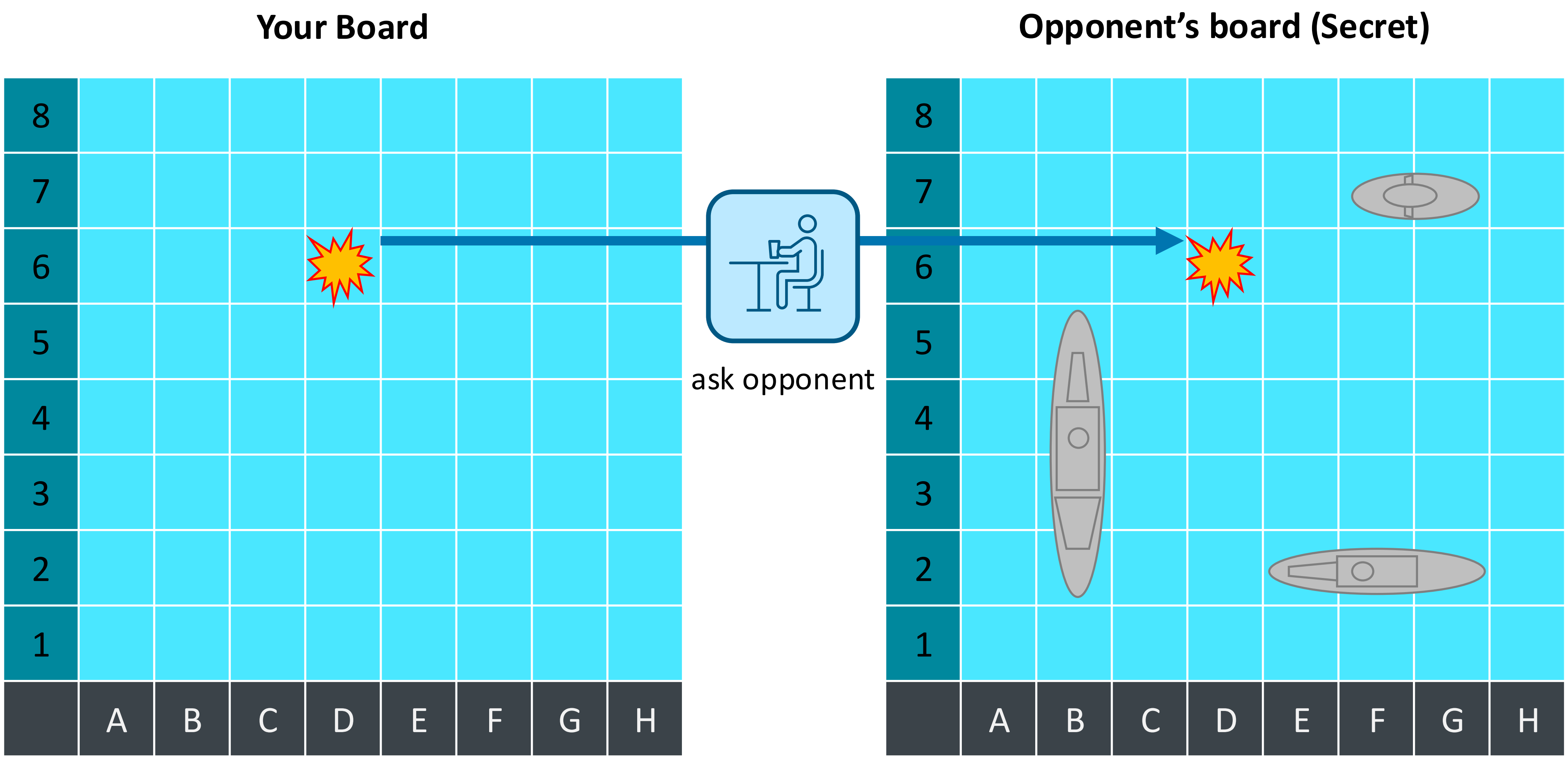
Goal: identify the positions of all your opponent’s ships

Your Board

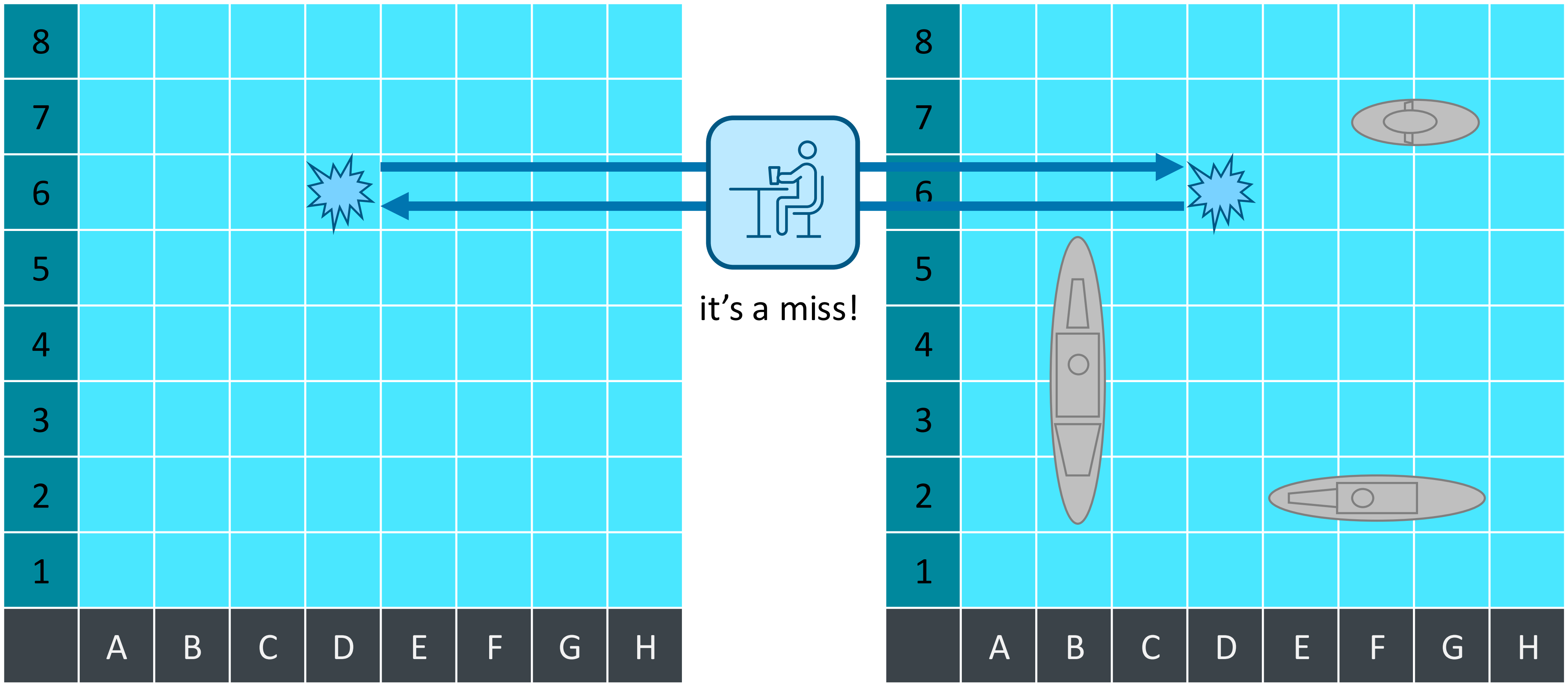


Opponent's board (Secret)

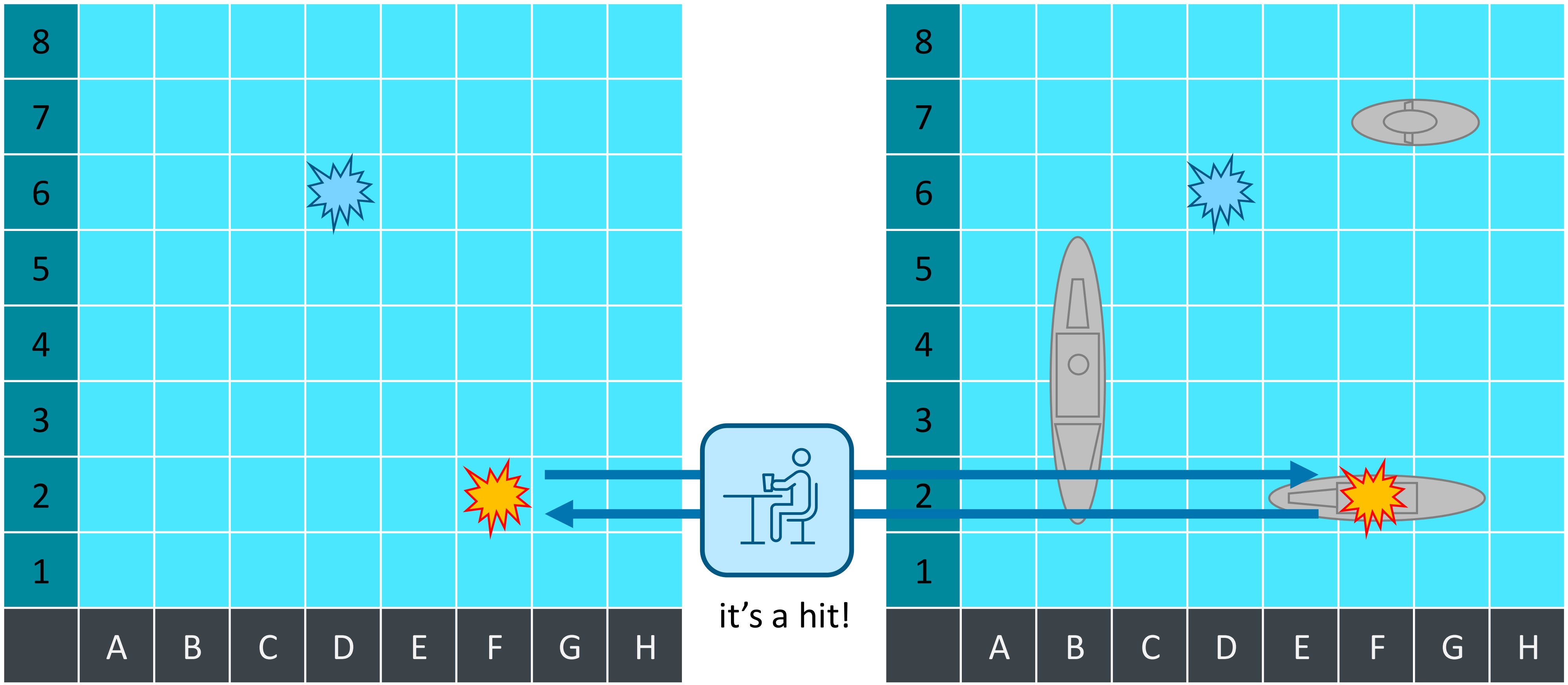



Your Board
Opponent's board (Secret)
ask opponent
8
7
6
5
4
3
2
1
A B C D E F G H

Your Board **Opponent’s board (Secret)**

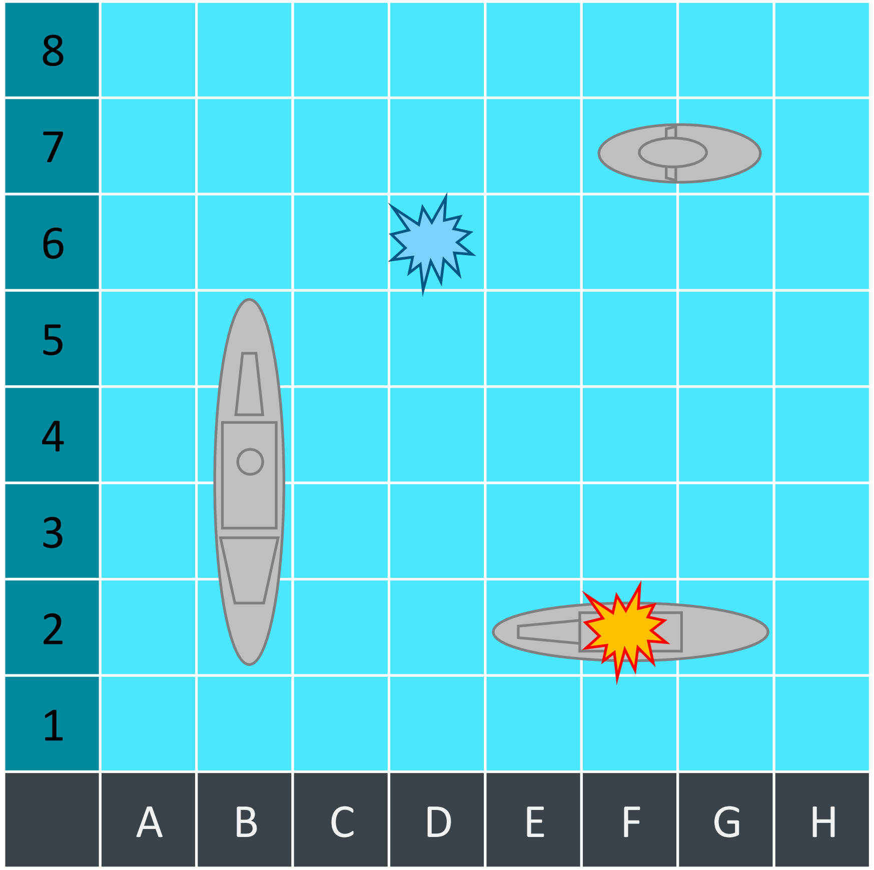


Your Board　　　　Opponent’s board (Secret)

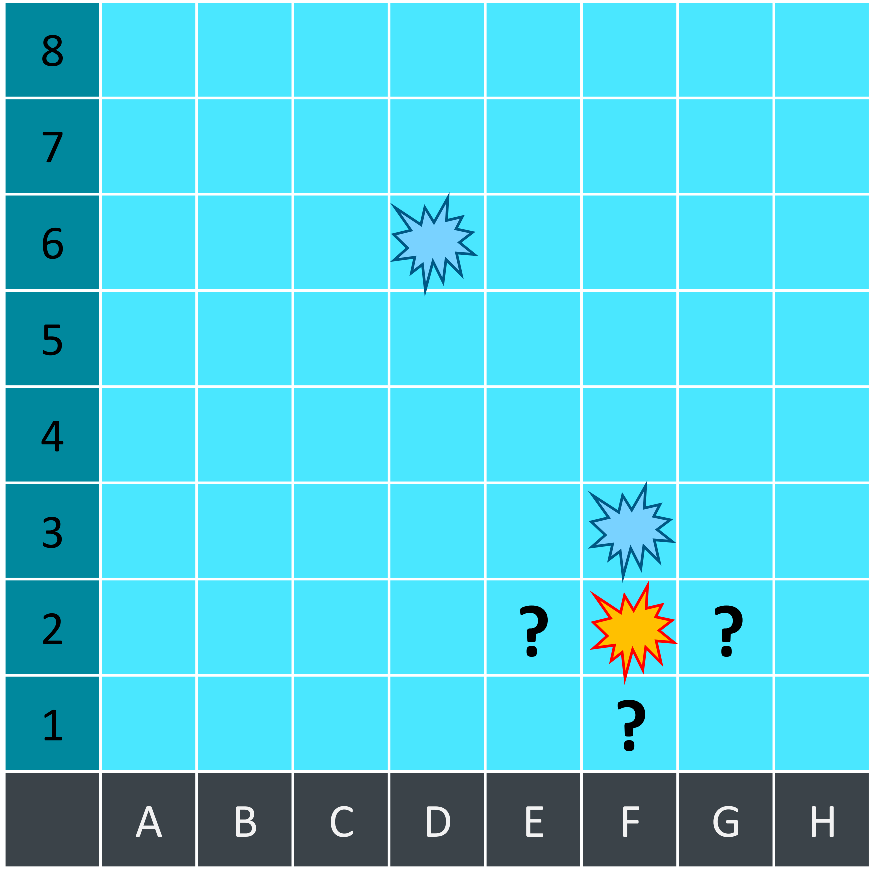


Your Board

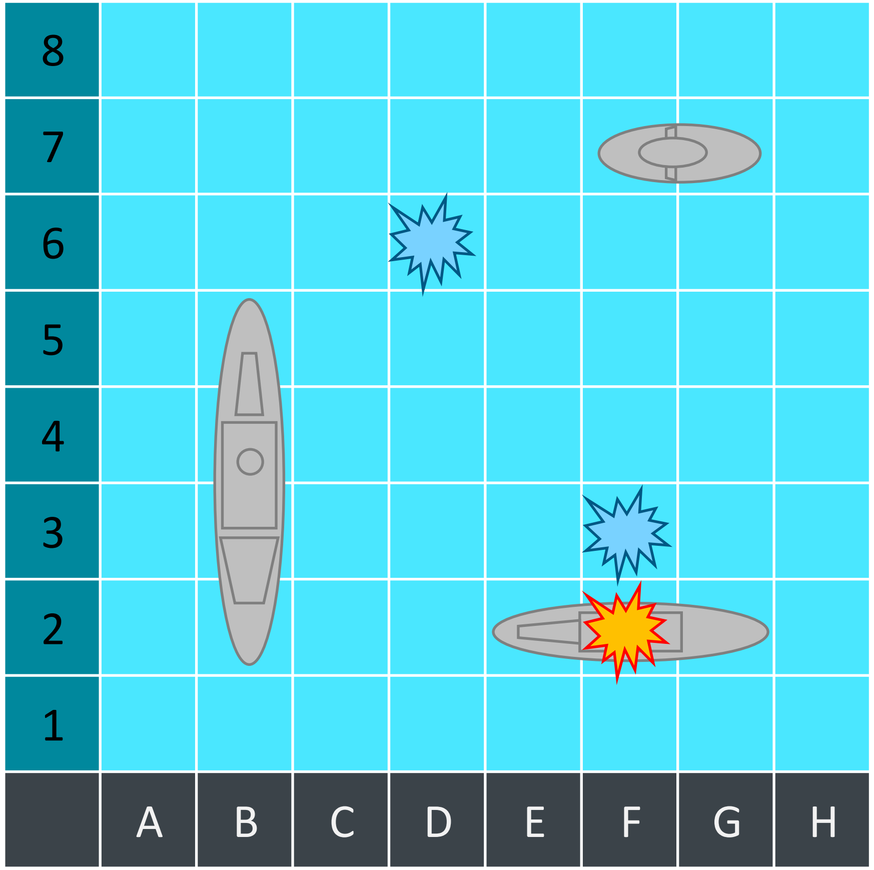
Opponent’s board (Secret)



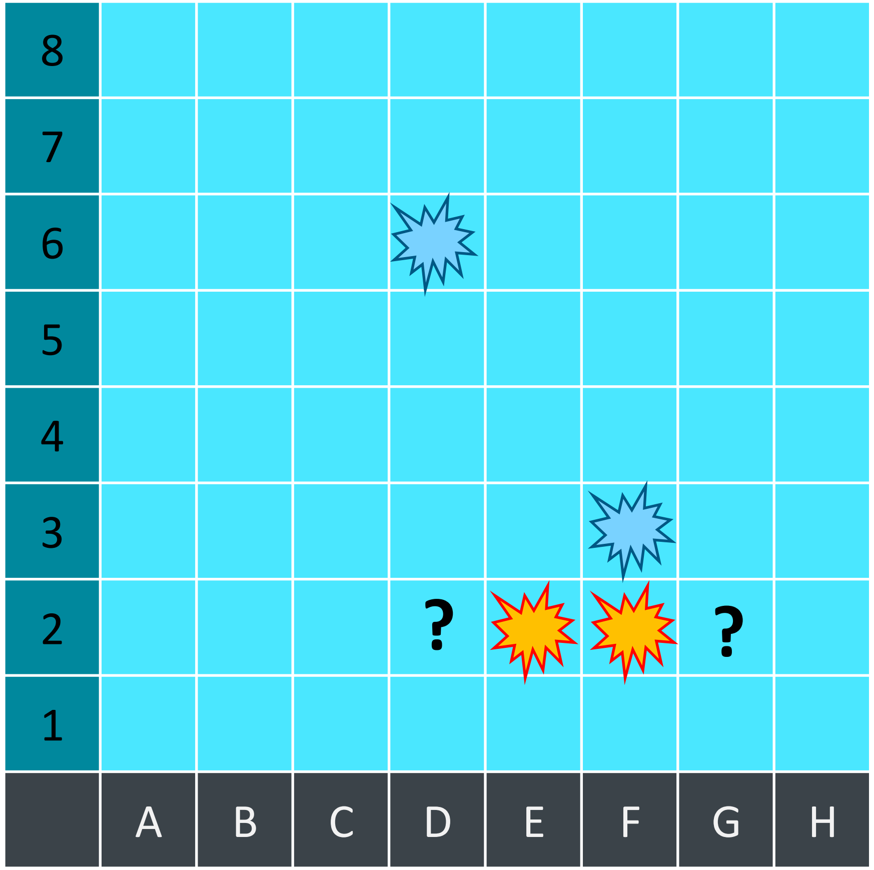
Your Board



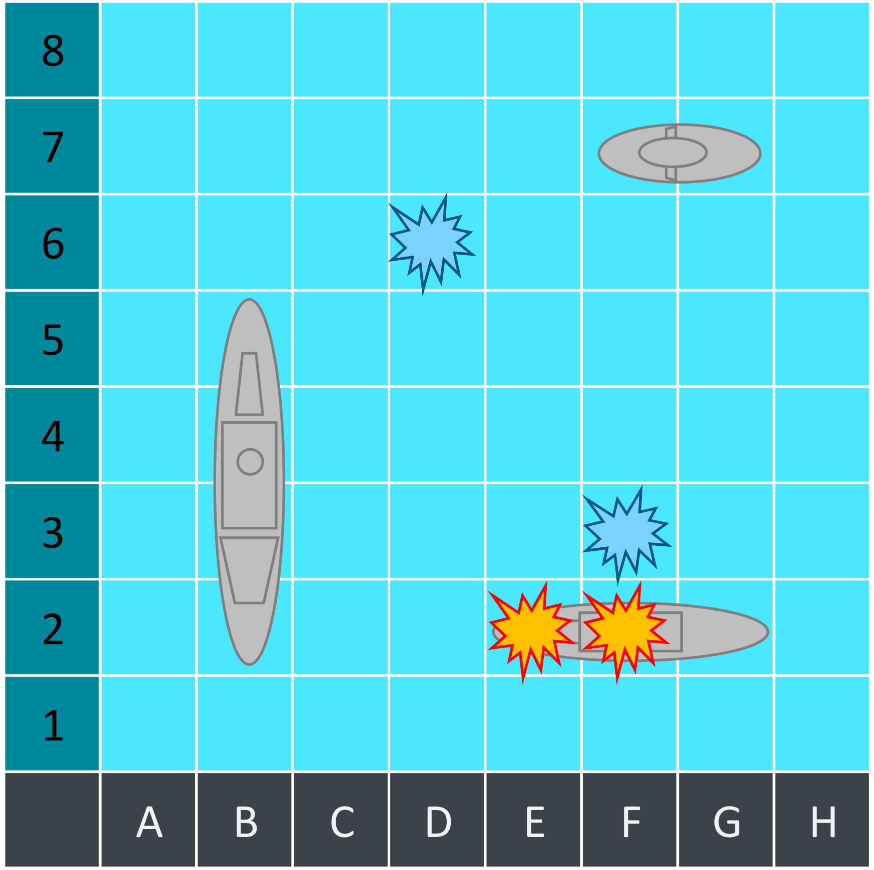
Opponent's board (Secret)



