Reaching the Pinnacle of Vector Search

A Journey to Optimize Neural Search Engine

현화림, 김인근
검색/데이터 플랫폼

CONTENTS

1. Vector Search & Search Engine

벡터 검색과 검색 엔진

1.1 Term-matching Search vs. Vector Search

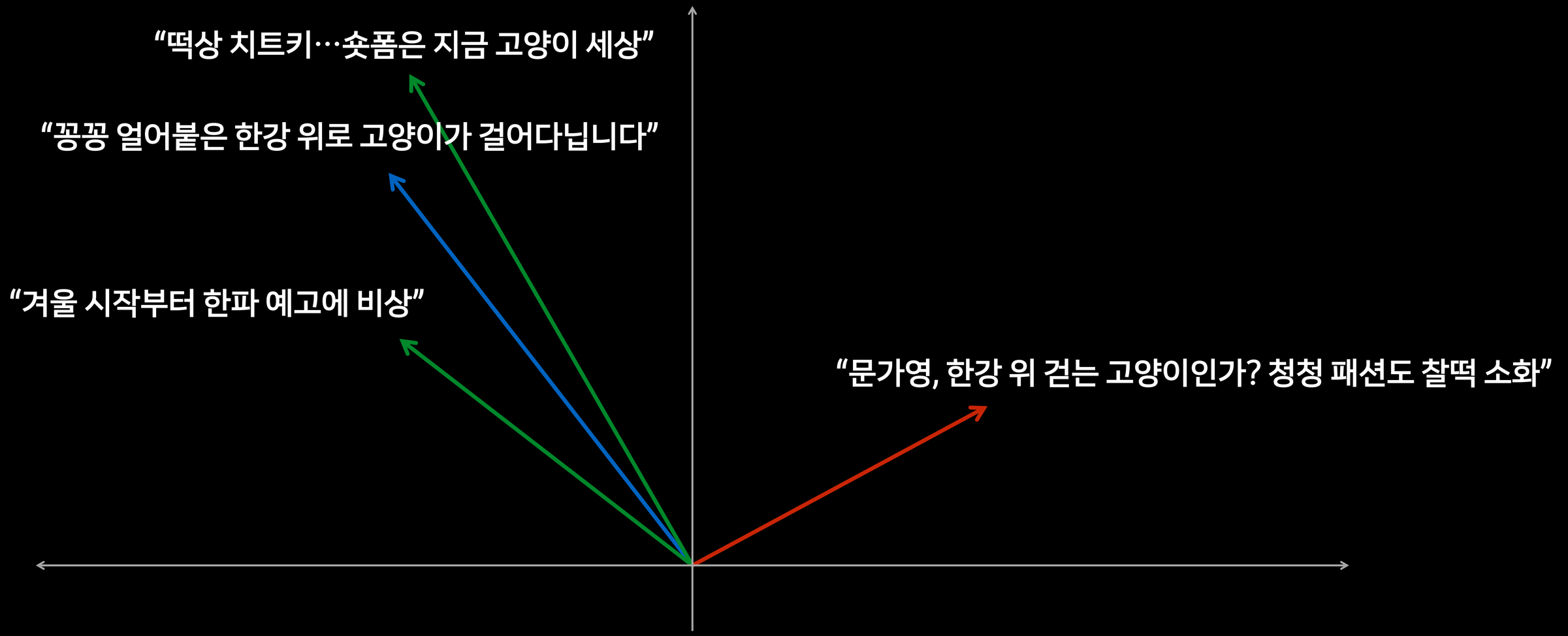
	Term-matching Search	Vector Search
“겨울 시작부터 한파 예고에 비상”		
“떡상 치트키…숏폼은 지금 고양이 세상”		
“문가영, 한강 위 걷는 고양이인가? 청청 패션도 찰떡 소화”		

1.1 Term-matching Search vs. Vector Search

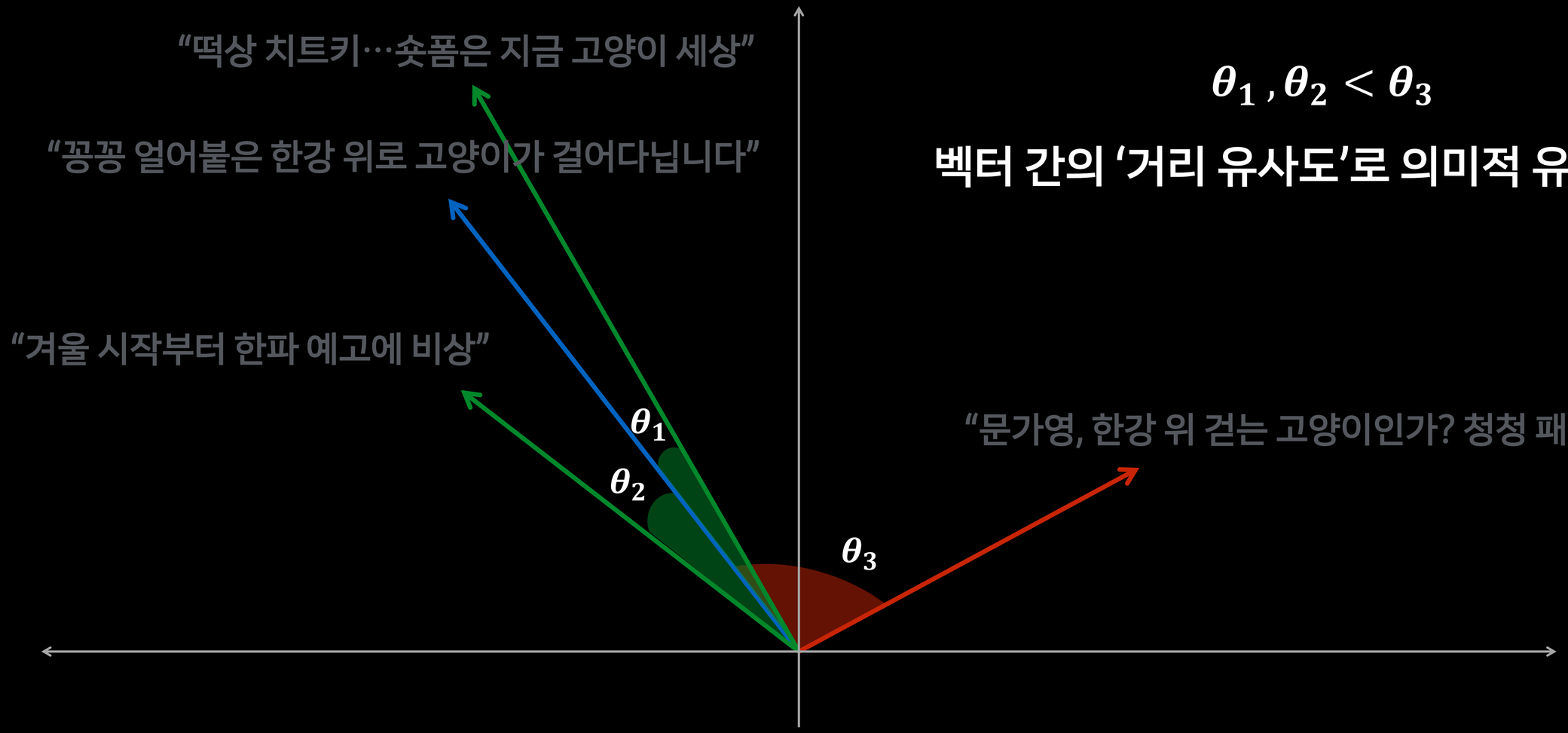
꽁꽁 얼어붙은 한강 위로 고양이가 걸어다닙니다

	Term-matching Search	Vector Search
“겨울 시작부터 한파 예고에 비상”	✗	
“떡상 치트키…숏폼은 지금 고양이 세상”	✓	
“문가영, 한강 위 걷는 고양이인가? 청청 패션도 찰떡 소화”	✓	

1.1 Term-matching Search vs. Vector Search



1.1 Term-matching Search vs. Vector Search



$\theta_1, \theta_2 < \theta_3$

벡터 간의 ‘거리 유사도’로 의미적 유사 파악

1.1 Term-matching Search vs. Vector Search

	Term-matching Search	Vector Search
“겨울 시작부터 한파 예고에 비상”	✗	✓
“떡상 치트키…숏폼은 지금 고양이 세상”	✓	✓
“문가영, 한강 위 걷는 고양이인가? 청청 패션도 찰떡 소화”	✓	✗

1.2 The Complexity of Vector Search

Exact Nearest Neighbor (ENN)

```
for (i=1부터 DB의 모든 점의 index N까지) {
  for (j=1부터 차원수 D까지) {
    score_i += query_j * DB_ij
  }
}
```

for 2중 루프

모든 DB vector access

Approximate Nearest Neighbor (ANN)