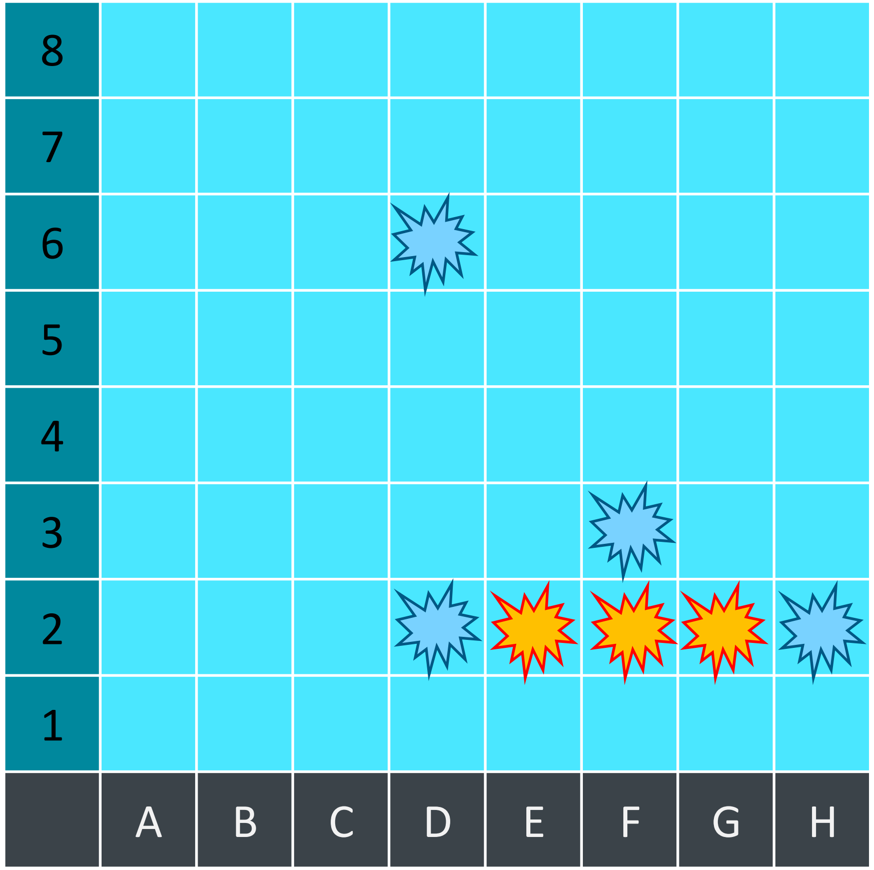
Your Board



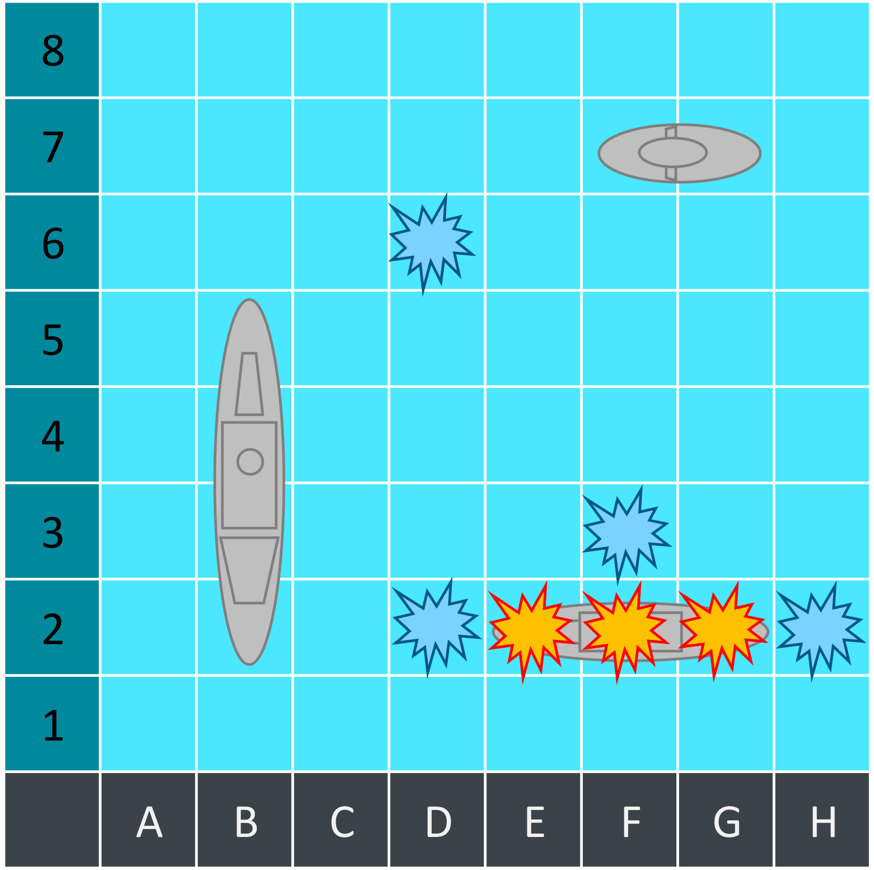
Opponent’s board (Secret)



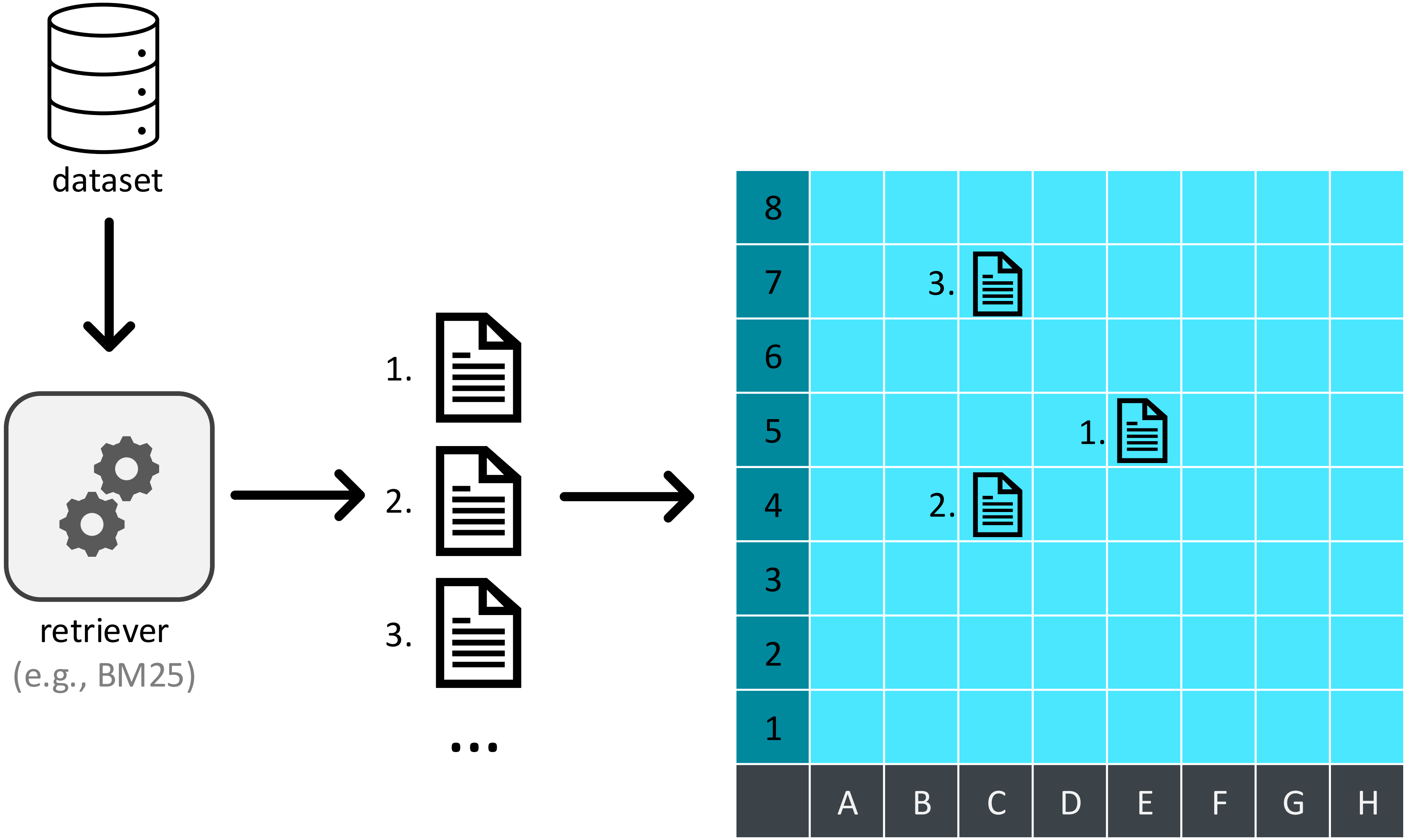
Your Board

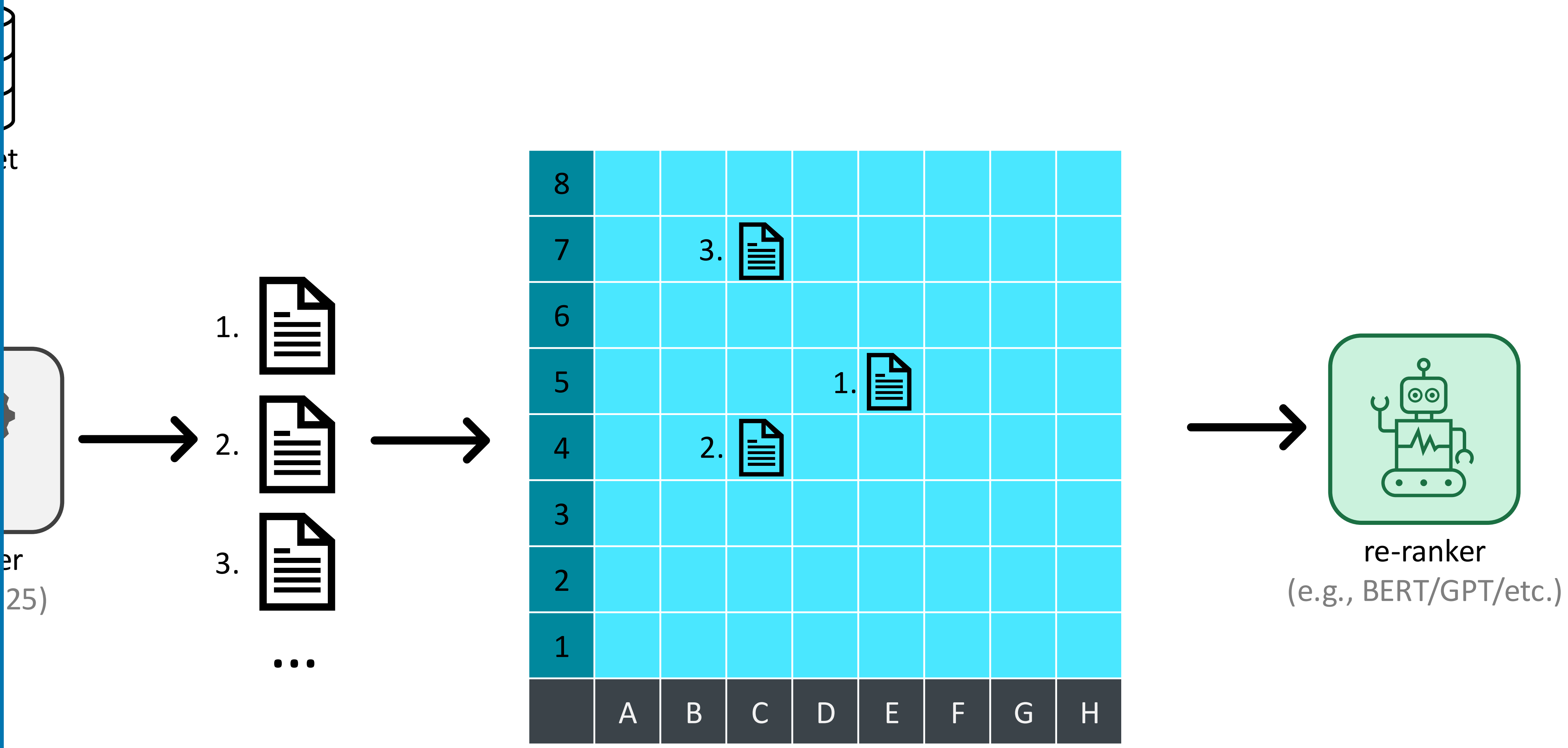


Opponent’s board (Secret)

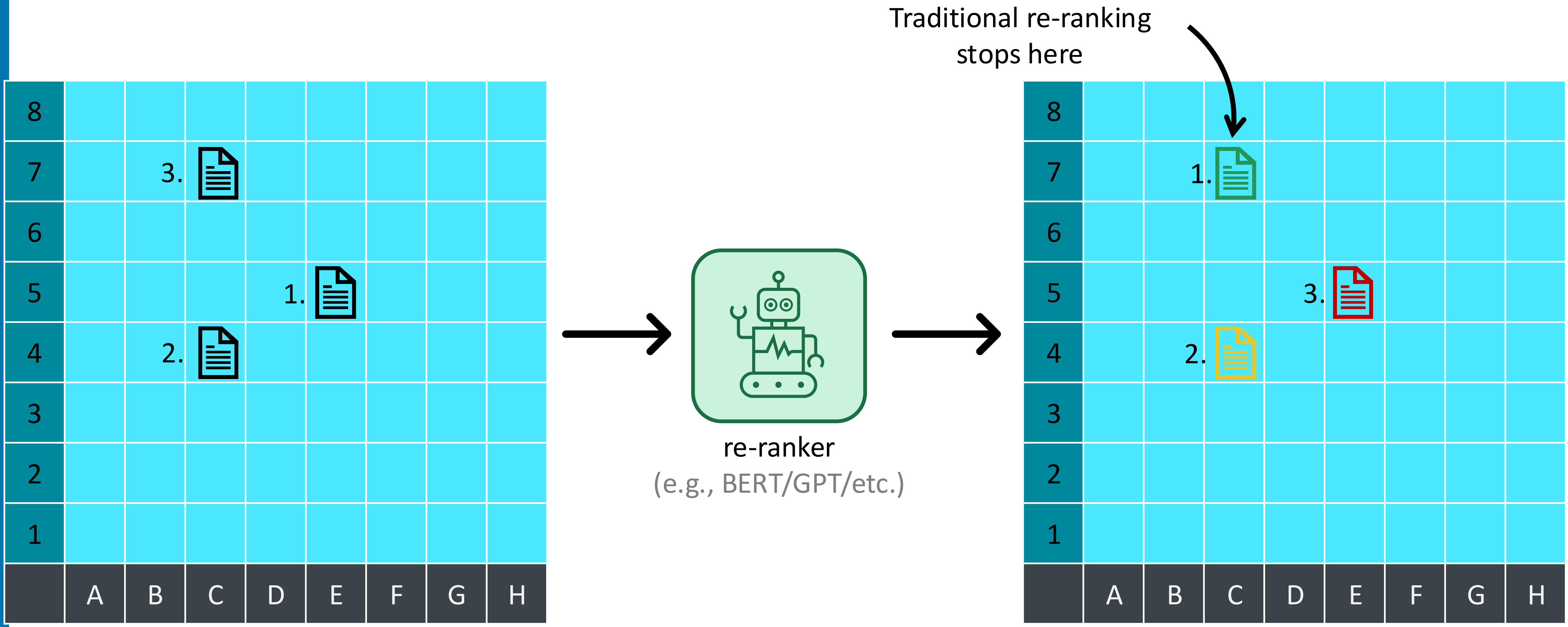


And so forth…

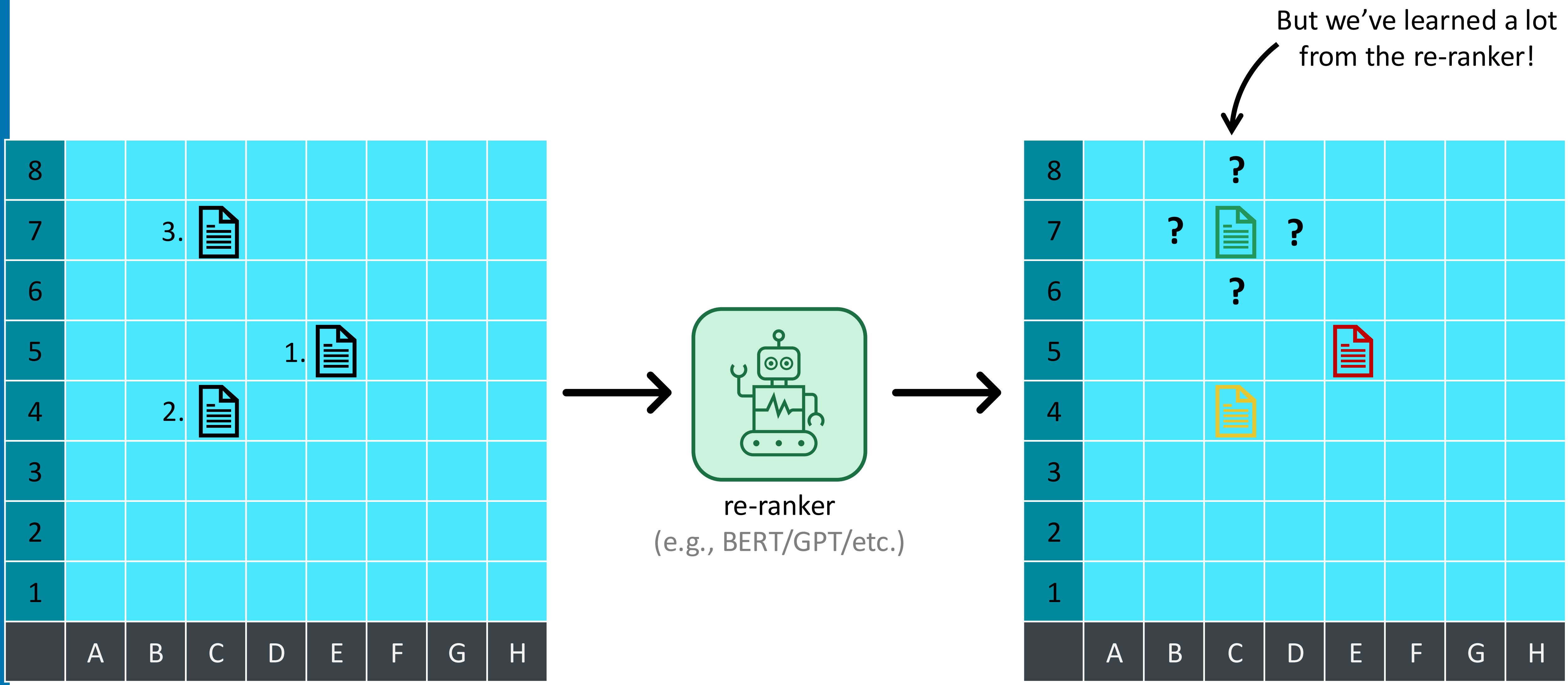

dataset
retriever
(e.g., BM25)
1.
2.
3.
...
8
7
6
5
4
3
2
1
A
B
C
D
E
F
G
H
3.
1.
2.



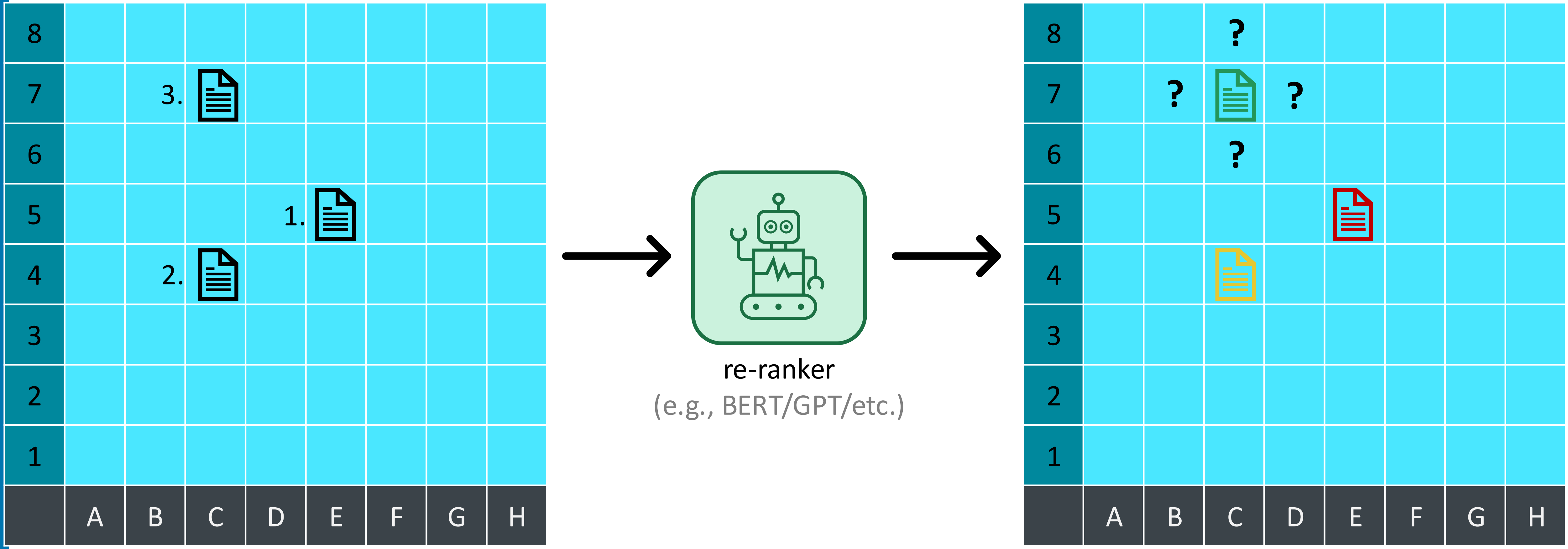
t
er
25)
1.
2.
3.
...
8
7
6
5
4
3
2
1
A
B
C
D
E
F
G
H
3.
1.
2.
re-ranker
(e.g., BERT/GPT/etc.)



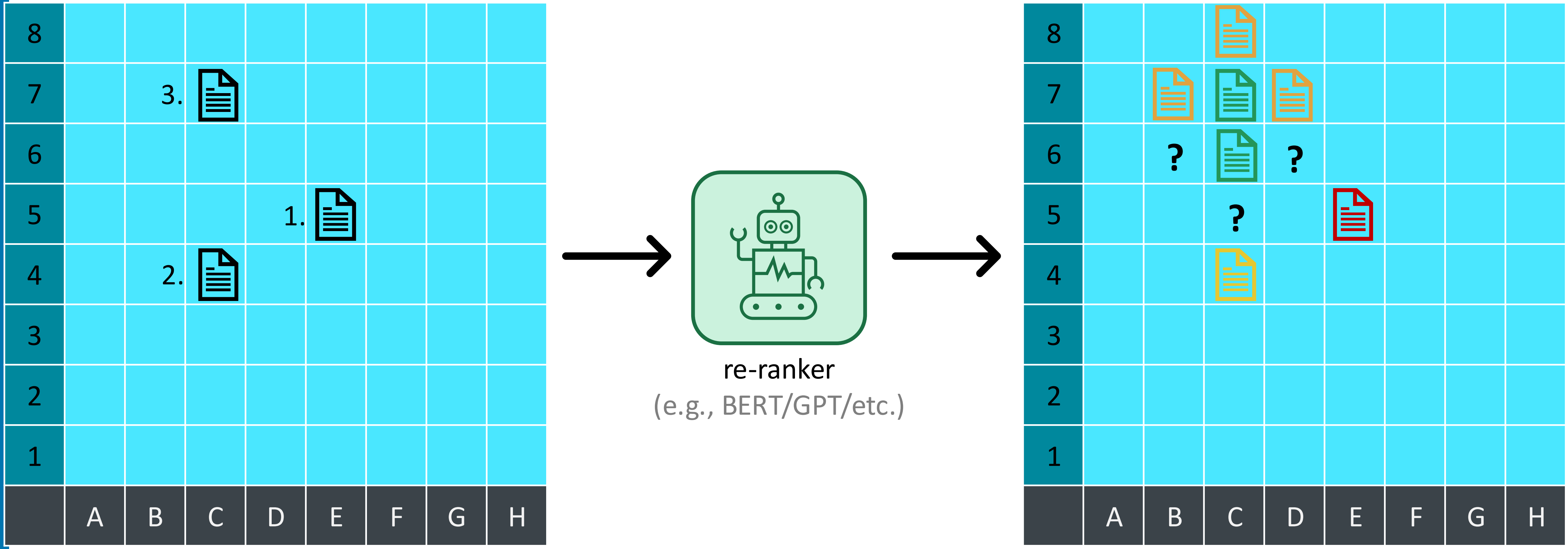
Traditional re-ranking
stops here
8
7
6
5
4
3
2
1
A
B
C
D
E
F
G
H
3.
1.
2.
re-ranker
(e.g., BERT/GPT/etc.)
1.
3.
2.


But we've learned a lot from the re-ranker!
8
7
6
5
4
3
2
1
A B C D E F G H
3.
1.
2.
re-ranker
(e.g., BERT/GPT/etc.)
?
?
?
?

Adaptive Re-Ranking leverages the information gained from high-scoring documents to find ones missed by the retriever.

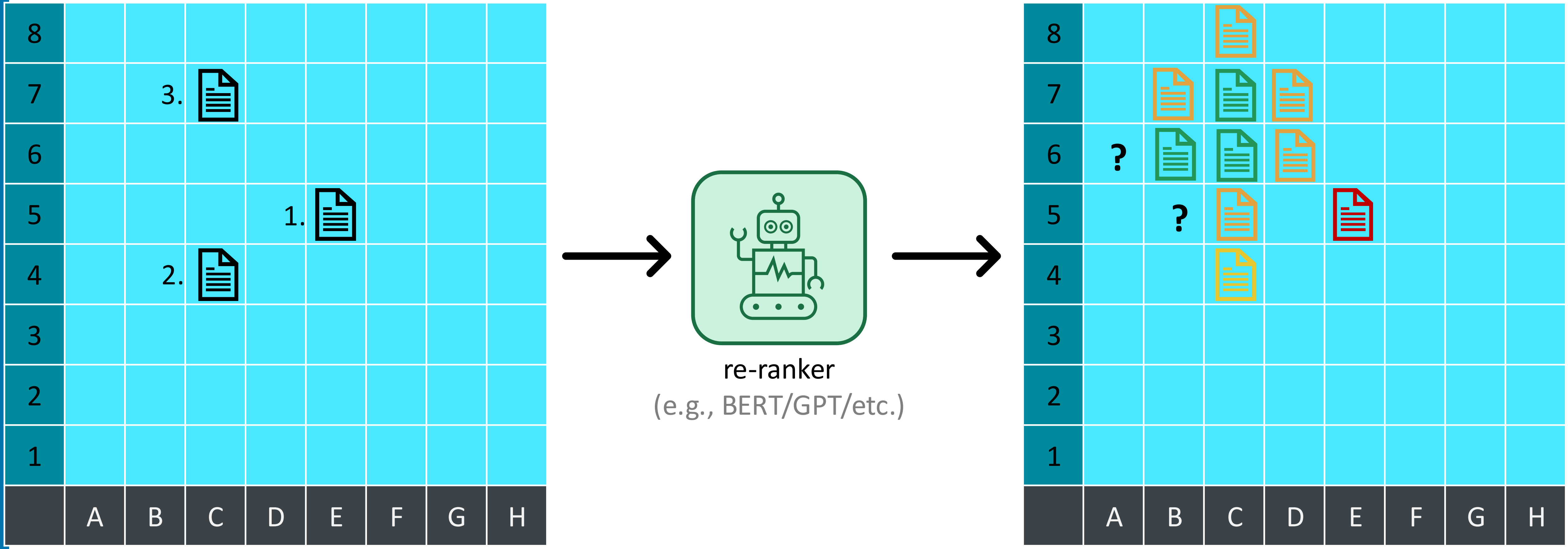


Adaptive Re-Ranking leverages the information gained from high-scoring documents to find ones missed by the retriever.



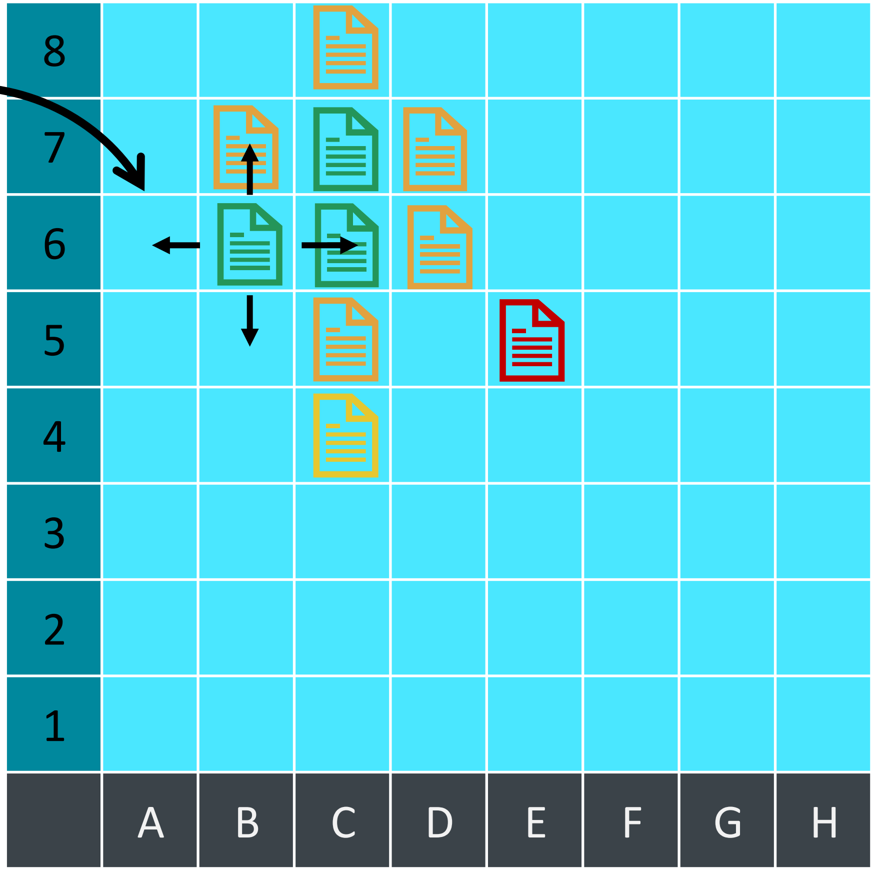
And so forth…

Adaptive Re-Ranking leverages the information gained from high-scoring documents to find ones missed by the retriever.



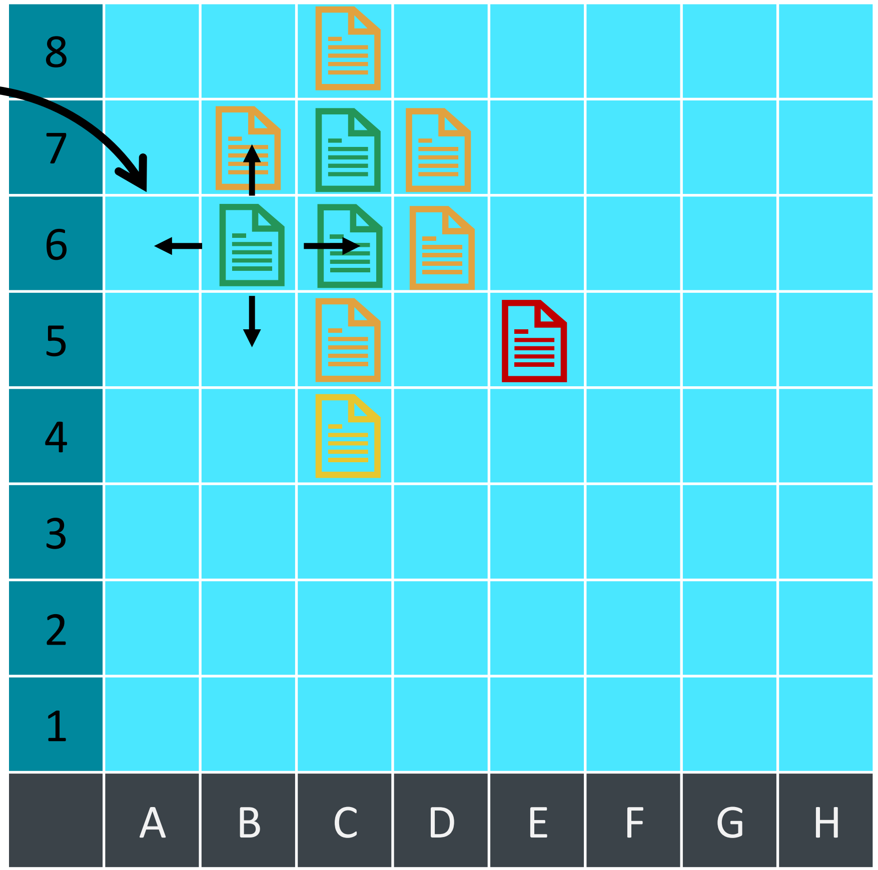
And so forth…

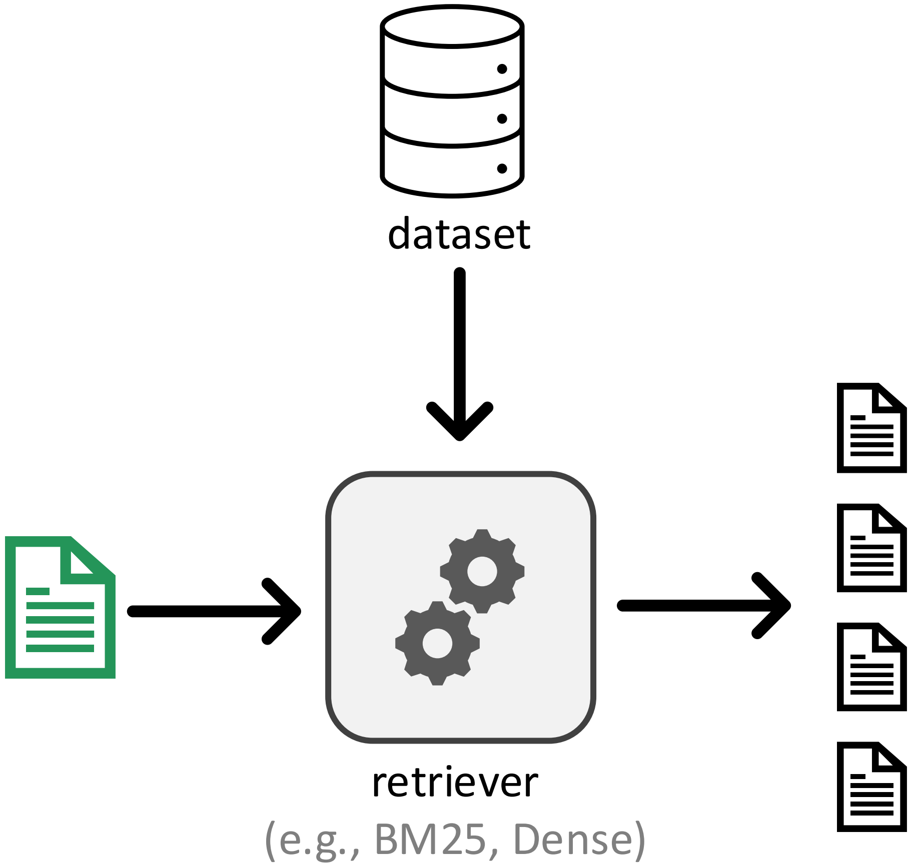
How to decide which documents to check?



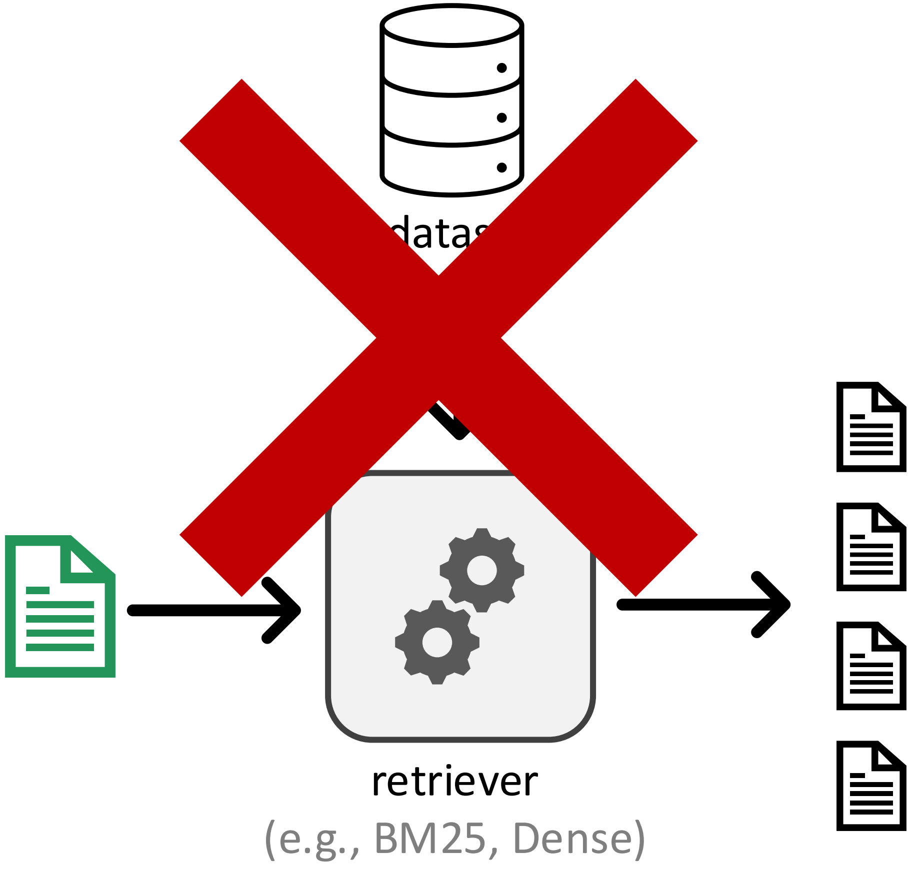
We could issue the document as a query to the engine and take the top *k* results.

How to decide which documents to check?





We could issue the document as a query to the engine and take the top *k* results.



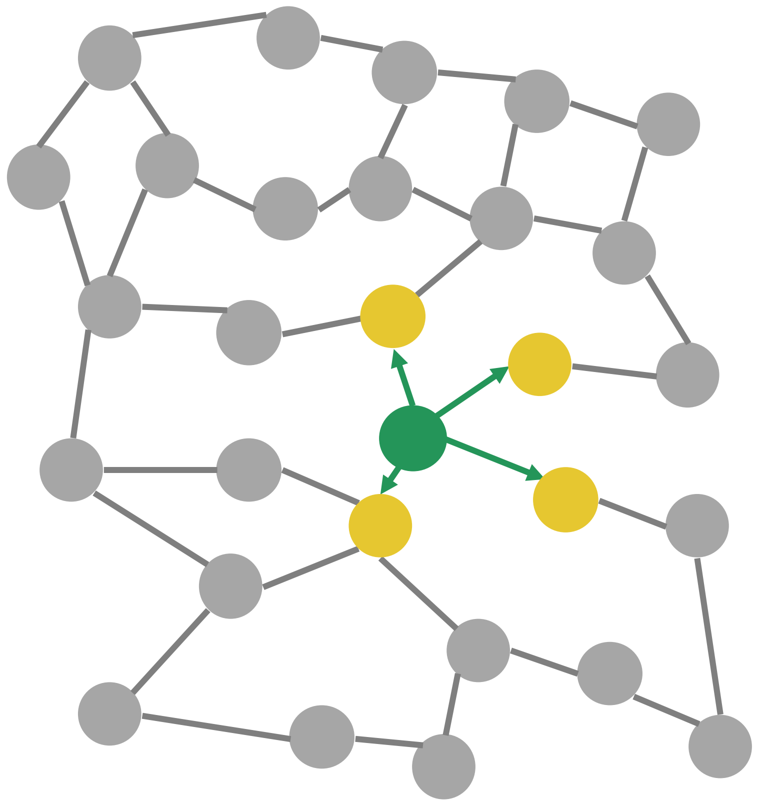
Really slow!

Right idea, bad execution.

Better: Use a KNN graph.

(You may recognize from HNSW search.)

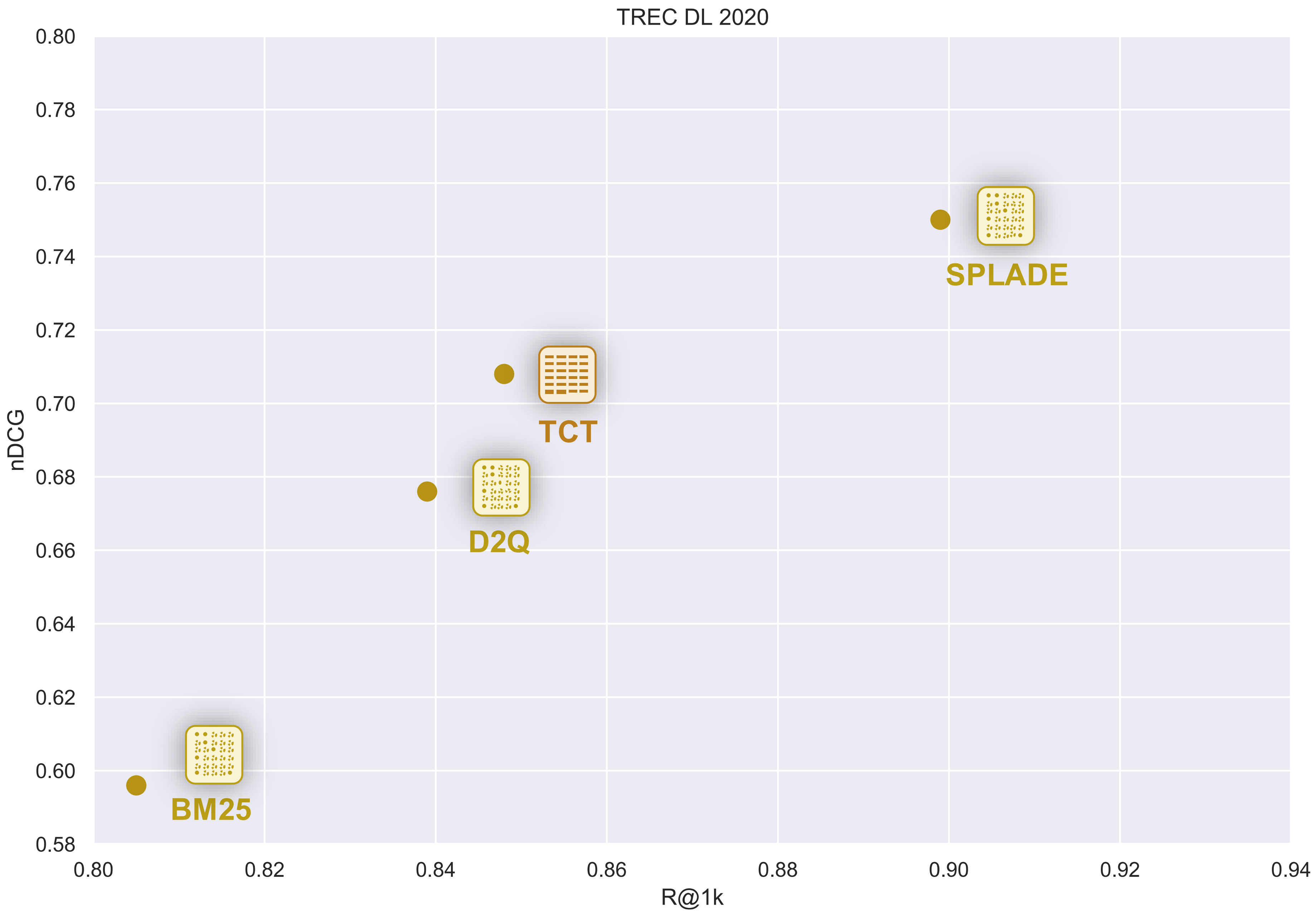
- Establishes proximity
- Fast lookups
- Constructed offline