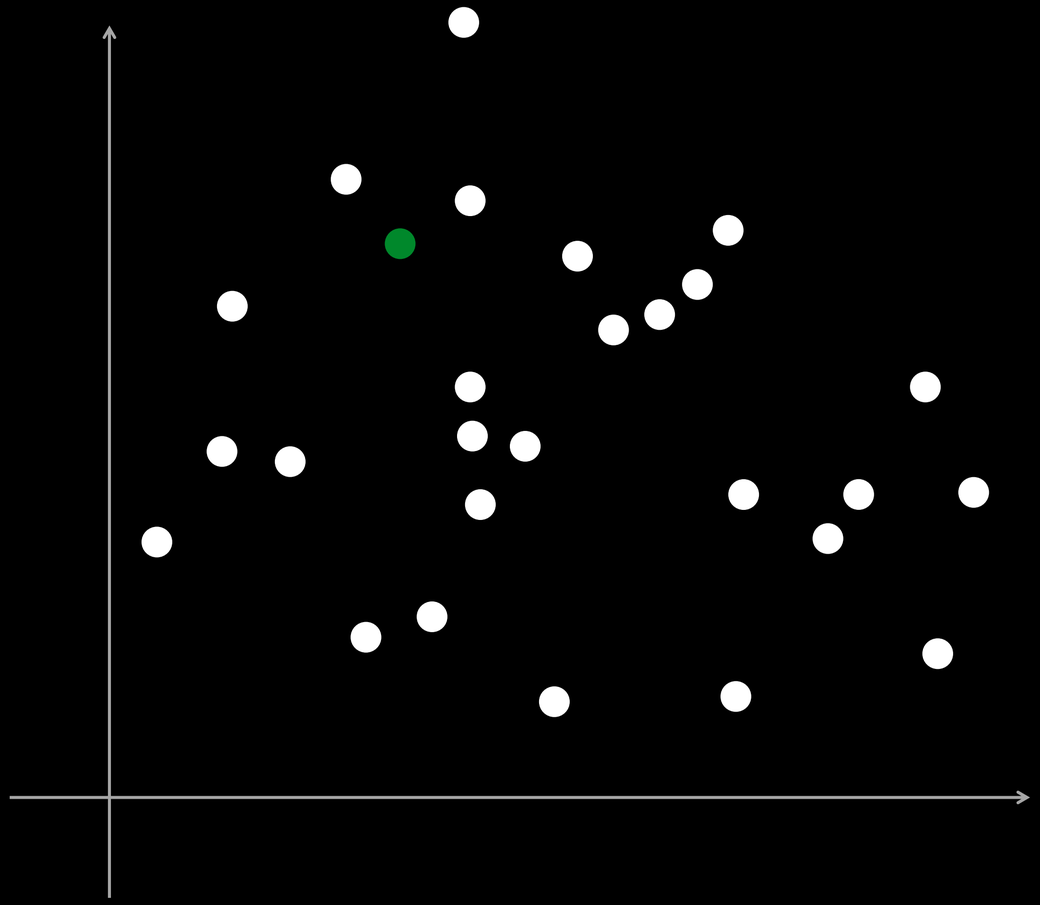
정확도⇣ 속도↑↑↑ **실서비스 가능**한 빠른 검색

분류 1. DB를 색인하여 query와
공간적으로 가까운 vector들로 검색 한정

분류 2. **그래프 탐색 문제**로 치환

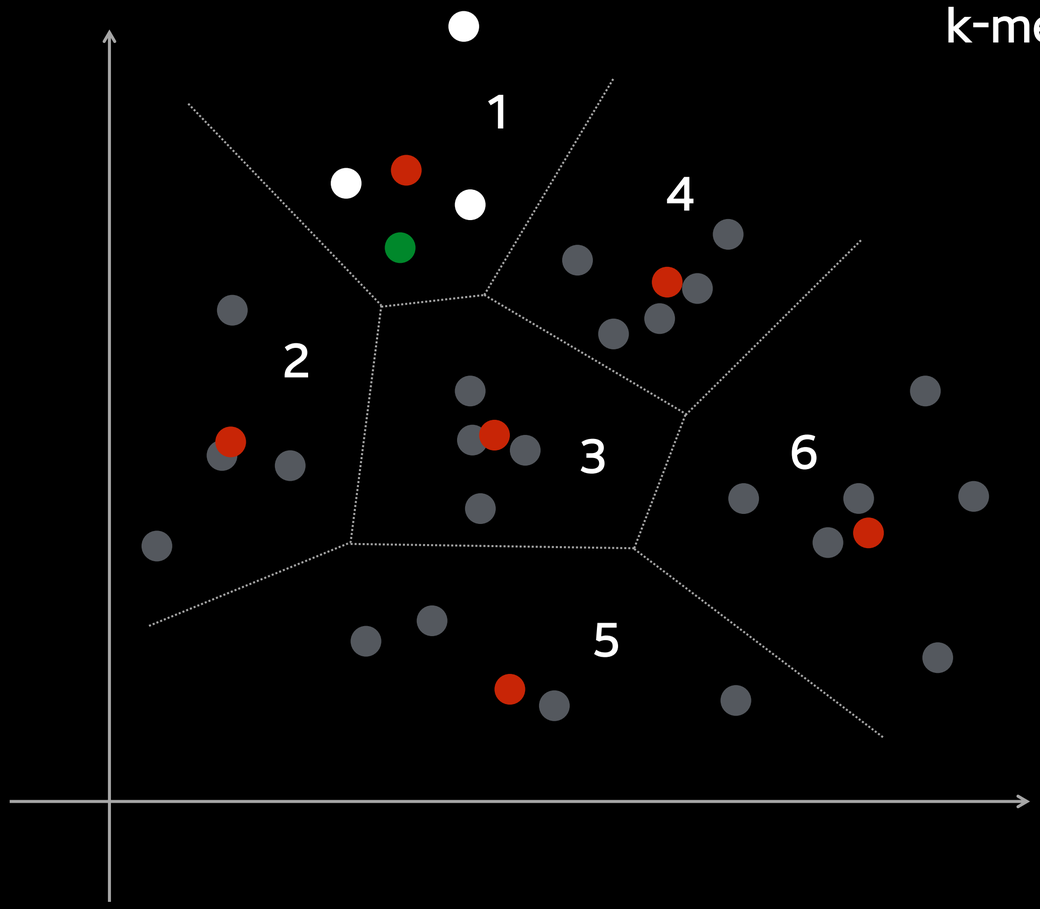
1.2.1 ANN Method: subspace based (IVF)