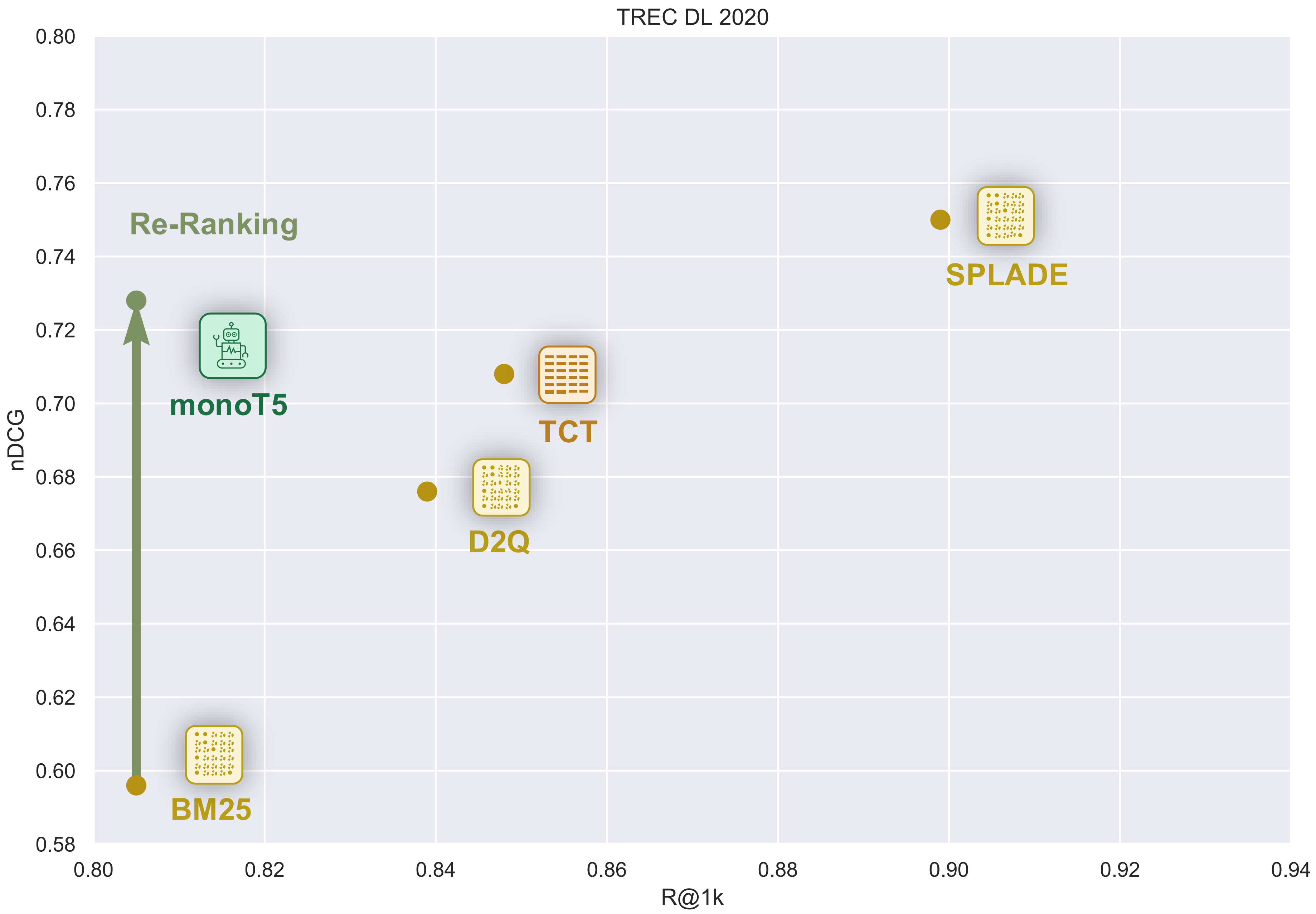


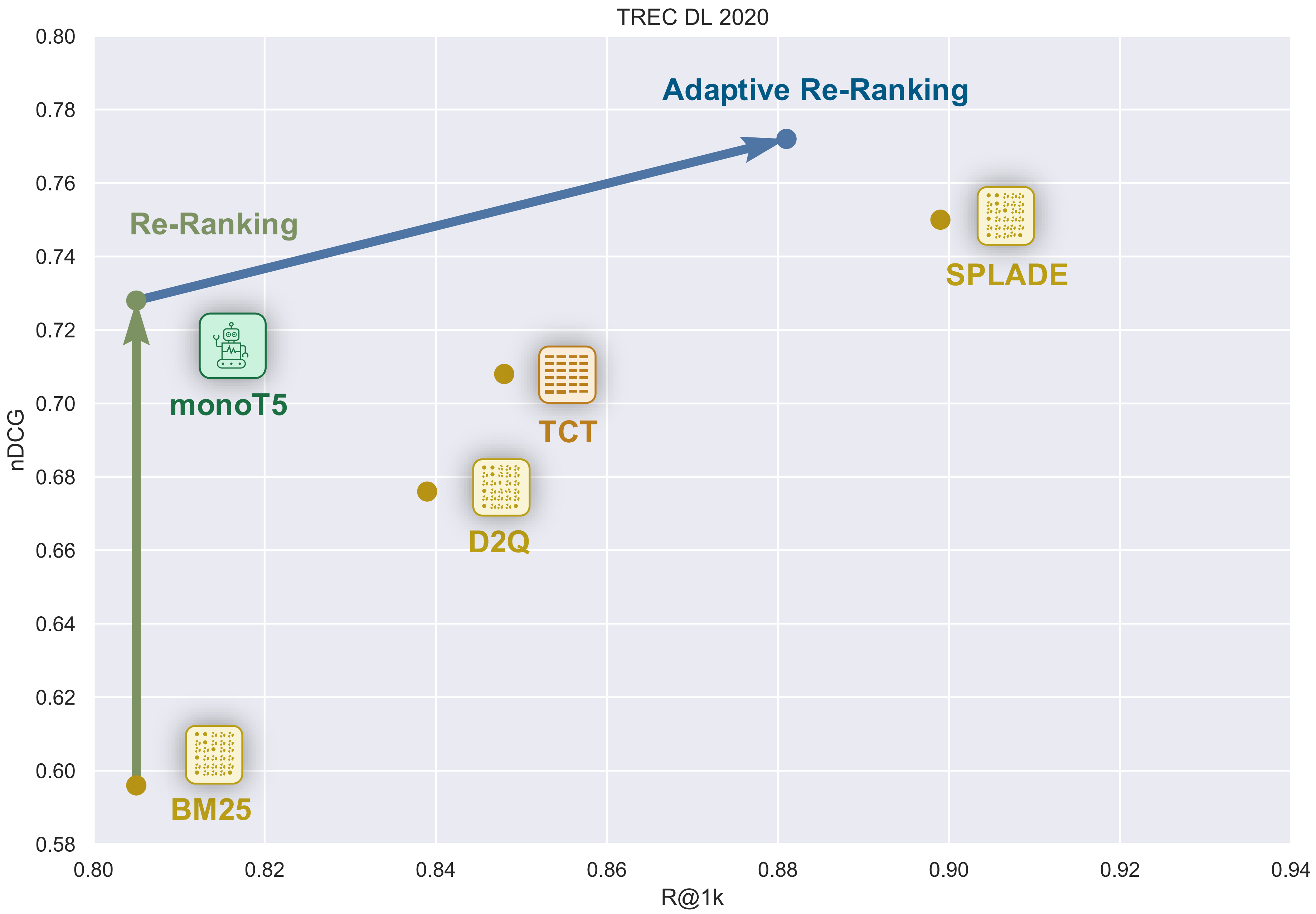
Alright, so how well does this *adaptive* re-ranking strategy work?

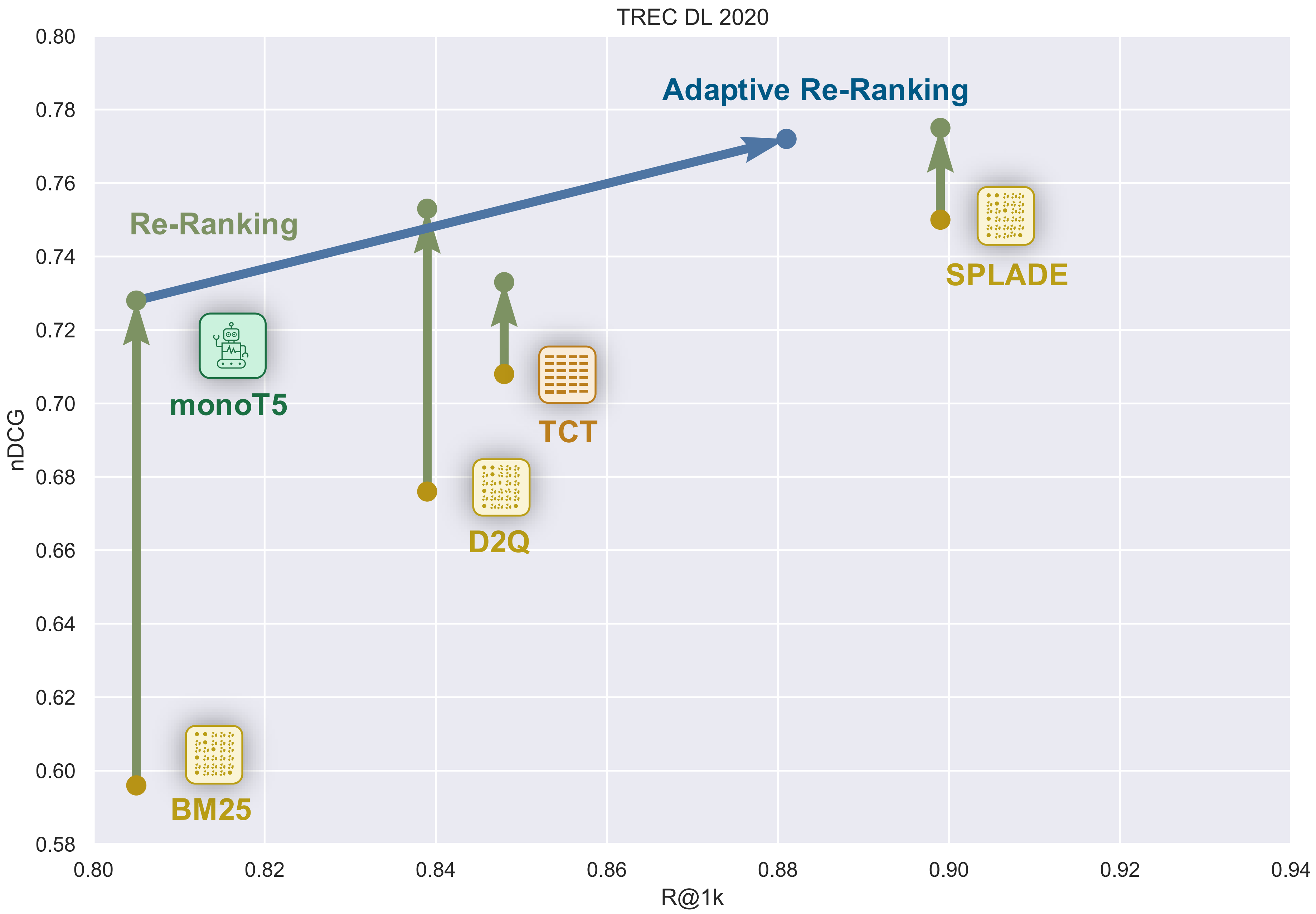
A few technical bits…

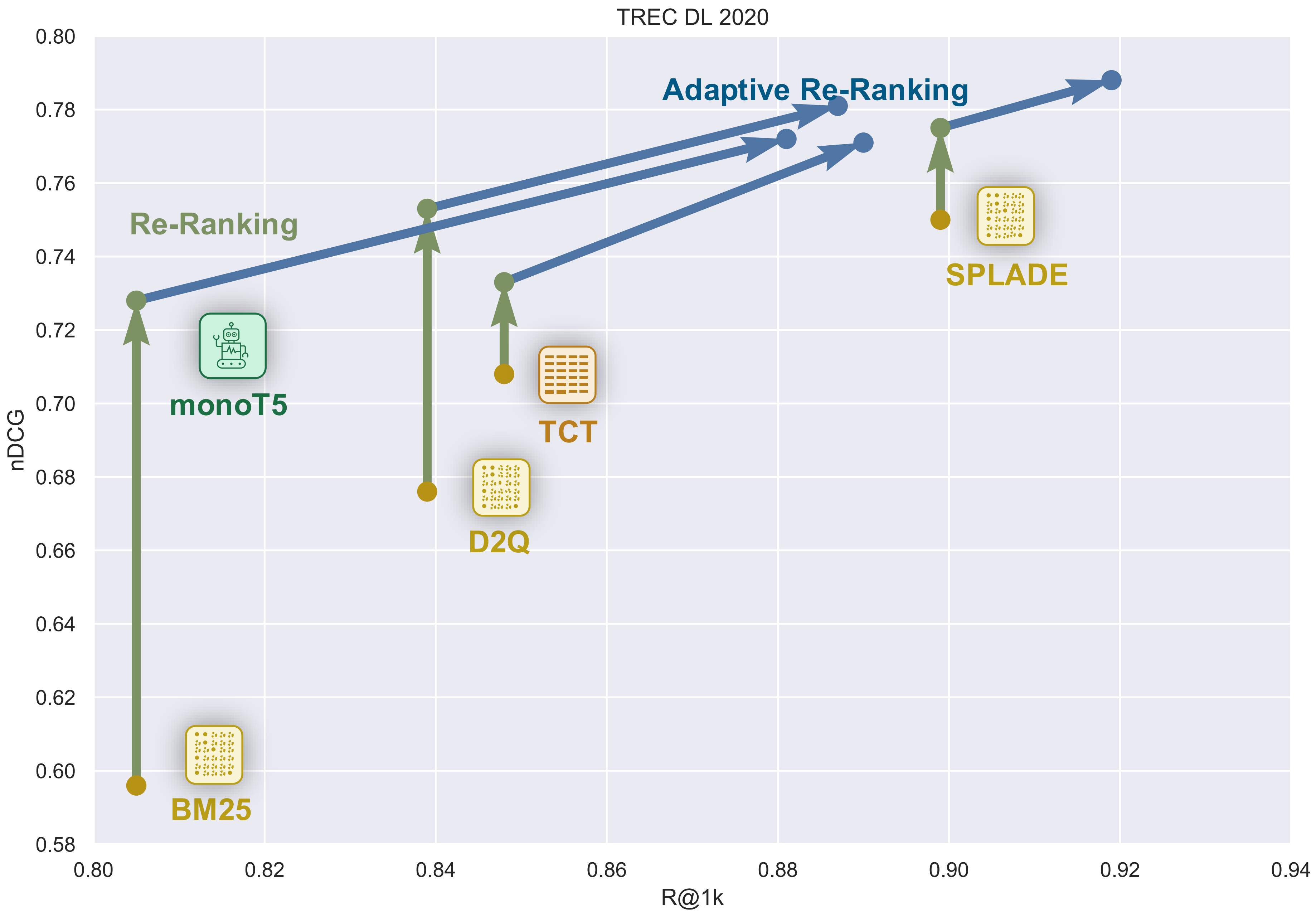
- We fix the re-ranking “budget” (number of docs to score) across all pipelines.
- In adaptive setting, we take half from first-stage ranker and half from graph.
- Measure nDCG (overall ranking quality) and Recall (% of relevant docs retrieved).
- Test on a variety of re-ranking pipelines.


TREC DL 2020
0.80
0.78
0.76
0.74
0.72
0.70
0.68
0.66
0.64
0.62
0.60
0.58
nDCG
SPLADE
TCT
D2Q
BM25
0.80
0.82
0.84
0.86
0.88
0.90
0.92
0.94
R@1k


TREC DL 2020
Re-Ranking
monoT5
SPLADE
TCT
D2Q
BM25
nDCG
R@1k
0.80
0.78
0.76
0.74
0.72
0.70
0.68
0.66
0.64
0.62
0.60
0.58
0.80
0.82
0.84
0.86
0.88
0.90
0.92
0.94


TREC DL 2020
Adaptive Re-Ranking
Re-Ranking
SPLADE
monoT5
TCT
D2Q
BM25
nDCG
R@1k
0.80
0.78
0.76
0.74
0.72
0.70
0.68
0.66
0.64
0.62
0.60
0.58
0.80
0.82
0.84
0.86
0.88
0.90
0.92
0.94


TREC DL 2020
Adaptive Re-Ranking
Re-Ranking
monoT5
TCT
D2Q
SPLADE
BM25
nDCG
0.80
0.78
0.76
0.74
0.72
0.70
0.68
0.66
0.64
0.62
0.60
0.58
R@1k
0.80
0.82
0.84
0.86
0.88
0.90
0.92
0.94


TREC DL 2020
Adaptive Re-Ranking
Re-Ranking
monoT5
TCT
D2Q
SPLADE
BM25
nDCG
R@1k
0.80
0.78
0.76
0.74
0.72
0.70
0.68
0.66
0.64
0.62
0.60
0.58
0.80
0.82
0.84
0.86
0.88
0.90
0.92
0.94

Sean MacAvaney
@macavaney

···

I ❤️ cross-encoders! Awesome to see another one from Cohere

A quick test showing it turbocharged when using Graph-based Adaptive Reranking :)

```
dataset = pt.get_dataset('irds:msmarco-passage/trec-dl-2019/judged')

pt.Experiment(
  [
    bm25,
    bm25 >> cohere_rerank,
    bm25 >> GAR(cohere_rerank, graph, num_results=100),
  ],
  dataset.get_topics(),
  dataset.get_qrels(),
  [nDCG@10, nDCG, R(rel=2)@100]
)

#                 name  nDCG@10   nDCG  R(rel=2)@100
#                 BM25    0.499  0.459         0.497
#       BM25 >> Rerank    0.708  0.523         0.497
#   BM25 >> GAR(Rerank)   0.753  0.604         0.609
```

ALT

In summary

Nearest neighbor graph exploration helps re-rankers

Focusing on the neighbors of the top scored documents helps prioritize the documents that are most likely to be relevant

Other findings

- A version works for LLM-based listwise re-rankers
- Even works when no relevant documents returned by the first stage
- Robust to various measures of document similarity (semantic or lexical)

Conference Papers:

MacAvaney, Tonellotto, Macdonald. Adaptive Re-Ranking with a Corpus Graph. CIKM 2022.
Rathee, MacAvaney, Anand. Guiding Retrieval using Large Language Models. ECIR 2025.

Retrievers

Adaptive Re-Ranking involved **two** strategies:

1. Score docs near the best ones found so far.

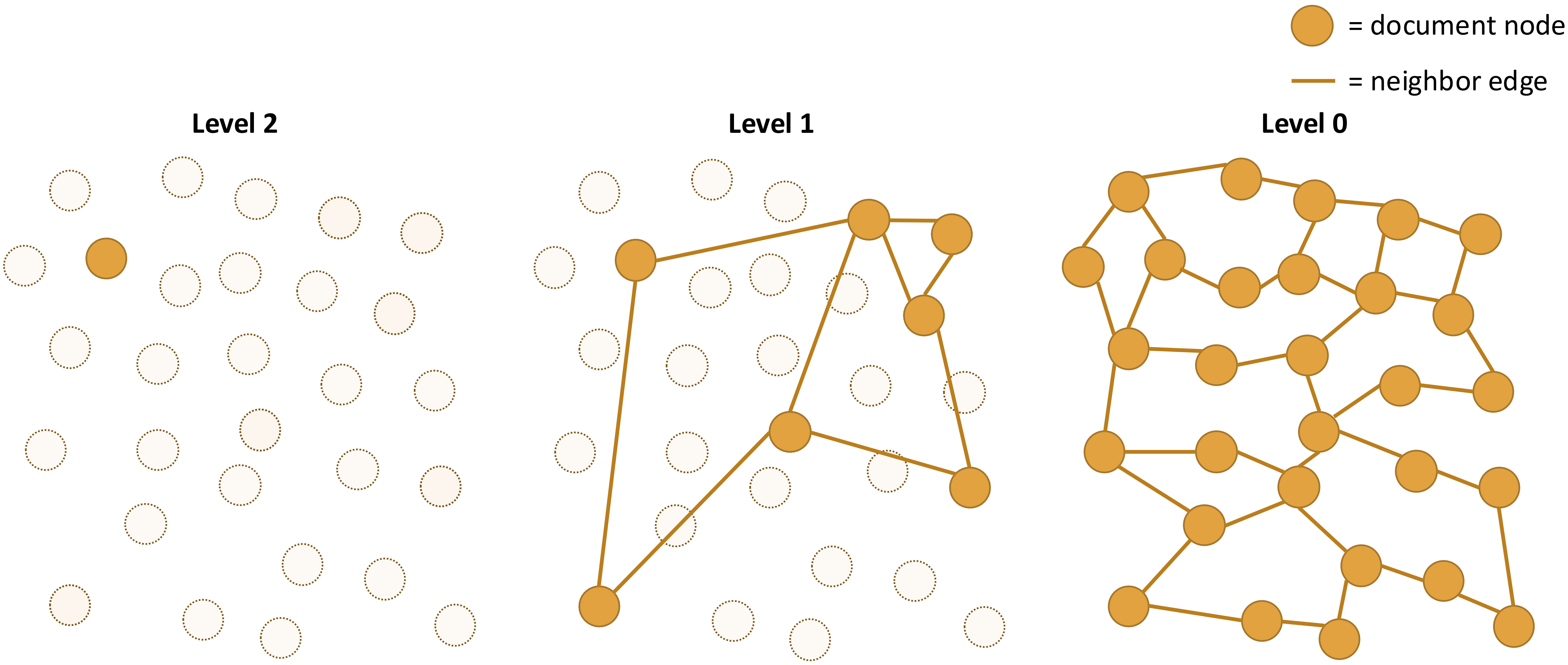
Adaptive Re-Ranking involved **two** strategies:

1 Score docs near the best ones found so far.

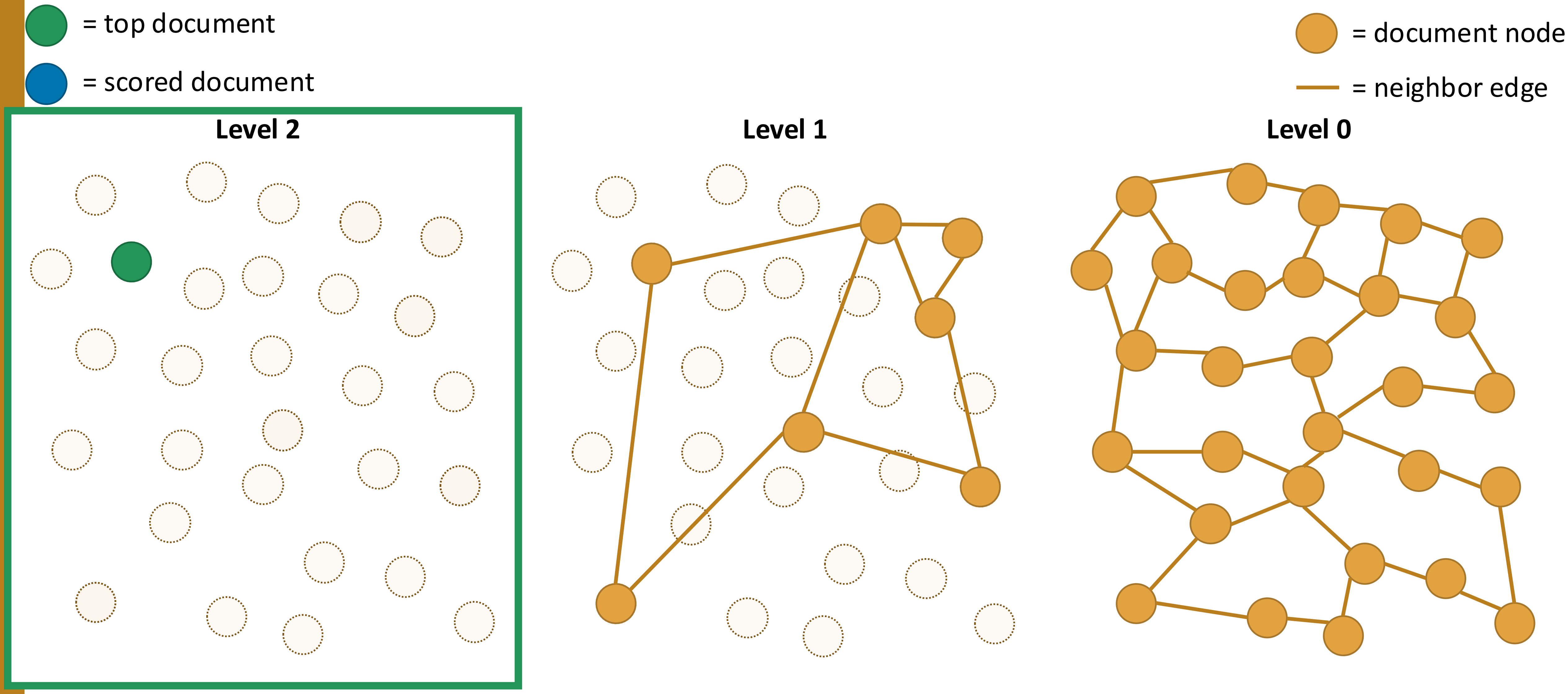
Adaptive **Re-Ranking** involved **two** strategies:

1. Score docs near the best ones found so far.
2. Start with good (but cheap) guesses.

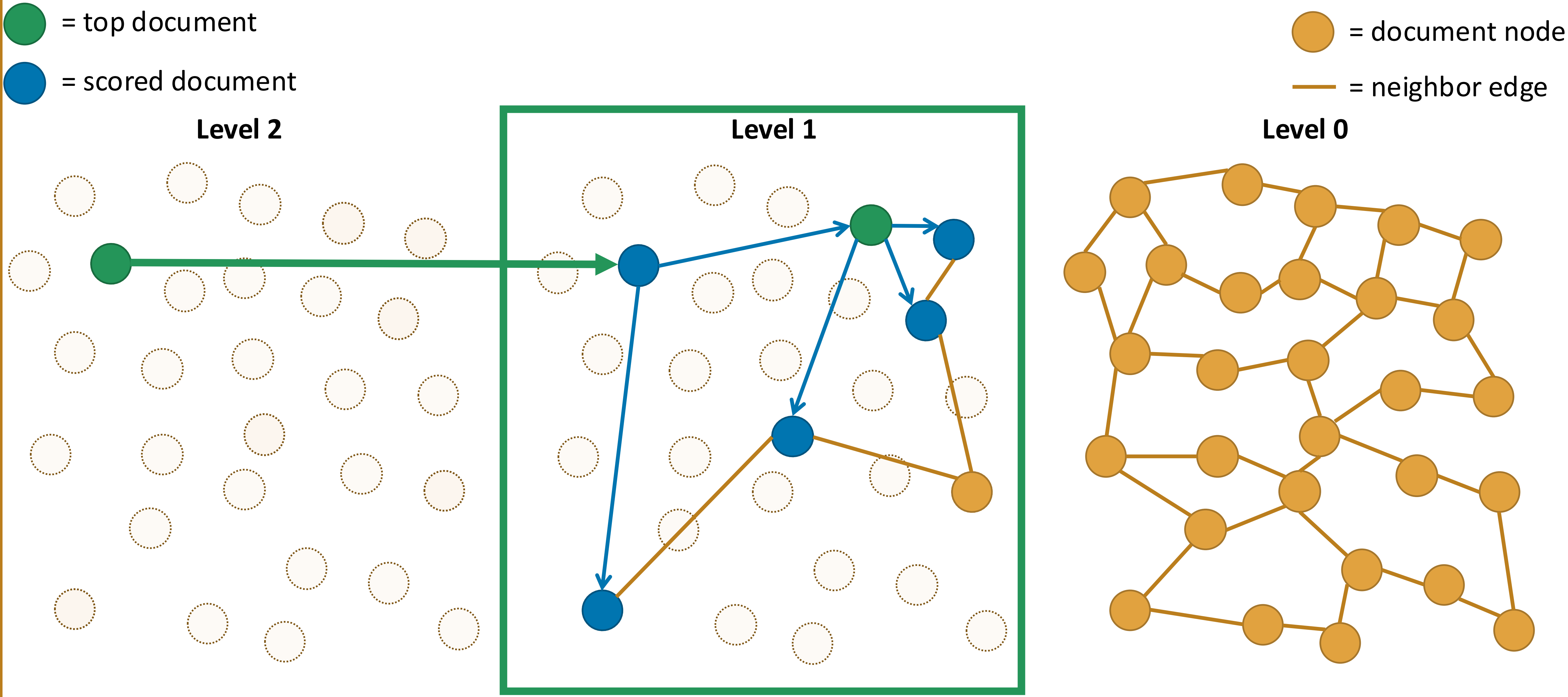
Can this strategy help dense retrievers?