query vector DB vector



1.2.1 ANN Method: subspace based (IVF)

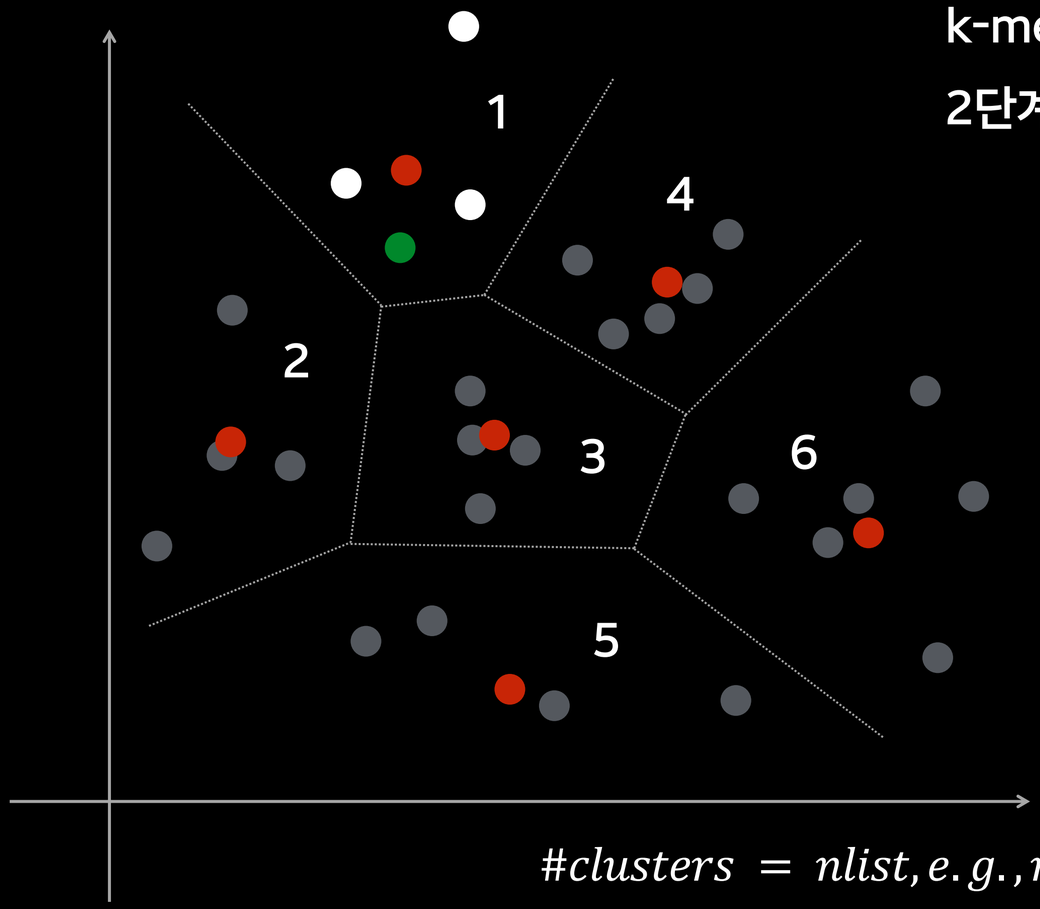
● query vector ● DB vector (interested in) ● coarse quantizer (c.q.)



k-means 알고리즘을 활용한 clustering 진행

1.2.1 ANN Method: subspace based (IVF)

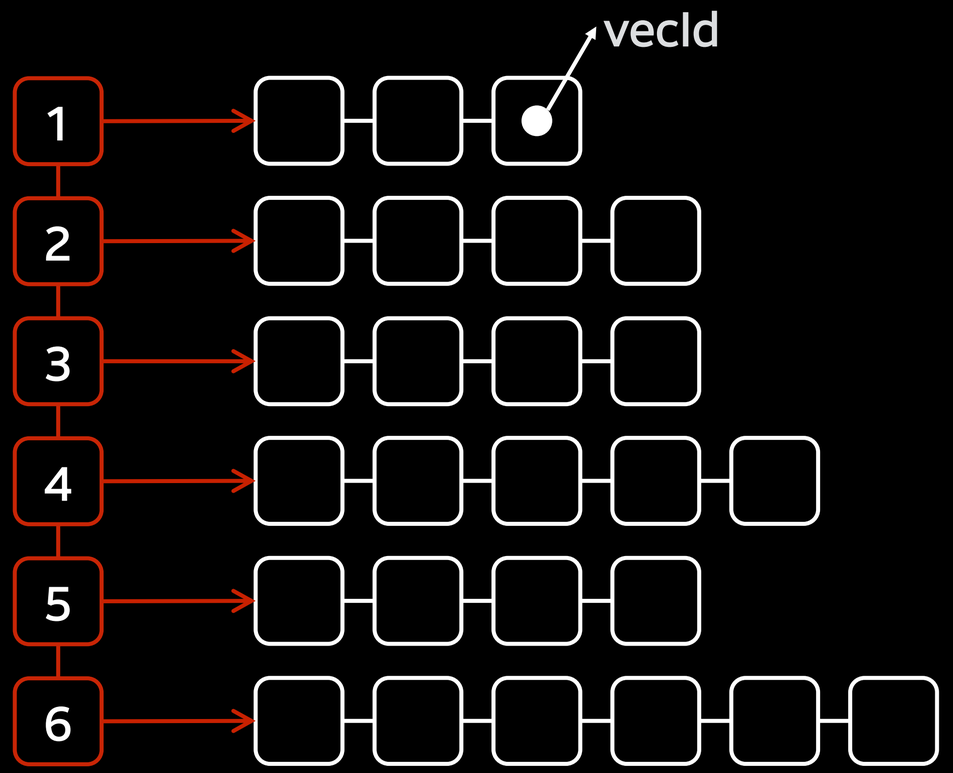
● query vector ● DB vector (interested in) ● coarse quantizer (c.q.)



$$\#clusters = nlist, e.g., nlist = 6$$

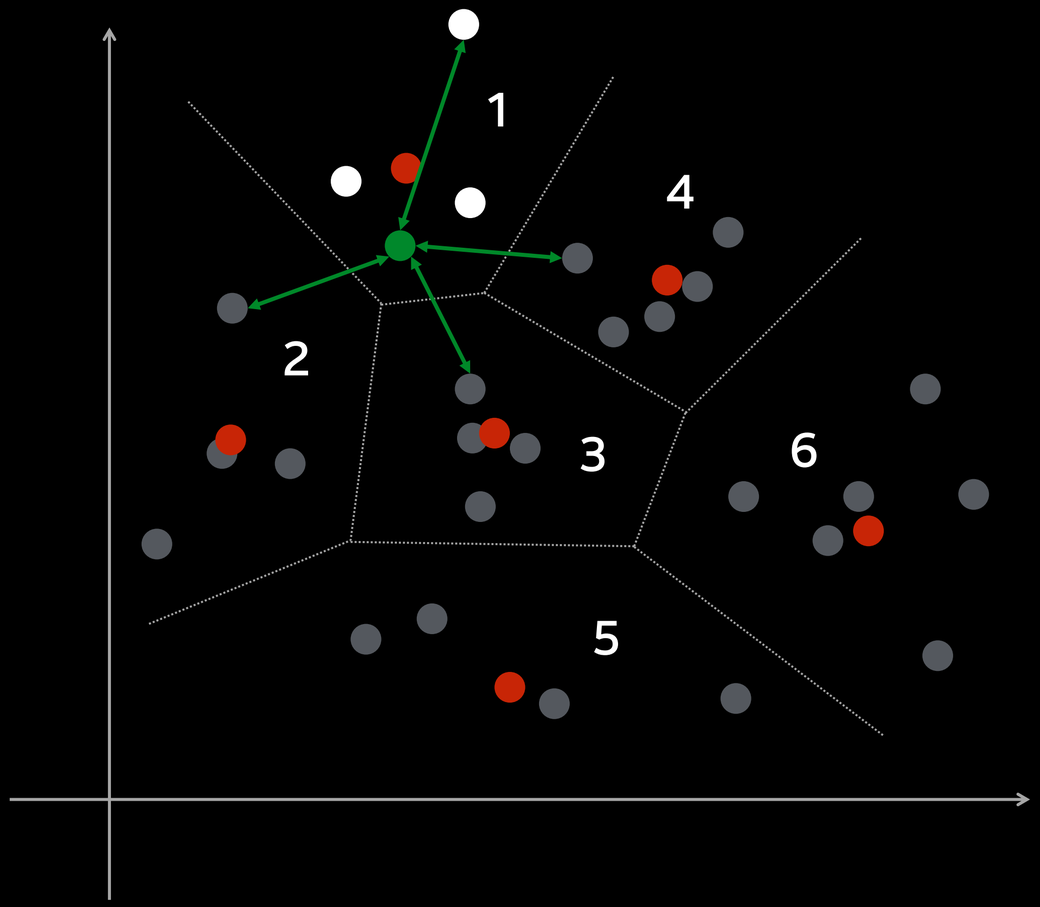
k-means 알고리즘을 활용한 clustering 진행

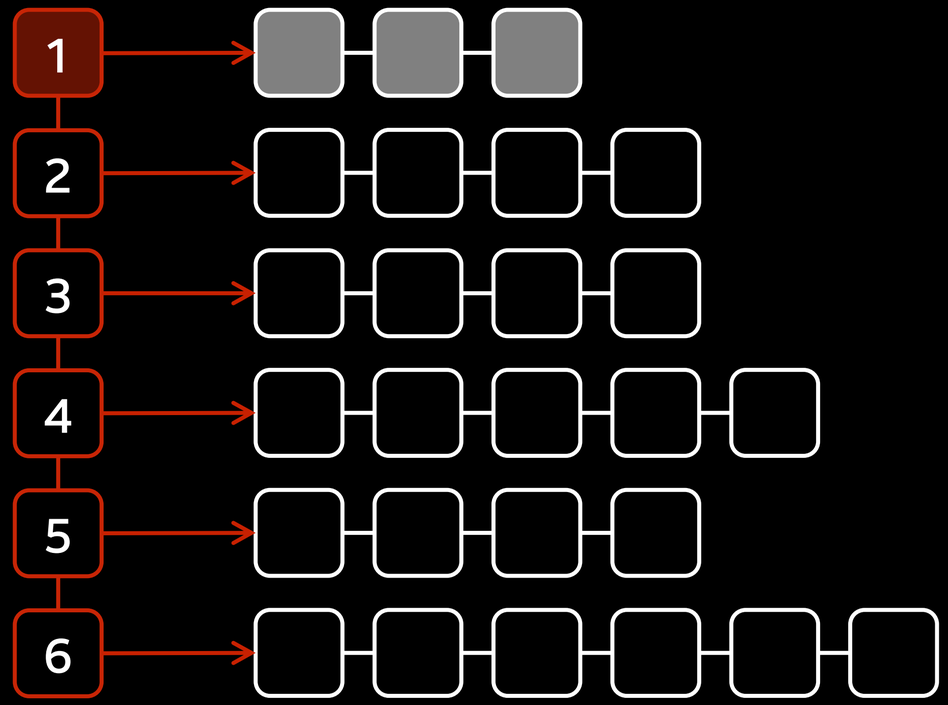