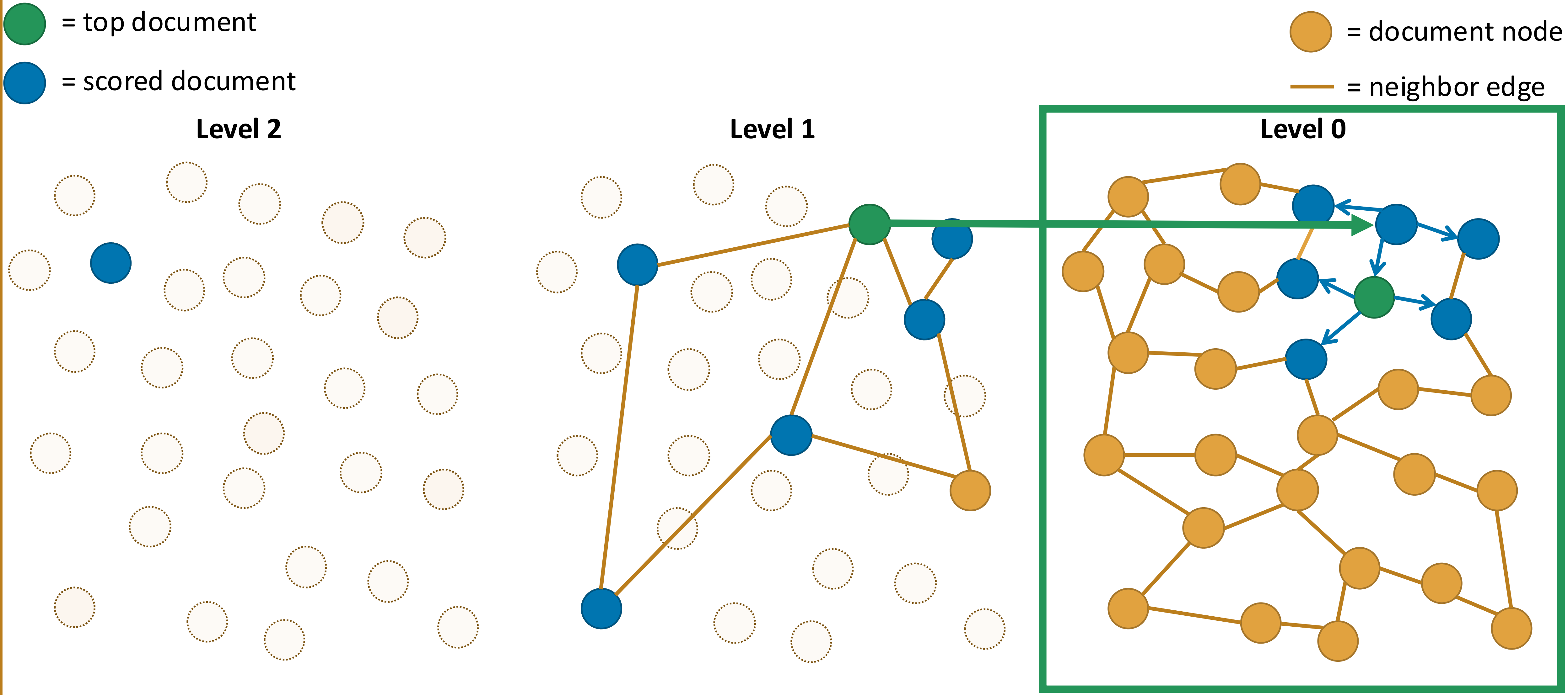
HNSW: Score random nodes to narrow in on the best ones.



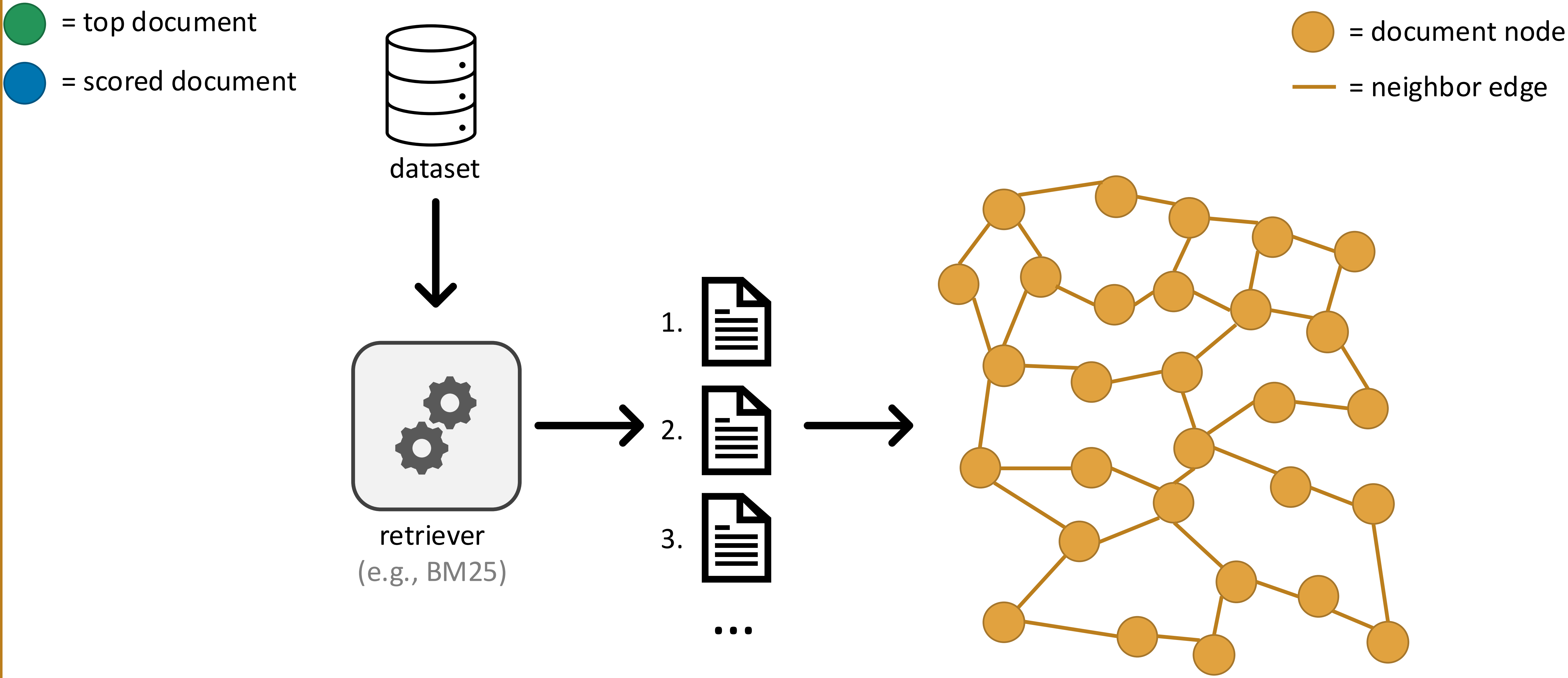
HNSW: Score random nodes to narrow in on the best ones.



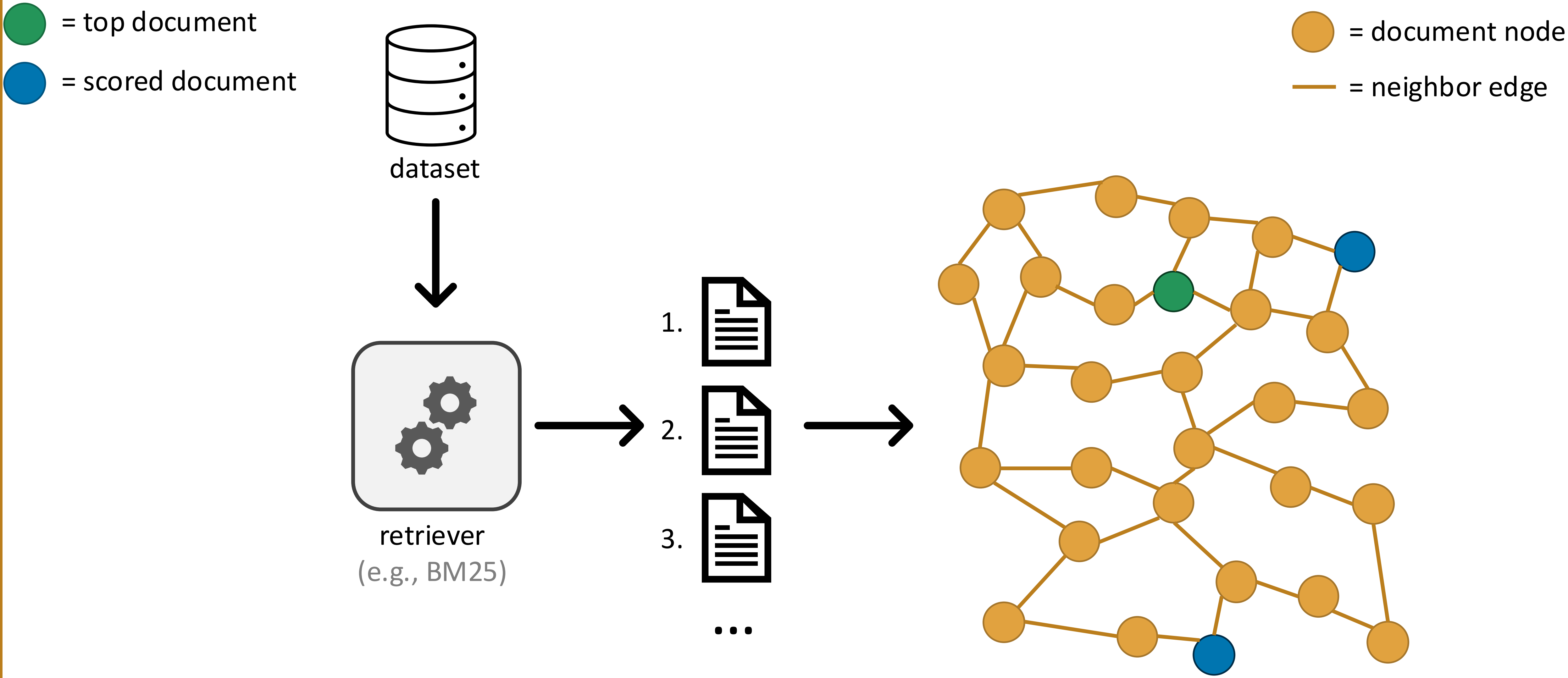
HNSW: Score random nodes to narrow in on the best ones.



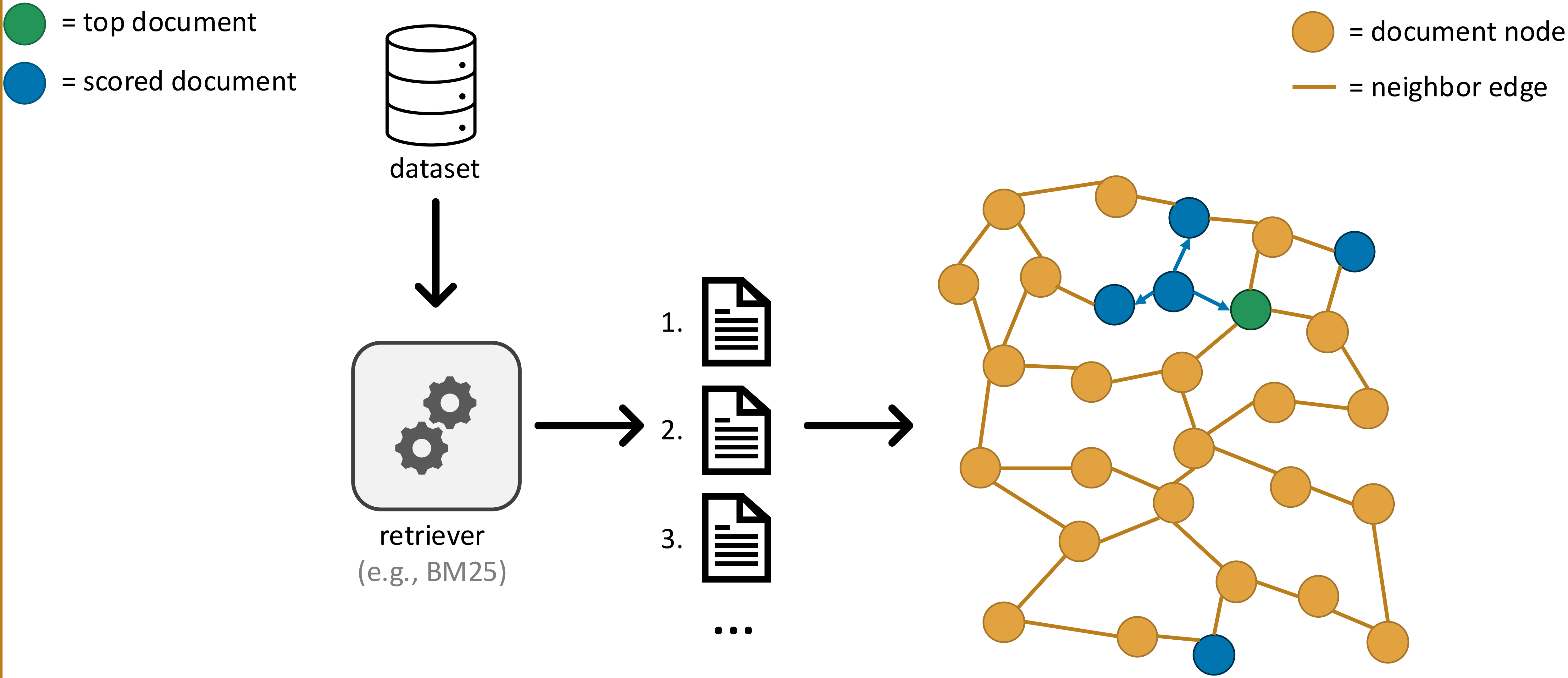
HNSW: Score random nodes to narrow in on the best ones.



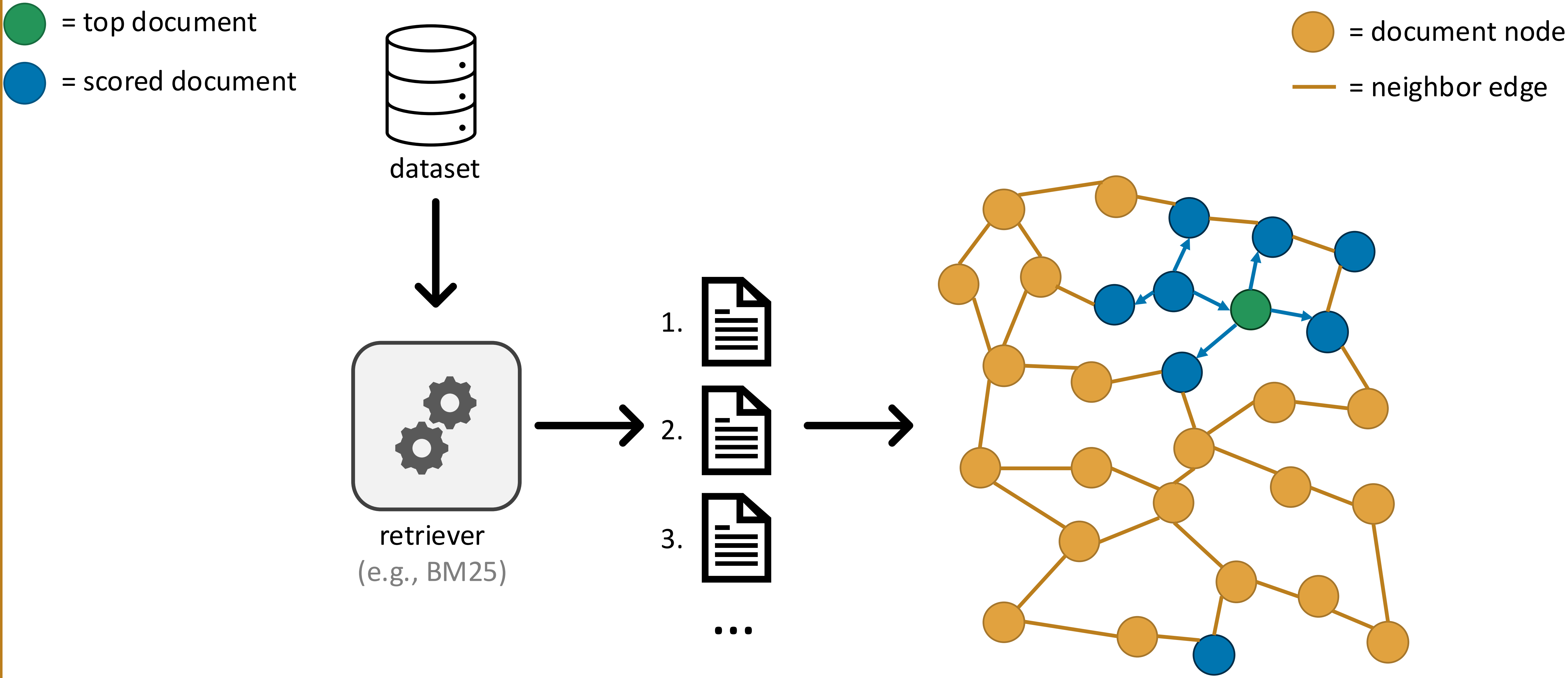
LADR (Lexically-Accelerated Dense Retrieval):
Use lexical search to seed dense retrieval.



LADR (Lexically-Accelerated Dense Retrieval):
Use lexical search to seed dense retrieval.



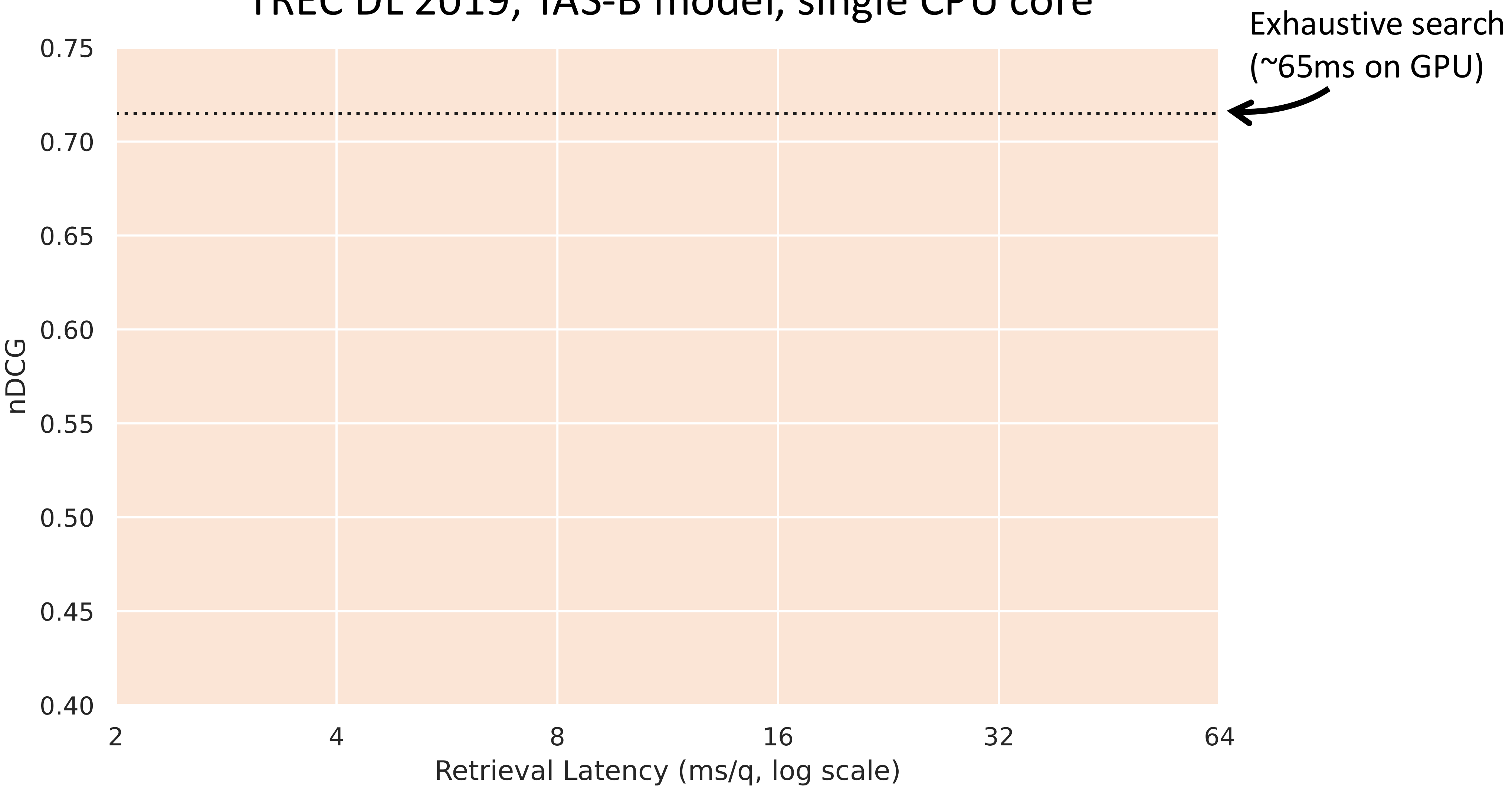
LADR (Lexically-Accelerated Dense Retrieval):
Use lexical search to seed dense retrieval.