2단계 list 형태로 구현



1.2.1 ANN Method: subspace based (IVF)

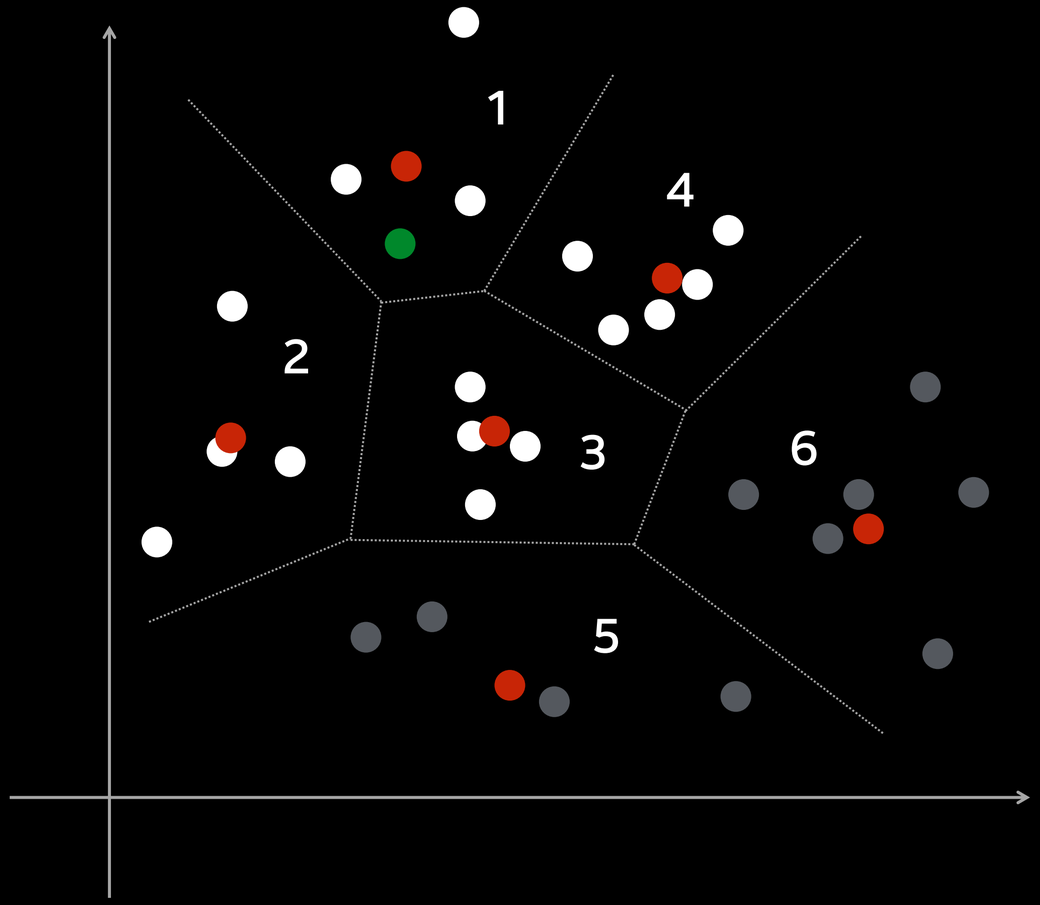
● query vector ● DB vector (interested in) ● coarse quantizer (c.q.)





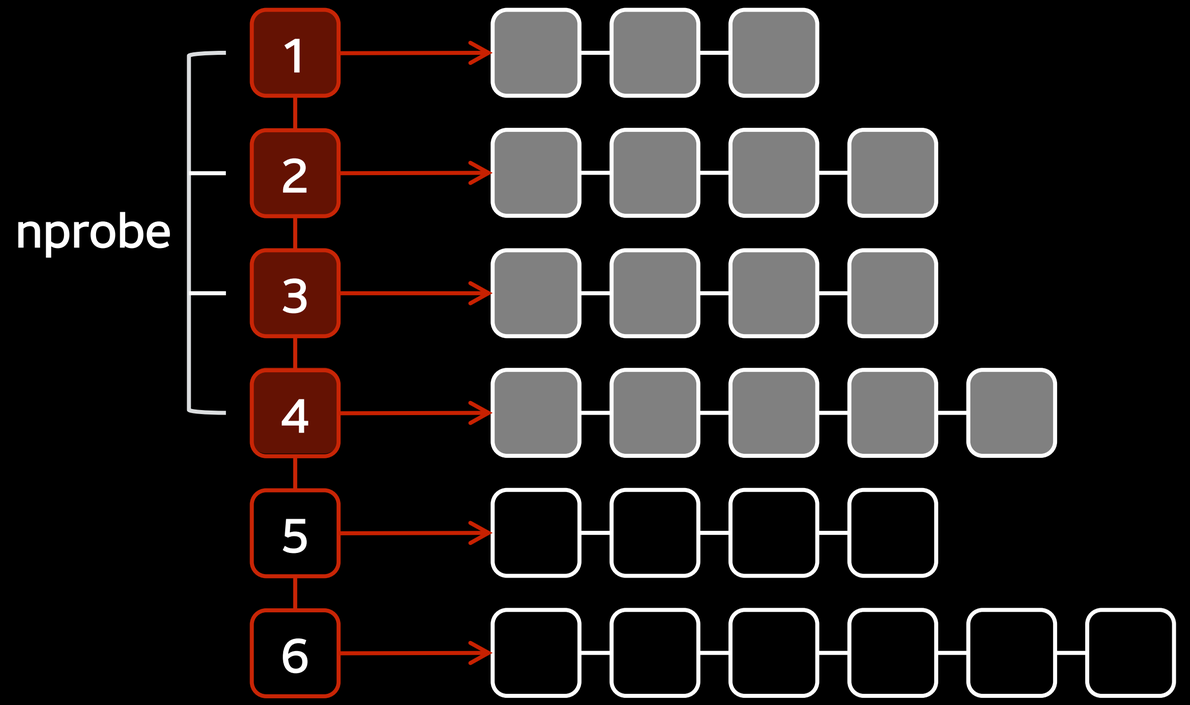
1.2.1 ANN Method: subspace based (IVF)

● query vector ● DB vector (interested in) ● coarse quantizer (c.q.)