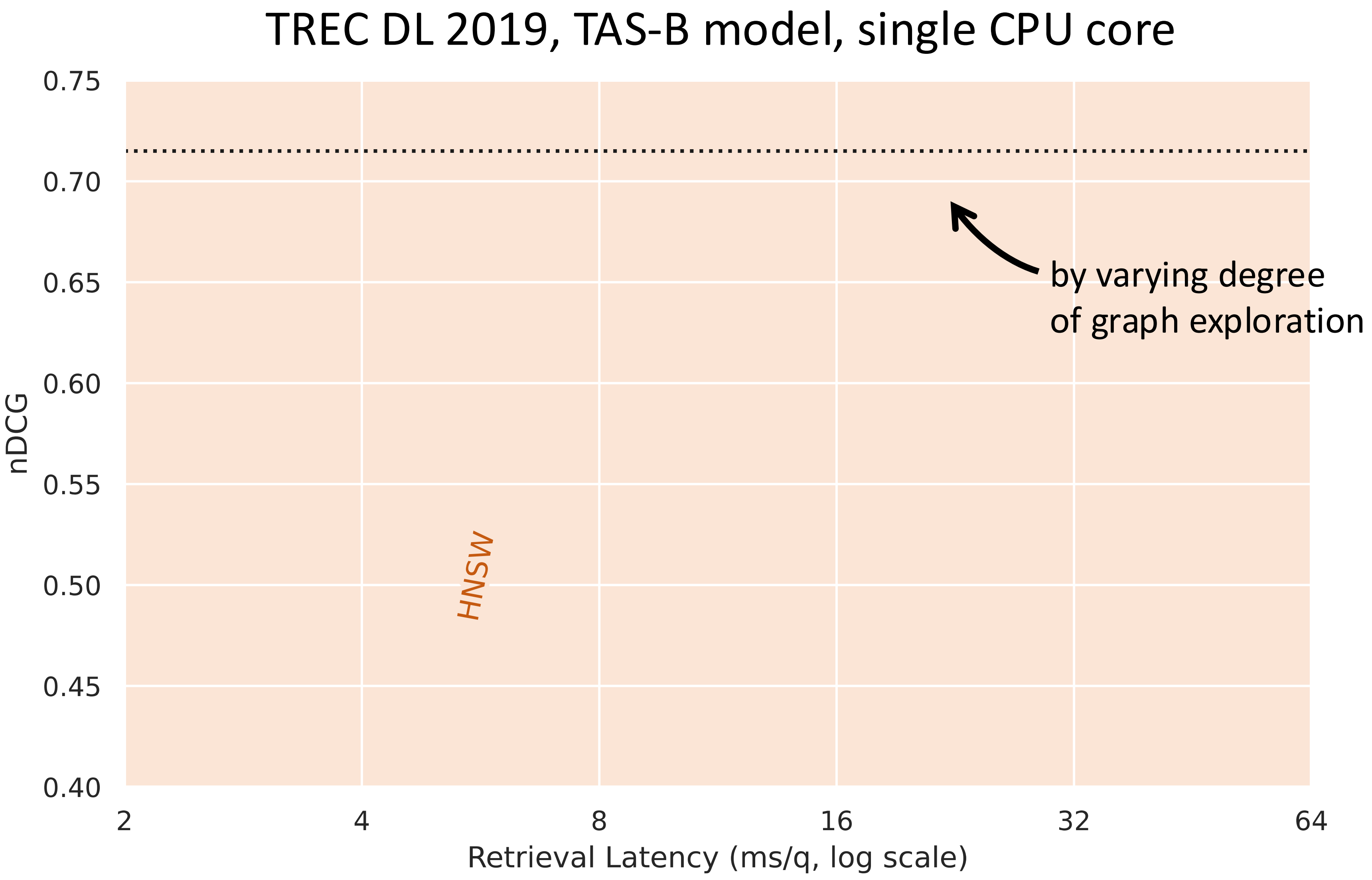

= top document
= scored document
dataset
retriever
(e.g., BM25)
1.
2.
3.
...
= document node
= neighbor edge

Iterate...

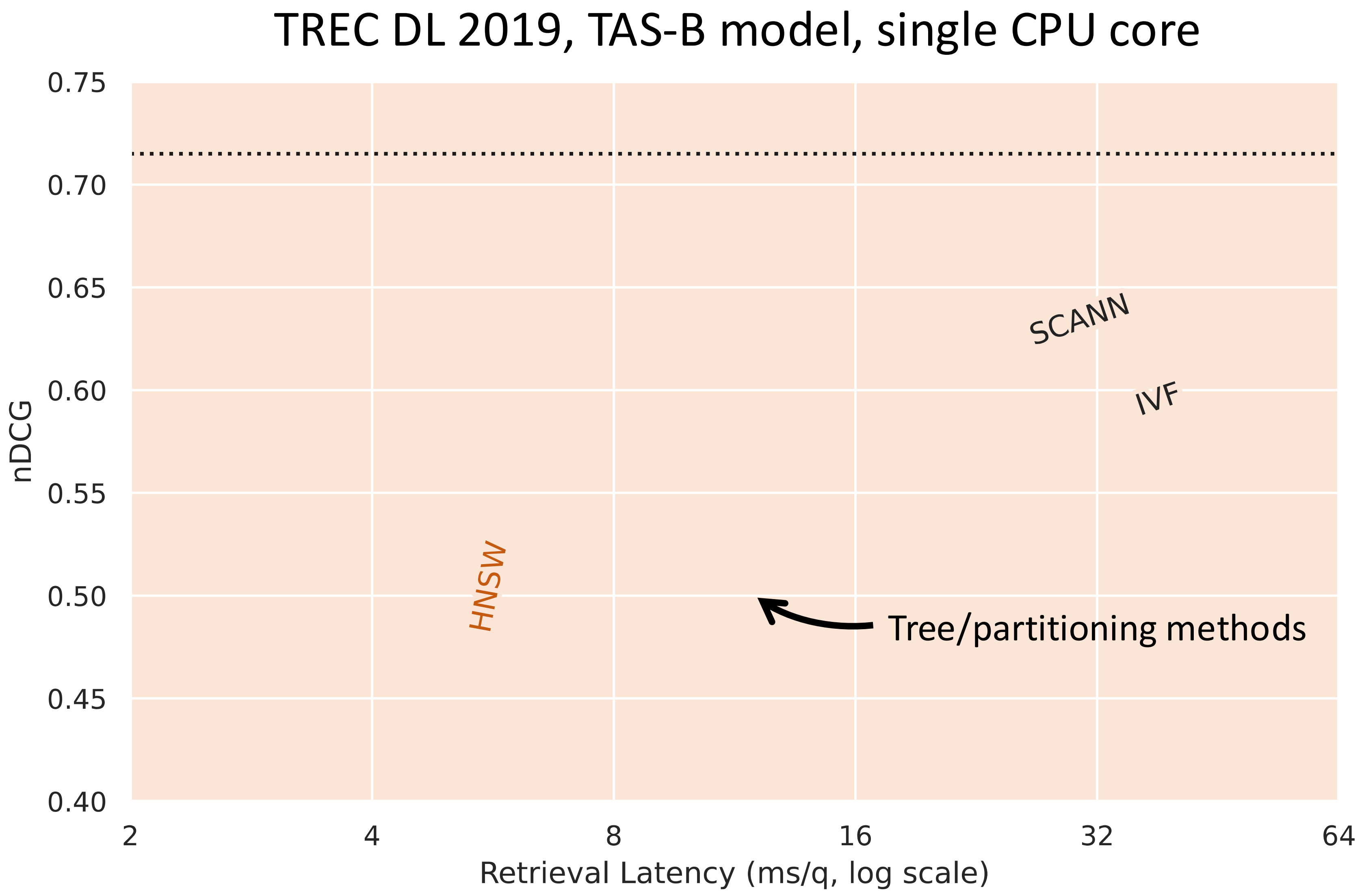
How well does it work?

TREC DL 2019, TAS-B model, single CPU core



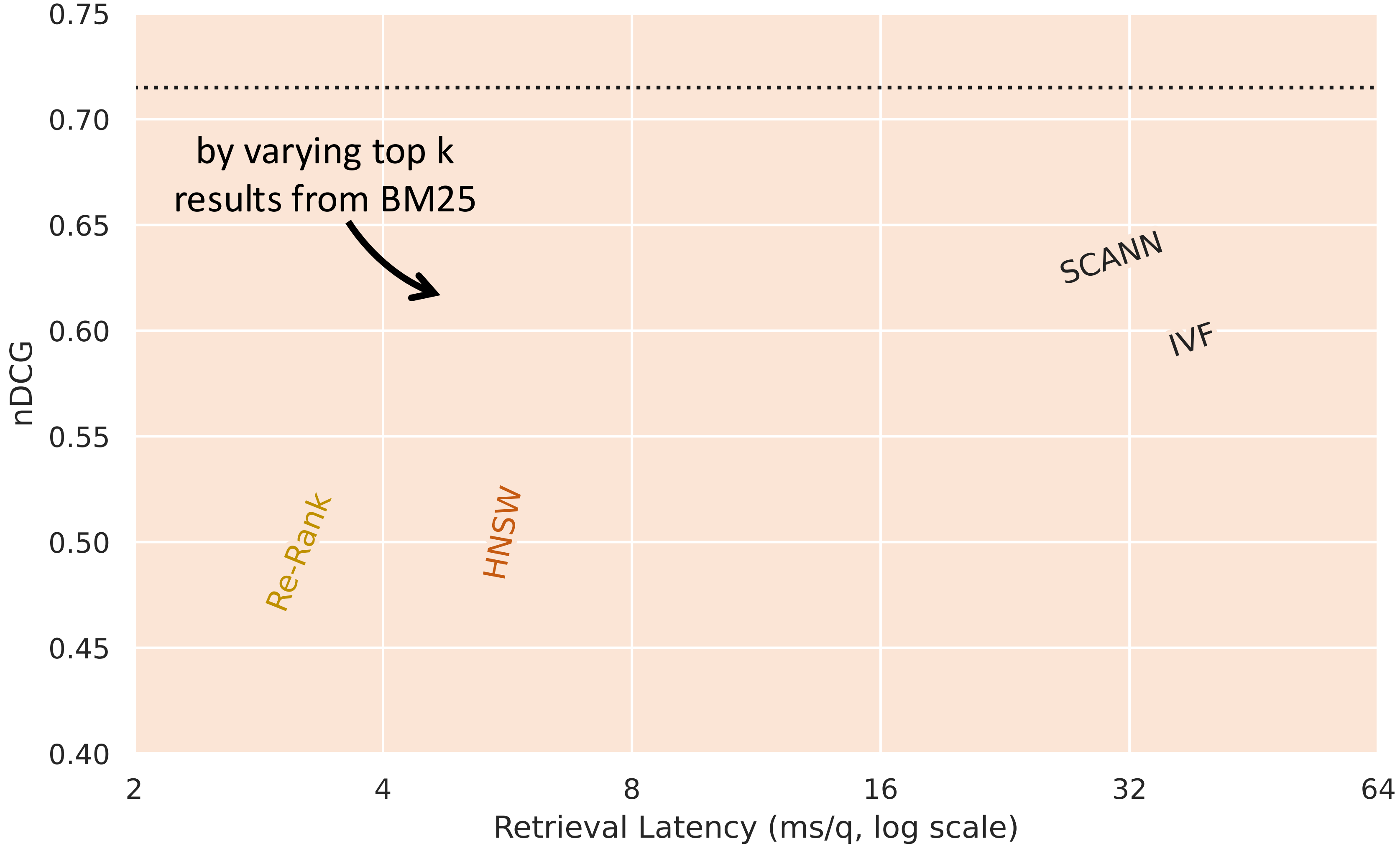


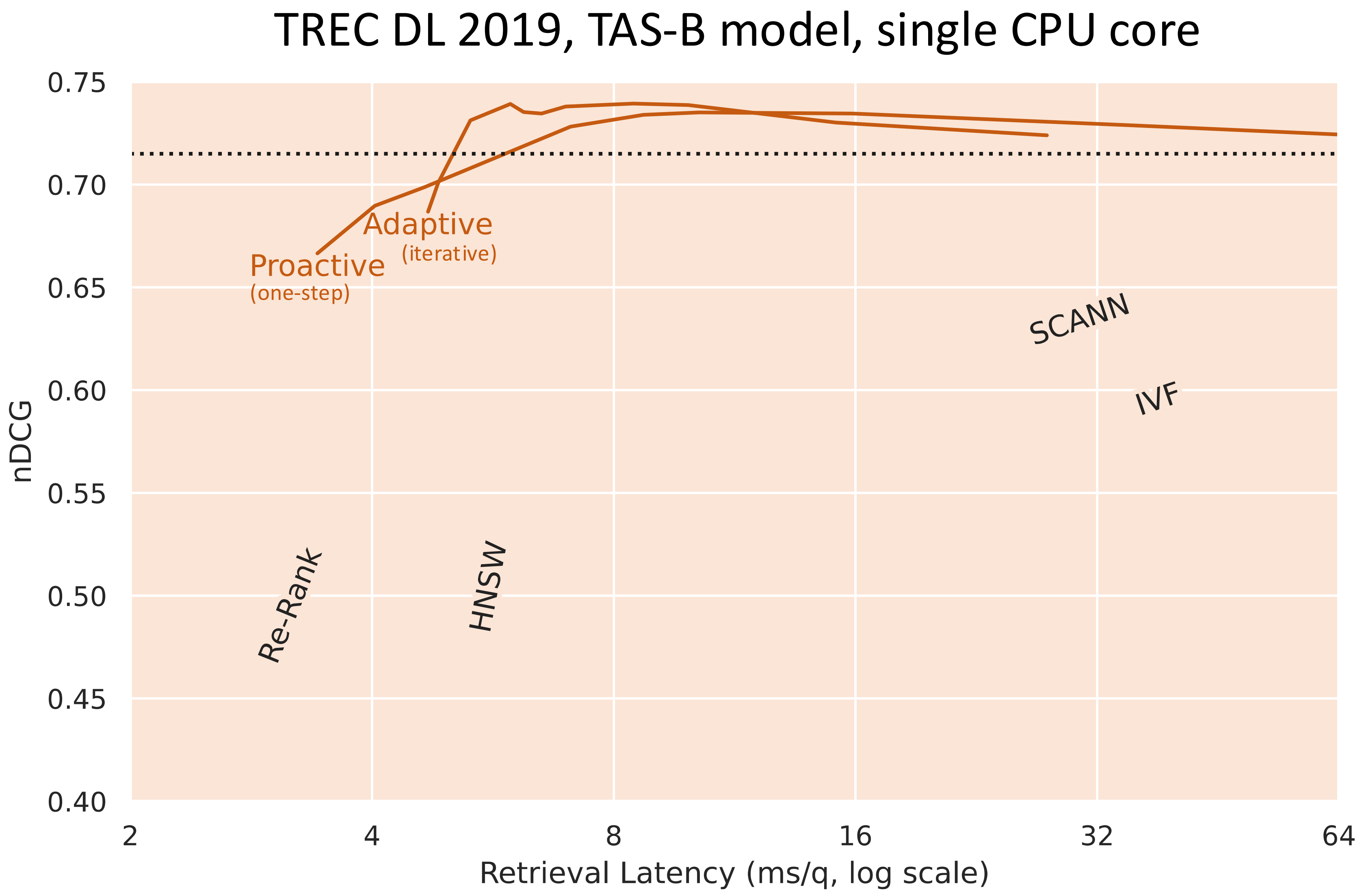
* FAISS implementation

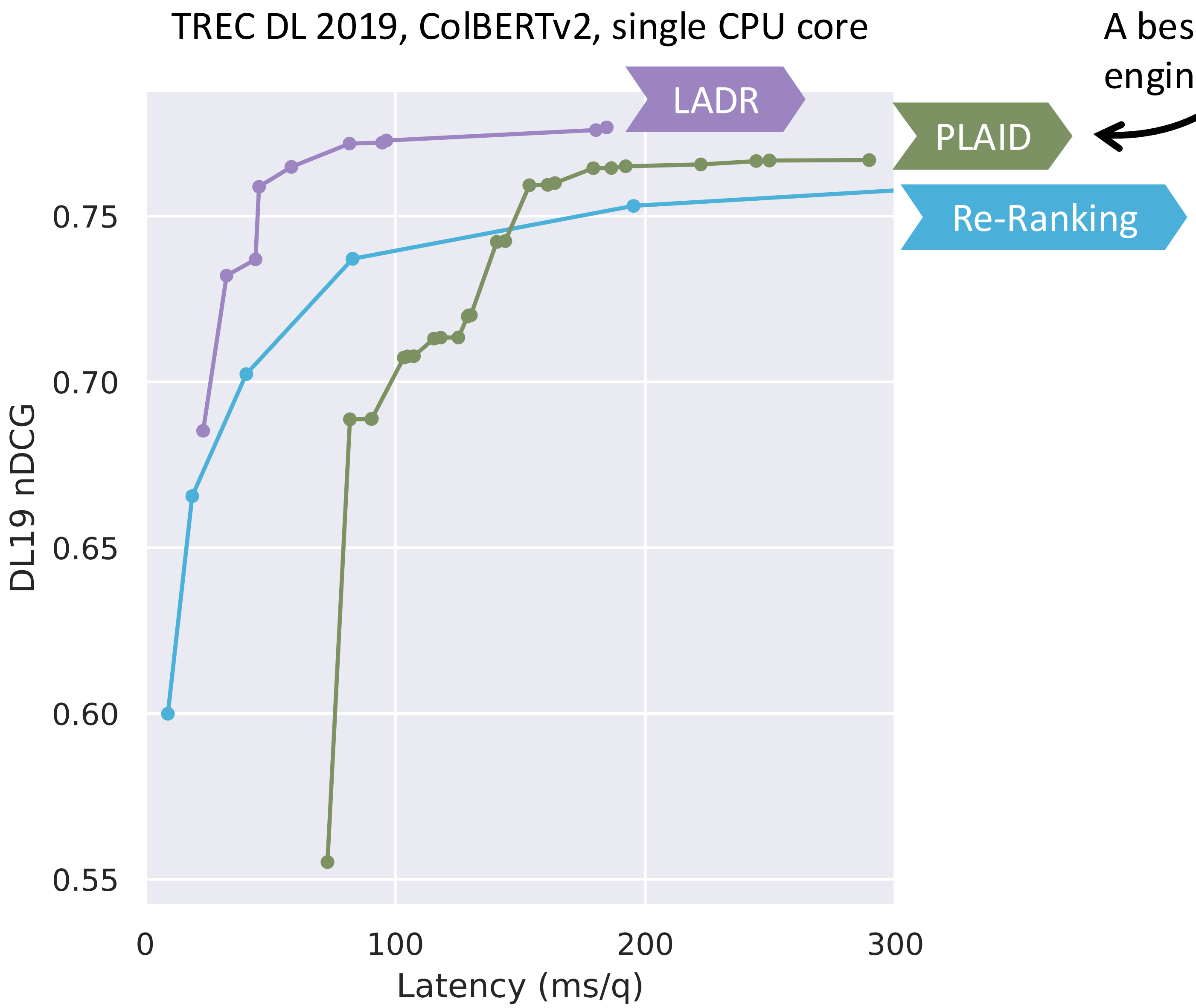


* FAISS implementation

TREC DL 2019, TAS-B model, single CPU core




TREC DL 2019, TAS-B model, single CPU core
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
nDCG
Adaptive
(iterative)
Proactive
(one-step)
SCANN
IVF
Re-Rank
HNSW
2
4
8
16
32
64
Retrieval Latency (ms/q, log scale)



A bespoke retrieval engine for ColBERT

In summary

Adaptive Re-Ranking improves **retrievers** too!

Both single-vector and multi-vector dense retrieval.

Other findings

- Clear trade-offs between the model parameters and efficiency/effectiveness
- Works across a variety of dense retrieval models and standard benchmarks
- Works with approximate NN graph, and even graphs constructed from other models

Conference Papers:

Kulkarni, MacAvaney, Goharian, Frieder. Lexically-Accelerated Dense Retrieval. SIGIR 2023.
MacAvaney, Tonellotto. A Reproducibility Study of PLAID. SIGIR 2024. [Best Paper Runner-Up]

Open Source!

Adaptive Re-Ranking and LADR in PyTerrier

```python
import pyterrier as pt
from pyterrier_t5 import MonoT5
from pyterrier_pisa import PisaIndex
from pyterrier_adaptive import GAR, CorpusGraph

bm25 = PisaIndex('my_index.pisa').bm25()
reranker = MonoT5()
graph = CorpusGraph.load('my_index.graph')

pipeline = bm25 >> GAR(reranker, graph)
```

```python
from pyterrier_dr import FlexIndex
from pyterrier_pisa import PisaIndex

sparse = PisaIndex('my_index.pisa').bm25()
dense = PisaIndex('my_index.flex').ladr()

ladr = sparse.bm25() >> dense.ladr()
```

https://github.com/terrierteam/pyterrier_dr
https://github.com/terrierteam/pyterrier_adaptive

Adaptive Re-Ranking using the `rerankers` package

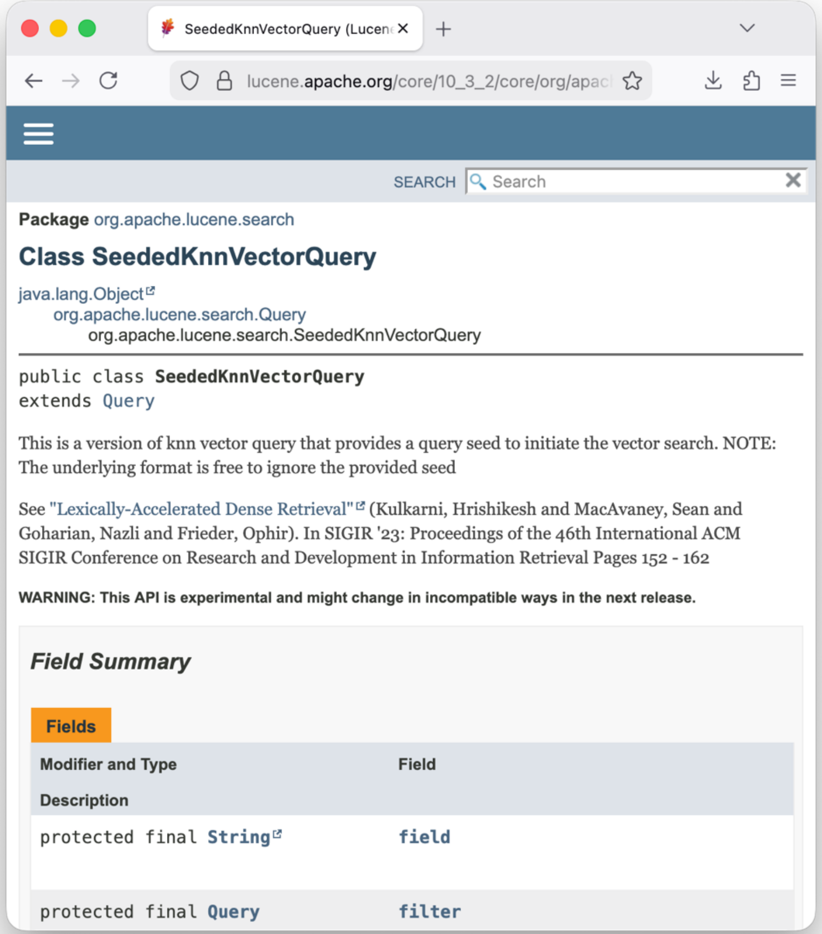
```
from rerankers import Reranker

reranker = Reranker(...)

adaptive_reranker = GAR(reranker.as_pyterrier_transformer(), graph)
```

Currently on my fork here: https://github.com/seanmacavaney/rerankers, pull request coming soon :)

Included in Lucene core



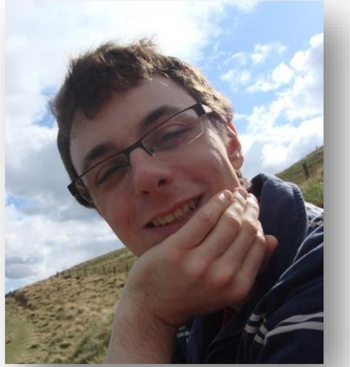
`org.apache.lucene.search.SeededKnnVectorQuery`

Adaptive re-ranking improves the quality of search results with minimal cost.

It improves retrievers like ColBERT, too.

Ready-to-use with Open-Source tools!

Thanks to my collaborators!



Craig Macdonald
University of Glasgow

Nicola Tonellotto
University of Pisa

Hrishikesh Kulkarni
Instacart

Nazli Goharian
Georgetown University

Ophir Frieder
Georgetown University

Mandeep Rathee
L3S Hanover

Venktesh V
Stockholm University

Avishek Anand
TU Delft

RE-THINKING RE-RANKING

Sean MacAvaney
University of Glasgow

Presented at:
Search Solutions 2025

Terrier

University of Glasgow

extra slides

Search Strategy	Effectiveness	Query Efficiency	Index Efficiency	Storage Costs
Sparse	Low	High	High	Low
Dense	High	Low	Low	High

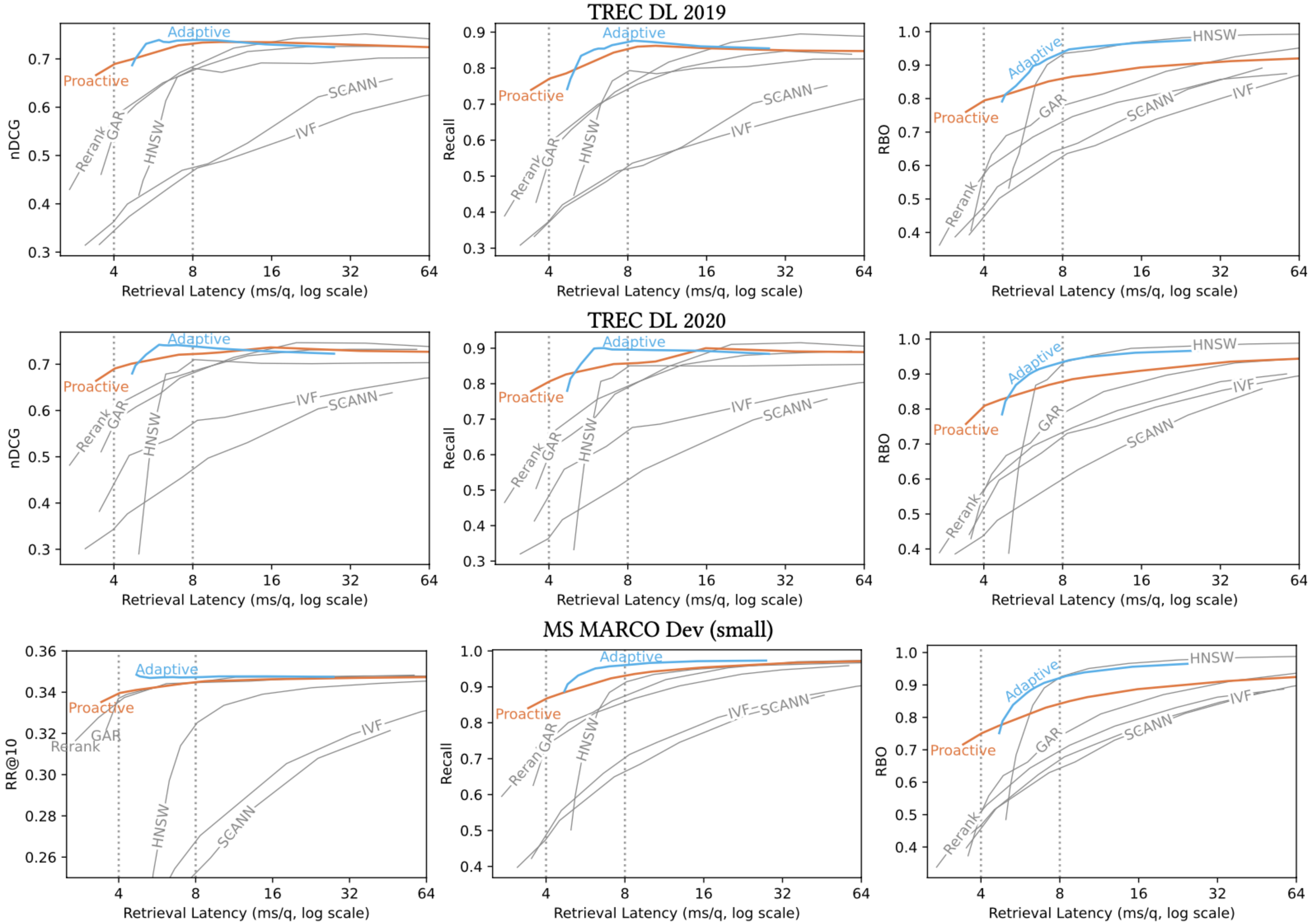
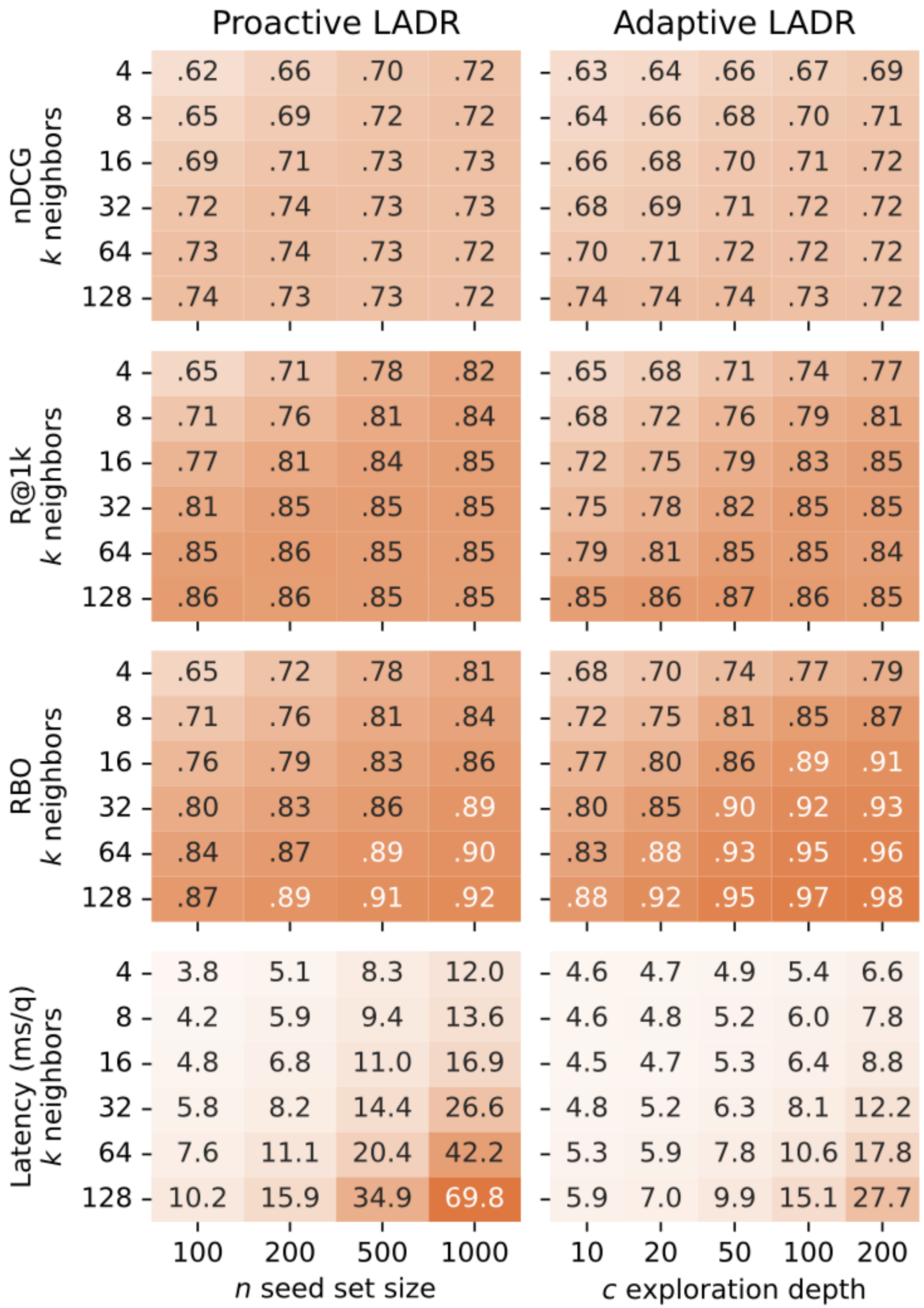


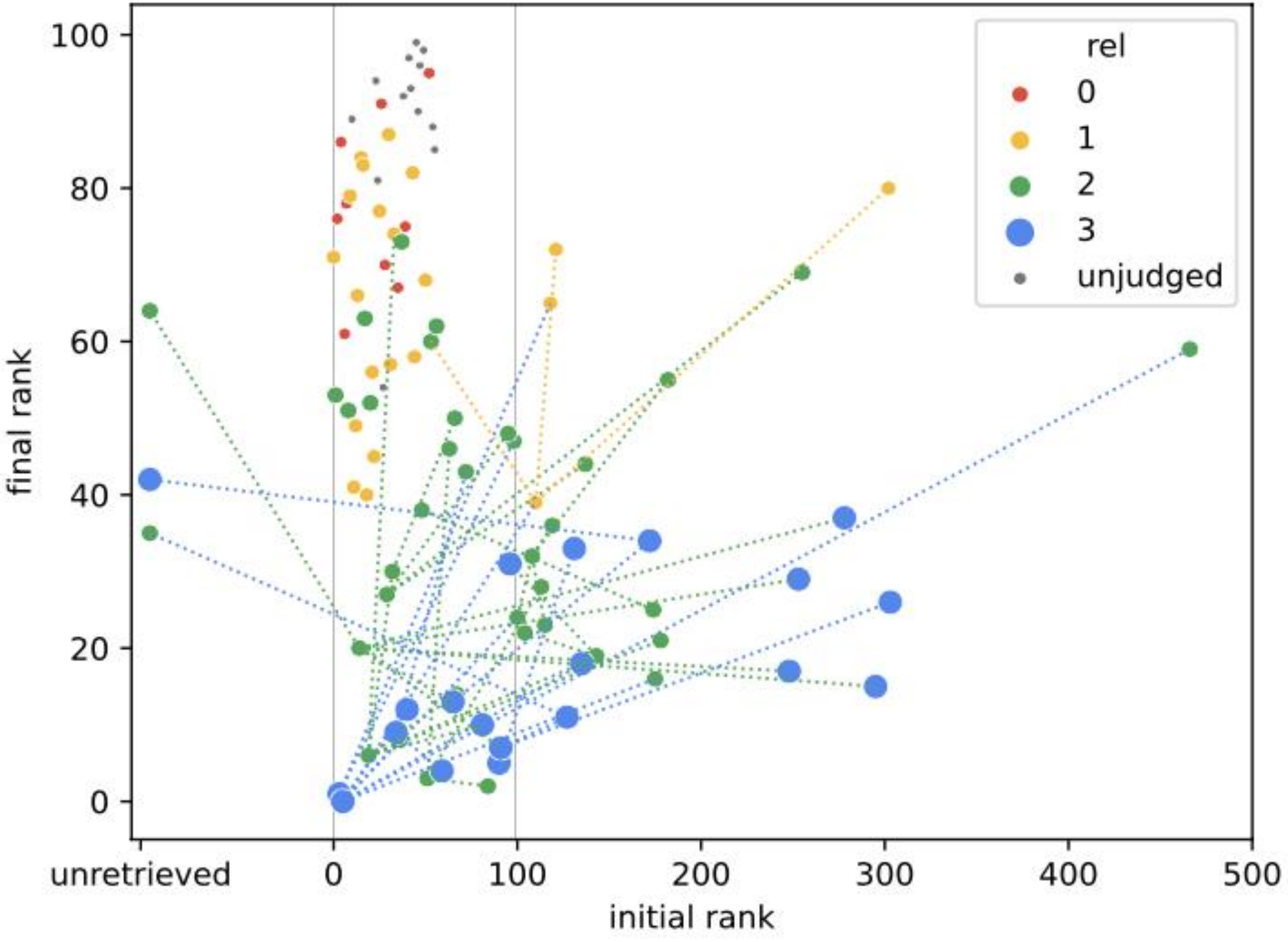
Figure 3: Performance of LADR over TAS-B and baselines across various operational points.

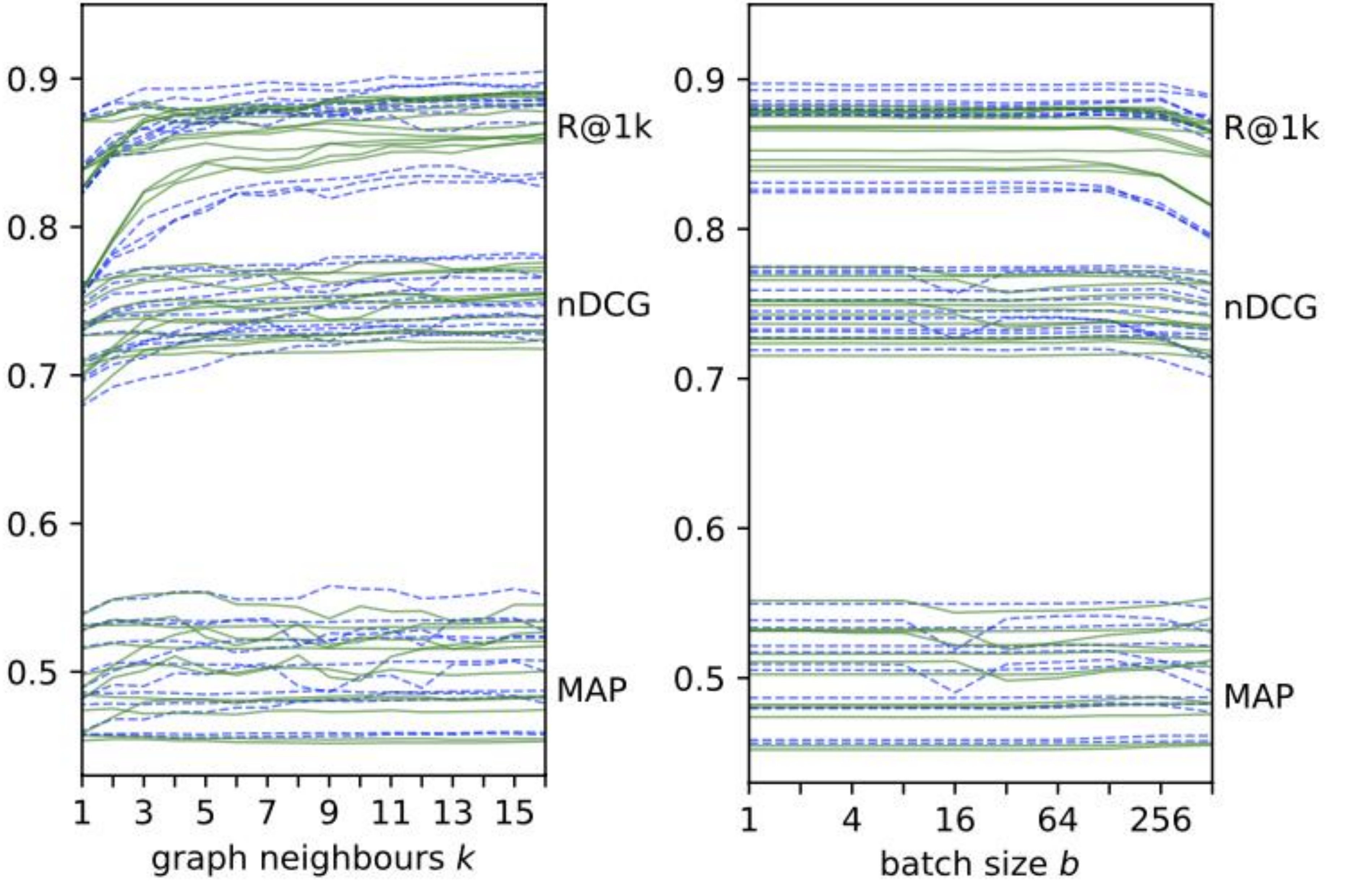
Method	DL19 ~4ms nDCG	DL19 ~4ms R@1k	DL19 ~8ms nDCG	DL19 ~8ms R@1k	DL20 ~4ms nDCG	DL20 ~4ms R@1k	DL20 ~8ms nDCG	DL20 ~8ms R@1k	Dev (sm) ~4ms RR@10	Dev (sm) ~4ms R@1k	Dev (sm) ~8ms RR@10	Dev (sm) ~8ms R@1k
TAS-B (Exh.)	0.715	0.842	0.715	0.842	0.713	0.875	0.713	0.875	0.347	0.978	0.347	0.978
IVF [I]	0.374	0.414	0.474	0.536	0.503	0.559	0.579	0.677	0.217	0.556	0.270	0.712
ScaNN [S]	0.475	0.519	0.537	0.598	0.476	0.527	0.553	0.641	0.254	0.669	0.292	0.774
HNSW [H]	-	-	0.614	0.707	-	-	0.699	0.836	-	-	0.310	0.872
GAR [G]	0.543	0.540	0.688	0.755	0.568	0.594	0.684	0.796	0.337	0.732	0.345	0.876
Re-Ranking [R]	0.589	0.605	0.684	0.755	0.615	0.667	0.691	0.805	0.337	0.748	0.345	0.868
Proactive LADR	$^{IS}_{GR}\mathbf{0.690}$	$^{IS}_{GR}\mathbf{0.771}$	$^{ISH}_{GR}0.730$	$^{ISH}_{GR}0.850$	$^{IS}_{GR}\mathbf{0.691}$	$^{IS}_{GR}\mathbf{0.807}$	$^{IS}_{GR}0.722$	$^{IS}_{GR}0.857$	$^{IS}\mathbf{0.340}$	$^{IS}_{GR}\mathbf{0.868}$	$^{ISH}0.345$	$^{ISH}_{GR}0.932$
Adaptive LADR	-	-	$^{ISH}_{GR}\mathbf{0.738}$	$^{ISH}_{GR}\mathbf{0.872}$	-	-	$^{ISH}_{GR}\mathbf{0.739}$	$^{ISH}_{GR}\mathbf{0.900}$	-	-	$^{ISH}_{GR}\mathbf{0.347}$	$^{ISH}_{GR}\mathbf{0.960}$
RetroMAE (Exh.)	0.699	0.806	0.699	0.806	0.701	0.839	0.701	0.839	0.375	0.981	0.375	0.981
IVF [I]	0.226	0.225	0.346	0.358	0.272	0.263	0.372	0.375	0.157	0.381	0.221	0.541
ScaNN [S]	0.468	0.502	0.525	0.588	0.486	0.509	0.555	0.606	0.275	0.665	0.312	0.769
HNSW [H]	-	-	0.630	0.720	-	-	0.673	0.798	-	-	0.338	0.874
GAR [G]	0.559	0.553	0.696	0.763	0.578	0.604	0.692	0.789	**0.357**	0.750	0.368	0.890
Re-Ranking [R]	0.594	0.605	0.685	0.755	0.622	0.667	0.696	0.805	0.355	0.748	0.369	0.868
Proactive LADR	$^{IS}_{GR}\mathbf{0.691}$	$^{IS}_{GR}\mathbf{0.765}$	$^{ISH}_{GR}0.733$	$^{ISH}_{GR}0.844$	$^{IS}_{GR}\mathbf{0.702}$	$^{IS}_{GR}\mathbf{0.811}$	$^{ISH}_{G}0.723$	$^{IS}_{G}0.846$	$^{IS}0.356$	$^{IS}_{GR}\mathbf{0.864}$	$^{ISH}0.368$	$^{ISH}_{GR}0.938$
Adaptive LADR	-	-	$^{ISH}_{R}\mathbf{0.740}$	$^{ISH}_{GR}\mathbf{0.866}$	-	-	$^{ISH}_{G}\mathbf{0.731}$	$^{ISH}_{GR}\mathbf{0.879}$	-	-	$^{ISH}_{GR}\mathbf{0.374}$	$^{ISH}_{GR}\mathbf{0.973}$
TCT-HNP (Exh.)	0.708	0.830	0.708	0.830	0.689	0.848	0.689	0.848	0.359	0.970	0.359	0.970
IVF [I]	0.340	0.366	0.437	0.469	0.369	0.383	0.470	0.522	0.219	0.527	0.276	0.687
ScaNN [S]	0.378	0.410	0.444	0.496	0.355	0.376	0.427	0.459	0.215	0.522	0.253	0.632
HNSW [H]	-	-	0.625	0.721	-	-	0.634	0.762	-	-	0.315	0.853
GAR [G]	0.546	0.547	0.687	0.755	0.569	0.598	0.678	0.797	0.342	0.733	0.354	0.878
Re-Ranking [R]	0.586	0.605	0.679	0.755	0.614	0.667	0.685	0.805	0.342	0.748	0.353	0.868
Proactive LADR	$^{IS}_{GR}\mathbf{0.680}$	$^{IS}_{GR}\mathbf{0.747}$	$^{ISH}_{GR}0.719$	$^{ISH}_{GR}0.827$	$^{IS}_{GR}\mathbf{0.682}$	$^{IS}_{GR}\mathbf{0.803}$	$^{ISH}_{G}0.709$	$^{ISH}0.841$	$^{IS}_{G}\mathbf{0.346}$	$^{IS}_{GR}\mathbf{0.856}$	$^{ISH}0.354$	$^{ISH}_{GR}0.927$
Adaptive LADR	-	-	$^{ISH}\mathbf{0.729}$	$^{ISH}_{GR}\mathbf{0.848}$	-	-	$^{ISH}_{GR}\mathbf{0.721}$	$^{ISH}_{GR}\mathbf{0.878}$	-	-	$^{ISH}_{GR}\mathbf{0.359}$	$^{ISH}_{GR}\mathbf{0.962}$
ANCE (Exh.)	0.617	0.755	0.617	0.755	0.634	0.777	0.634	0.777	0.330	0.957	0.330	0.957
IVF [I]	0.358	0.395	0.441	0.500	0.407	0.437	0.498	0.549	0.212	0.530	0.268	0.703
ScaNN [S]	0.374	0.405	0.433	0.488	0.440	0.495	0.535	0.614	0.262	0.691	0.287	0.783
HNSW [H]	-	-	0.606	0.737	-	-	0.635	0.790	-	-	0.311	0.897
GAR [G]	0.527	0.540	0.648	0.750	0.568	0.622	0.655	0.794	**0.326**	0.751	0.329	0.888
Re-Ranking [R]	0.578	0.605	0.653	0.755	0.602	0.667	**0.674**	0.805	0.325	0.748	**0.333**	0.868
Proactive LADR	$^{IS}_{GR}\mathbf{0.645}$	$^{IS}_{GR}\mathbf{0.751}$	$^{IS}0.657$	$^{ISH}0.800$	$^{IS}_{GR}\mathbf{0.660}$	$^{IS}_{GR}\mathbf{0.807}$	$^{IS}0.666$	$^{IS}0.822$	$^{IS}0.321$	$^{IS}_{GR}\mathbf{0.872}$	$^{ISH}0.327$	$^{ISH}_{GR}0.932$
Adaptive LADR	-	-	$^{ISH}\mathbf{0.665}$	$^{ISH}\mathbf{0.820}$	-	-	$^{ISH}0.665$	$^{IS}\mathbf{0.830}$	-	-	$^{ISH}0.329$	$^{ISH}_{GR}\mathbf{0.959}$


Proactive LADR
Adaptive LADR
nDCG
k neighbors
4 .62 .66 .70 .72 .63 .64 .66 .67 .69
8 .65 .69 .72 .72 .64 .66 .68 .70 .71
16 .69 .71 .73 .73 .66 .68 .70 .71 .72
32 .72 .74 .73 .73 .68 .69 .71 .72 .72
64 .73 .74 .73 .72 .70 .71 .72 .72 .72
128 .74 .73 .73 .72 .74 .74 .74 .73 .72
R@1k
k neighbors
4 .65 .71 .78 .82 .65 .68 .71 .74 .77
8 .71 .76 .81 .84 .68 .72 .76 .79 .81
16 .77 .81 .84 .85 .72 .75 .79 .83 .85
32 .81 .85 .85 .85 .75 .78 .82 .85 .85
64 .85 .86 .85 .85 .79 .81 .85 .85 .84
128 .86 .86 .85 .85 .85 .86 .87 .86 .85
RBO
k neighbors
4 .65 .72 .78 .81 .68 .70 .74 .77 .79
8 .71 .76 .81 .84 .72 .75 .81 .85 .87
16 .76 .79 .83 .86 .77 .80 .86 .89 .91
32 .80 .83 .86 .89 .80 .85 .90 .92 .93
64 .84 .87 .89 .90 .83 .88 .93 .95 .96
128 .87 .89 .91 .92 .88 .92 .95 .97 .98
Latency (ms/q)
k neighbors
4 3.8 5.1 8.3 12.0 4.6 4.7 4.9 5.4 6.6
8 4.2 5.9 9.4 13.6 4.6 4.8 5.2 6.0 7.8
16 4.8 6.8 11.0 16.9 4.5 4.7 5.3 6.4 8.8
32 5.8 8.2 14.4 26.6 4.8 5.2 6.3 8.1 12.2
64 7.6 11.1 20.4 42.2 5.3 5.9 7.8 10.6 17.8
128 10.2 15.9 34.9 69.8 5.9 7.0 9.9 15.1 27.7
100 200 500 1000
n seed set size
10 20 50 100 200
c exploration depth

	DL19		DL20		Dev (sm)	
Graph	nDCG	R@1k	nDCG	R@1k	RR@10	R@1k
Proactive LADR						
Exact	0.730	**0.850**	**0.722**	**0.857**	**0.345**	**0.932**
Approx.	$^{=}$0.731	0.845	$^{=}$0.720	0.849	*0.343	*0.916
BM25	$^{=}$**0.732**	0.835	$^{=}$0.720	0.853	*0.339	*0.883
Adaptive LADR						
Exact	0.738	**0.872**	0.739	**0.900**	**0.347**	0.960
Approx.	$^{=}$0.736	0.861	$^{=}$0.737	$^{=}$**0.900**	$^{=}$**0.347**	***0.966**
BM25	**0.743**	0.859	$^{=}$**0.742**	**0.900**	*0.345	*0.933

	DL19 (valid.) $c = 100$			DL19 (valid.) $c = 1000$			DL20 (test) $c = 100$			DL20 (test) $c = 1000$		
Pipeline	nDCG	MAP	R@1k	nDCG	MAP	R@1k	nDCG	MAP	R@1k	nDCG	MAP	R@1k
BM25»MonoT5-base	0.665	0.417	0.755	0.699	0.483	0.755	0.672	0.421	0.805	0.711	0.498	0.805
w/ GAR$_{BM25}$	*0.697	*0.456	*0.786	0.727	0.490	*0.827	*0.695	0.439	*0.823	*0.743	**0.501**	*0.874
w/ GAR$_{TCT}$	* **0.722**	* **0.491**	* **0.800**	* **0.743**	**0.511**	* **0.839**	* **0.714**	* **0.472**	* **0.831**	* **0.749**	**0.501**	* **0.892**
BM25»MonoT5-3b	0.667	0.418	0.755	0.700	0.489	0.755	0.678	0.442	0.805	0.728	0.534	0.805
w/ GAR$_{BM25}$	*0.693	0.454	*0.790	*0.741	0.517	*0.831	*0.715	*0.469	*0.829	*0.772	0.556	*0.881
w/ GAR$_{TCT}$	* **0.715**	* **0.484**	* **0.806**	* **0.746**	**0.522**	* **0.846**	* **0.735**	* **0.512**	* **0.837**	* **0.787**	* **0.564**	* **0.899**
BM25»ColBERT	0.663	0.409	0.755	0.681	0.458	0.755	0.667	0.421	0.805	0.697	0.469	0.805
w/ GAR$_{BM25}$	*0.690	*0.442	*0.783	*0.720	0.480	*0.825	*0.695	*0.446	*0.823	*0.732	0.479	*0.870
w/ GAR$_{TCT}$	* **0.716**	* **0.475**	* **0.798**	* **0.727**	**0.482**	* **0.841**	* **0.707**	* **0.463**	* **0.829**	* **0.740**	**0.481**	* **0.887**
TCT»MonoT5-base	0.708	0.472	0.830	0.704	0.473	0.830	0.698	0.488	0.848	0.693	0.471	0.848
w/ GAR$_{BM25}$	* **0.728**	**0.484**	**0.852**	* **0.733**	**0.480**	* **0.883**	* **0.719**	* **0.501**	**0.861**	* **0.719**	**0.473**	* **0.881**
w/ GAR$_{TCT}$	0.722	0.481	0.847	*0.724	0.474	0.866	*0.712	0.494	0.856	*0.710	0.471	0.871
TCT»MonoT5-3b	0.720	0.498	0.830	0.725	0.513	0.830	0.723	0.534	0.848	0.733	0.544	0.848
w/ GAR$_{BM25}$	* **0.748**	* **0.521**	* **0.857**	* **0.759**	**0.521**	* **0.885**	* **0.743**	**0.546**	* **0.864**	* **0.771**	* **0.555**	* **0.890**
w/ GAR$_{TCT}$	*0.742	*0.517	0.849	*0.749	0.516	*0.868	*0.741	*0.545	*0.861	*0.759	0.551	*0.880
TCT»ColBERT	0.708	0.464	0.830	0.701	0.452	0.830	0.698	0.476	0.848	0.697	0.470	0.848
w/ GAR$_{BM25}$	* **0.729**	* **0.480**	**0.853**	* **0.727**	**0.459**	**0.876**	* **0.715**	**0.485**	**0.857**	* **0.722**	* **0.477**	* **0.877**
w/ GAR$_{TCT}$	*0.722	0.474	0.845	*0.715	0.452	0.852	*0.711	*0.484	* **0.857**	*0.713	0.473	0.864
D2Q»MonoT5-base	0.736	0.503	0.830	0.747	0.531	0.830	0.726	0.499	0.839	0.731	**0.508**	0.839
w/ GAR$_{BM25}$	*0.748	0.506	0.848	0.757	0.519	* **0.880**	*0.734	0.497	*0.847	**0.748**	0.504	*0.880
w/ GAR$_{TCT}$	* **0.760**	* **0.528**	**0.850**	* **0.766**	**0.533**	*0.879	**0.740**	**0.508**	* **0.856**	**0.748**	0.499	* **0.895**
D2Q»MonoT5-3b	0.737	0.506	0.830	0.751	0.542	0.830	0.738	0.531	0.839	0.753	0.557	0.839
w/ GAR$_{BM25}$	0.744	0.512	*0.850	**0.772**	**0.549**	* **0.880**	*0.751	0.535	*0.852	*0.781	0.561	*0.887
w/ GAR$_{TCT}$	**0.755**	**0.524**	* **0.857**	0.769	0.544	* **0.880**	* **0.764**	**0.550**	* **0.860**	* **0.790**	**0.565**	* **0.905**
D2Q»ColBERT	0.724	0.475	0.830	0.733	0.501	0.830	0.718	0.483	0.839	0.717	0.479	0.839
w/ GAR$_{BM25}$	0.734	0.484	0.845	**0.753**	**0.505**	*0.876	*0.731	0.487	*0.849	*0.737	0.482	*0.872
w/ GAR$_{TCT}$	* **0.744**	* **0.496**	**0.849**	*0.752	0.503	* **0.878**	* **0.735**	**0.488**	* **0.856**	* **0.746**	**0.485**	* **0.893**
SPLADE»MonoT5-base	0.750	0.506	0.872	0.737	**0.487**	0.872	0.748	0.505	0.899	0.731	**0.480**	0.899
w/ GAR$_{BM25}$	* **0.762**	0.509	**0.888**	**0.745**	**0.487**	**0.893**	* **0.757**	**0.509**	0.902	**0.737**	0.479	**0.909**
w/ GAR$_{TCT}$	*0.759	**0.512**	0.878	0.737	0.481	0.875	0.751	0.506	**0.903**	0.734	0.475	0.908
SPLADE»MonoT5-3b	0.761	0.526	0.872	0.764	**0.533**	0.872	0.774	0.559	0.899	0.775	0.560	0.899
w/ GAR$_{BM25}$	* **0.775**	0.532	* **0.891**	**0.774**	**0.533**	**0.896**	* **0.780**	0.559	0.903	* **0.788**	**0.562**	* **0.919**
w/ GAR$_{TCT}$	*0.773	**0.539**	0.884	0.769	0.531	0.881	* **0.780**	**0.561**	**0.905**	0.783	0.559	0.910
SPLADE»ColBERT	0.741	0.479	0.872	0.727	**0.456**	0.872	0.747	0.495	0.899	0.733	0.474	0.899
w/ GAR$_{BM25}$	* **0.753**	**0.490**	**0.885**	**0.730**	**0.456**	**0.875**	* **0.755**	**0.501**	0.902	* **0.742**	* **0.477**	**0.914**
w/ GAR$_{TCT}$	*0.750	0.489	0.876	0.727	0.455	0.868	*0.752	0.500	**0.903**	0.740	*0.476	0.911


rel
0
1
2
3
unjudged
final rank
100
80
60
40
20
0
unretrieved
0
100
200
300
400
500
initial rank



0.9
0.8
0.7
0.6
0.5
R@1k
nDCG
MAP
1 3 5 7 9 11 13 15
graph neighbours k
1 4 16 64 256
batch size b

	$\textsc{Gar}_{\text{TCT}}$		MonoT5-base
c	$b = 16$	$b = 64$	Scoring
100	2.68 ± 0.02	0.57 ± 0.01	267.06 ± 6.12
250	8.10 ± 0.05	4.34 ± 0.01	652.30 ± 7.53
500	17.38 ± 0.07	13.66 ± 0.02	$1,362.14 \pm 5.27$
750	26.96 ± 0.12	22.29 ± 0.07	$2,047.20 \pm 6.71$
1000	37.37 ± 0.07	30.82 ± 0.04	$2,631.75 \pm 6.28$

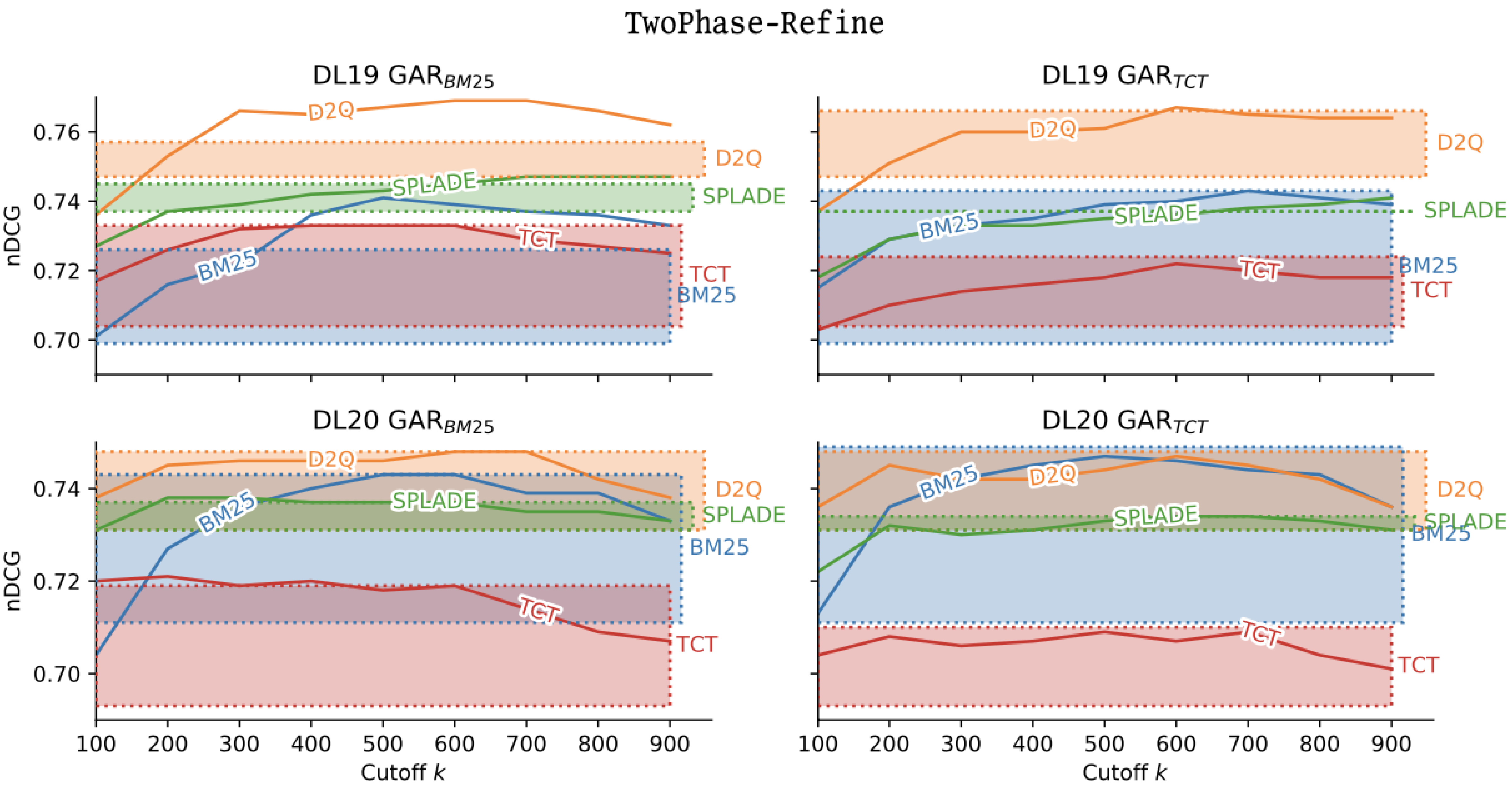
Table 6: Intra-List Similarity (ILS) among retrieved relevant documents. Since the set of retrieved documents does not change using typical Re-Ranking (RR), each value in this column is only listed once. ILS scores that are statistically equivalent to the RR setting are indicated with * (procedure described in Section 6.5).

		GAR_{BM25}		GAR_{TCT}	
Pipeline	RR	c=100	c=1k	c=100	c=1k
BM25»MonoT5-base	0.947	* 0.946	* 0.946	* 0.947	* 0.946
BM25»MonoT5-3b		* 0.946	* 0.946	* 0.946	* 0.946
BM25»ColBERT		* 0.946	* 0.946	* 0.947	* 0.946
TCT»MonoT5-base	0.969	* 0.969	* 0.968	* 0.969	* 0.969
TCT»MonoT5-3b		* 0.969	* 0.968	* 0.969	* 0.969
TCT»ColBERT		* 0.969	* 0.969	* 0.969	* 0.969
D2Q»MonoT5-base	0.969	* 0.968	* 0.968	* 0.969	* 0.968
D2Q»MonoT5-3b		* 0.968	* 0.968	* 0.968	* 0.968
D2Q»ColBERT		* 0.968	* 0.968	* 0.969	* 0.968
SPLADE»MonoT5-base	0.969	* 0.968	* 0.968	* 0.969	* 0.969
SPLADE»MonoT5-3b		* 0.968	* 0.968	* 0.968	* 0.969
SPLADE»ColBERT		* 0.968	* 0.969	* 0.969	* 0.969

		TREC DL 2019 (dev)				TREC DL 2020 (test)			
		GAR_{bm25}		GAR_{tct}		GAR_{bm25}		GAR_{tct}	
Pipeline	Agent	nDCG	R@1k	nDCG	R@1k	nDCG	R@1k	nDCG	R@1k
BM25»MonoT5	Non-Adaptive	0.699	0.755	0.699	0.755	0.711	0.805	0.711	0.805
	Oracle	0.747	0.804	0.786	0.853	0.748	0.791	0.768	0.828
	`Alternate`	0.726	^{N}0.827	^{NO}0.743	^{N}0.839	^{N}0.743	^{NO}**0.874**	^{N}**0.749**	^{N}**0.892**
	`TwoPhase-Fixed`	^{N}0.729	^{N}0.815	^{NO}0.740	^{N}0.836	^{N}0.732	^{NA}0.838	^{N}0.742	^{NA}0.858
	`TwoPhase-Refine`	^{N}0.741	^{N}0.826	^{NO}0.743	^{N}0.841	^{N}0.743	^{NO}0.871	^{NA}0.744	^{NA}0.879
	`Threshold`	^{N}**0.742**	^{N}**0.829**	^{NOA}**0.751**	^{N}**0.849**	^{N}**0.744**	^{NO}**0.874**	^{N}0.744	^{NA}0.874
	`Greedy`	0.723	^{N}0.823	^{NO}0.737	^{N}0.839	^{N}0.743	^{NO}0.868	^{N}0.744	^{N}0.882
TCT»MonoT5	Non-Adaptive	0.704	0.830	0.704	0.830	0.693	0.848	0.693	0.848
	Oracle	0.793	0.891	0.766	0.846	0.762	0.874	0.754	0.861
	`Alternate`	^{NO}**0.733**	^{N}0.883	^{NO}0.724	^{N}0.866	^{NO}**0.719**	^{N}0.881	^{NO}**0.710**	^{N}**0.871**
	`TwoPhase-Fixed`	^{NO}**0.733**	^{N}0.874	^{NO}0.719	^{N}0.857	^{NO}0.717	^{N}0.877	^{NO}**0.710**	^{N}0.868
	`TwoPhase-Refine`	^{NO}**0.733**	^{N}0.882	^{NO}0.722	0.859	^{NO}**0.719**	^{N}**0.883**	^{NOA}0.707	^{A}0.866
	`Threshold`	^{NO}0.731	^{N}**0.886**	^{NO}0.720	^{N}0.866	^{NOA}0.711	0.871	^{NOA}0.705	0.862
	`Greedy`	^{NO}0.731	^{N}0.881	^{NO}**0.725**	^{N}**0.871**	^{NOA}0.713	^{N}0.873	^{NO}0.708	^{N}0.868
D2Q»MonoT5	Non-Adaptive	0.747	0.830	0.747	0.830	0.731	0.839	0.731	0.839
	Oracle	0.797	0.867	0.798	0.867	0.791	0.884	0.793	0.889
	`Alternate`	0.757	^{N}**0.880**	^{NO}0.766	^{N}**0.879**	^{NO}**0.748**	^{N}**0.880**	^{O}**0.748**	^{N}**0.895**
	`TwoPhase-Fixed`	^{N}0.765	^{N}0.866	^{NO}0.765	^{N}0.870	^{NO}**0.748**	^{NA}0.867	^{O}0.745	^{NA}0.870
	`TwoPhase-Refine`	^{N}**0.769**	^{N}0.875	^{N}**0.767**	^{N}0.878	^{NO}**0.748**	^{N}0.877	^{O}0.747	^{N}0.892
	`Threshold`	^{N}0.766	^{N}0.876	^{NO}**0.767**	^{N}0.877	^{O}0.746	^{NA}0.874	^{O}0.745	^{N}0.881
	`Greedy`	0.754	^{N}0.874	^{O}0.757	^{N}0.873	^{O}0.744	^{N}0.878	^{O}**0.748**	^{N}0.894
SPLADE»MonoT5	Non-Adaptive	0.737	0.872	0.737	0.872	0.731	0.899	0.731	0.899
	Oracle	0.807	0.898	0.783	0.859	0.777	0.886	0.781	0.899
	`Alternate`	^{O}0.745	0.893	^{O}0.737	0.875	^{O}0.737	**0.909**	^{O}0.734	**0.908**
	`TwoPhase-Fixed`	^{O}0.763	0.863	**0.764**	0.869	**0.748**	^{A}0.868	^{O}0.742	^{NA}0.867
	`TwoPhase-Refine`	^{O}**0.769**	0.875	**0.764**	0.870	^{O}**0.748**	0.877	^{O}0.736	^{A}0.869
	`Threshold`	^{O}0.766	0.871	0.759	0.857	^{O}0.746	0.874	^{O}**0.744**	^{NA}0.865
	`Greedy`	^{NO}0.747	^{N}**0.895**	^{O}0.740	**0.882**	^{O}0.734	0.903	^{O}0.734	0.906

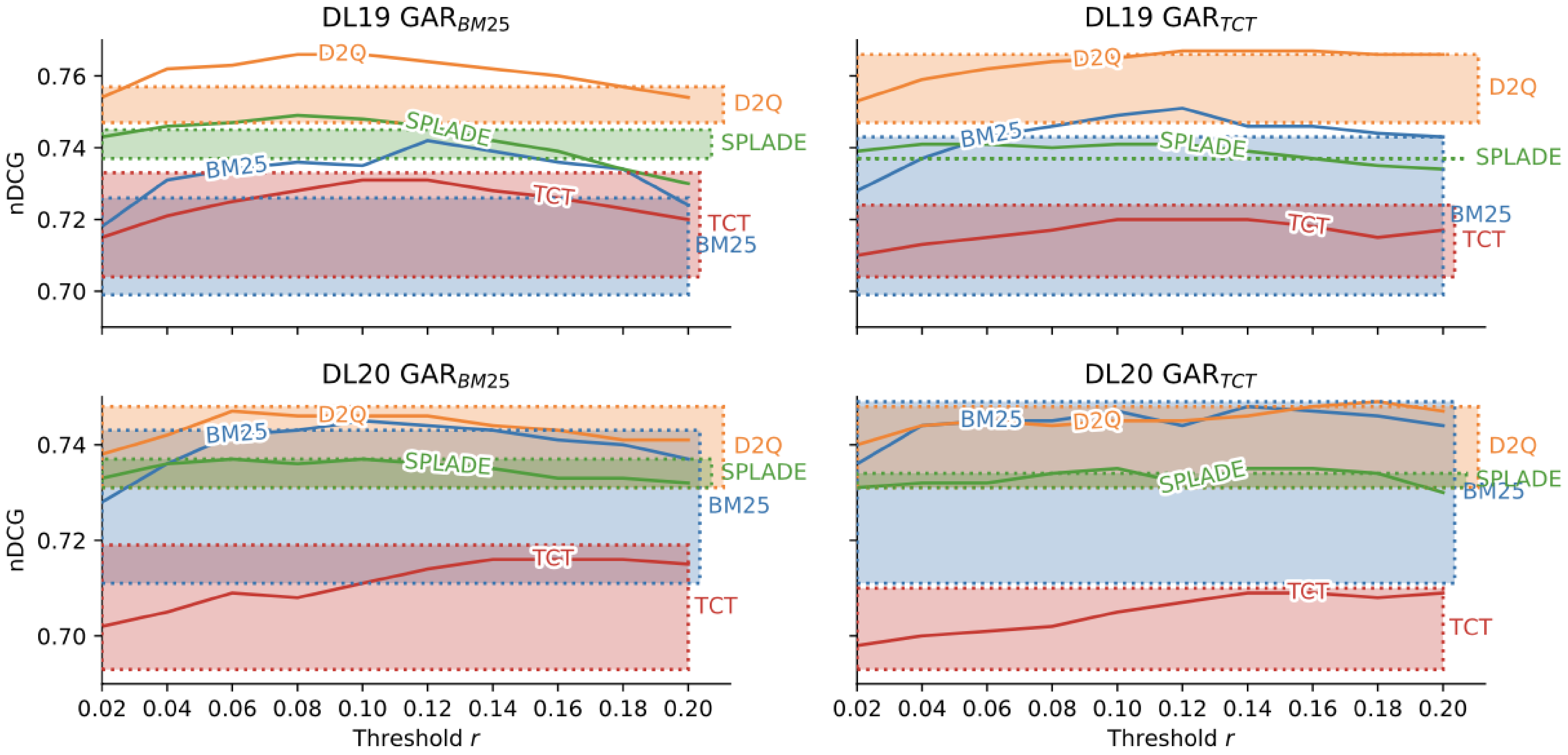
Table 1

Re-ranking performance on TREC Deep Learning 2019 and 2020 using various agents. The best-performing (non-oracle) agent in section is listed in bold. Significant differences compared to the Non-Adaptive, Oracle, and `Adaptive` systems are marked with ^{NOA}, respectively (paired t-test, $p < 0.05$).



TwoPhase-Refine
DL19 GAR$_{BM25}$
DL19 GAR$_{TCT}$
DL20 GAR$_{BM25}$
DL20 GAR$_{TCT}$
nDCG
nDCG
0.76
0.74
0.72
0.70
0.74
0.72
0.70
D2Q
SPLADE
BM25
TCT
100
200
300
400
500
600
700
800
900
Cutoff k
Cutoff k

Threshold



DL19 GAR$_{BM25}$
DL19 GAR$_{TCT}$
DL20 GAR$_{BM25}$
DL20 GAR$_{TCT}$
nDCG
Threshold r
D2Q
SPLADE
BM25
TCT
0.70
0.72
0.74
0.76
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16
0.18
0.20