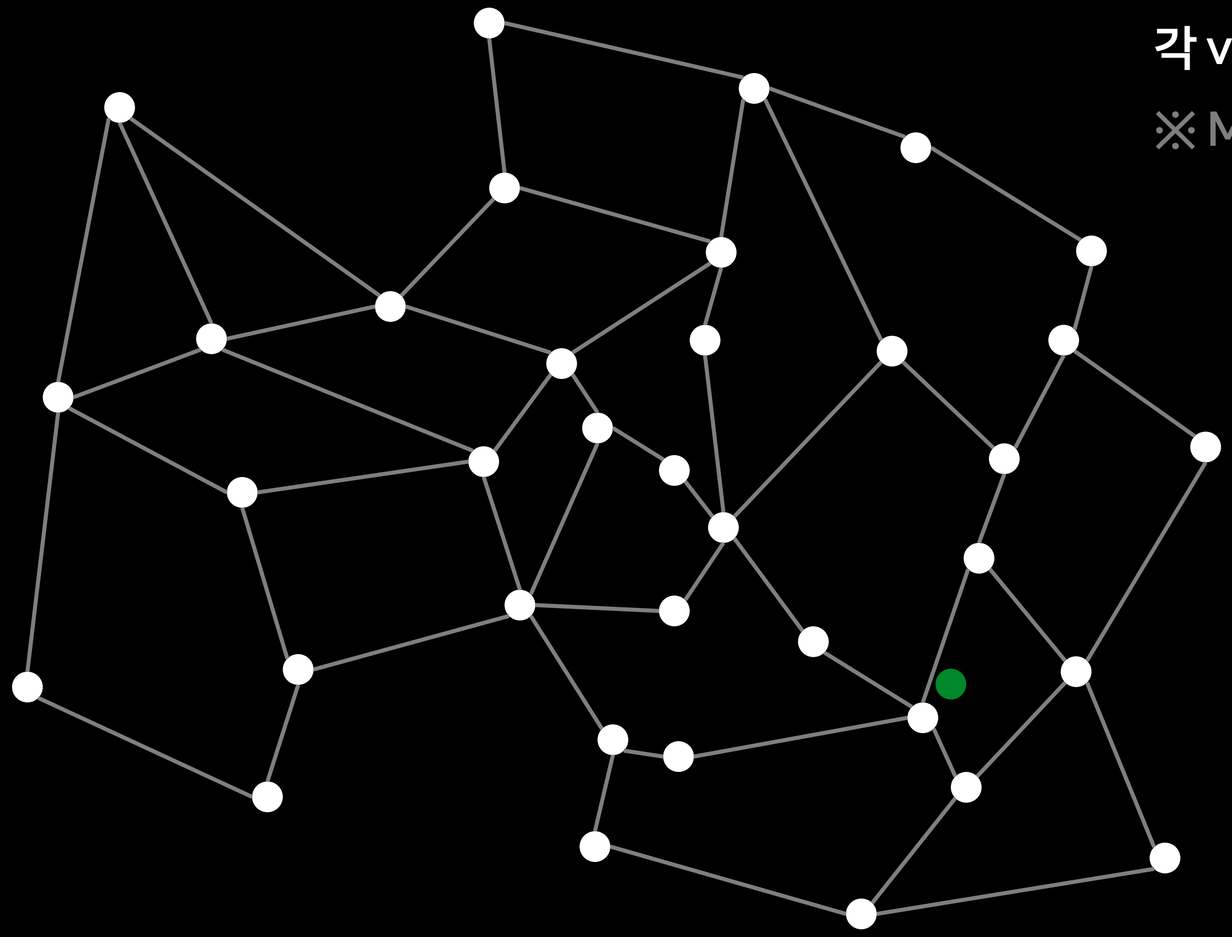
$nlist \propto \sqrt{(\#vector)}$

$nprobe$: 품질-성능 trade-off



1.2.2 ANN Method: graph based (basic)

● query vector ● DB vector

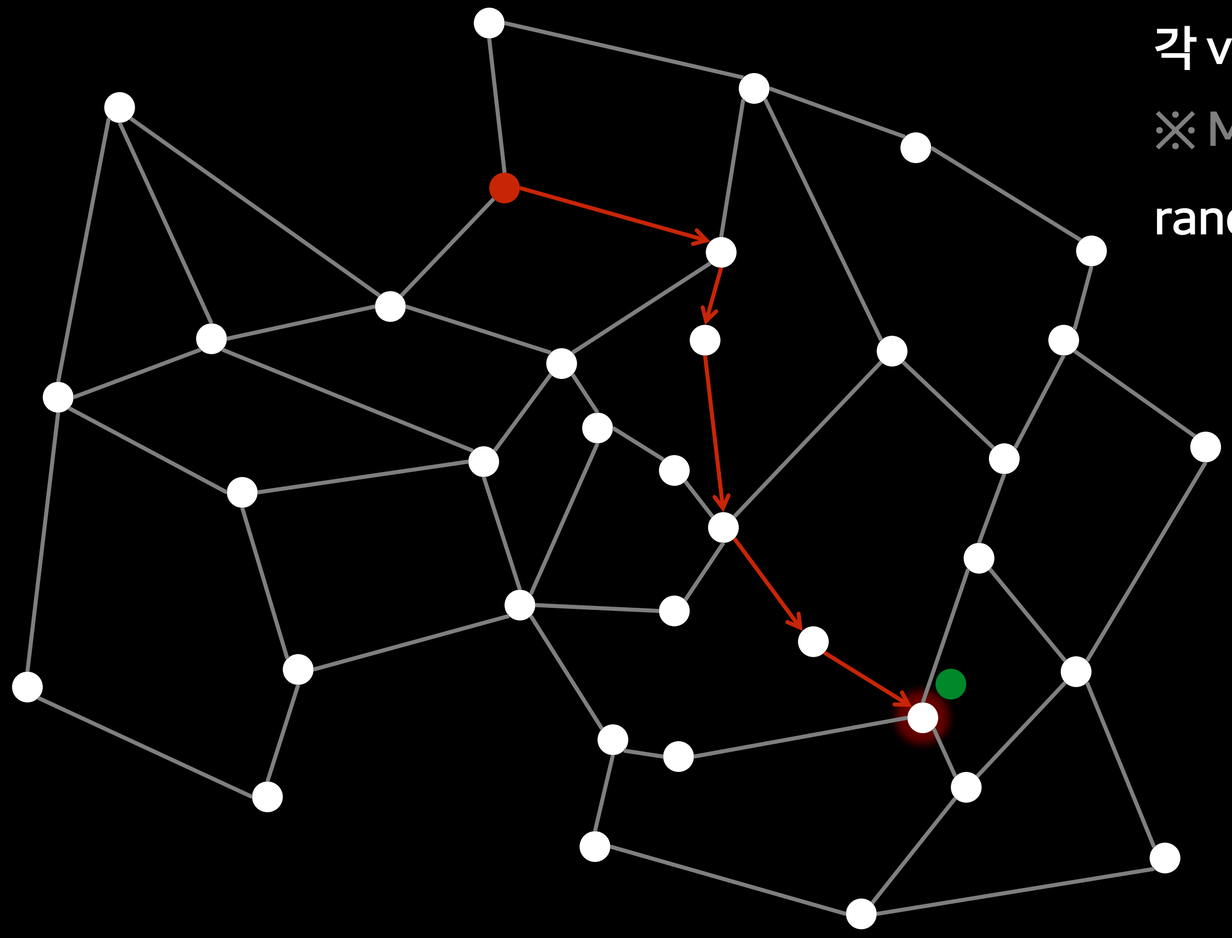


각 vector 마다 적당히 M개의 이웃 연결

※ M은 그래프 구축 시 활용되는 파라미터

1.2.2 ANN Method: graph based (basic)

● query vector ● DB vector ● starting point



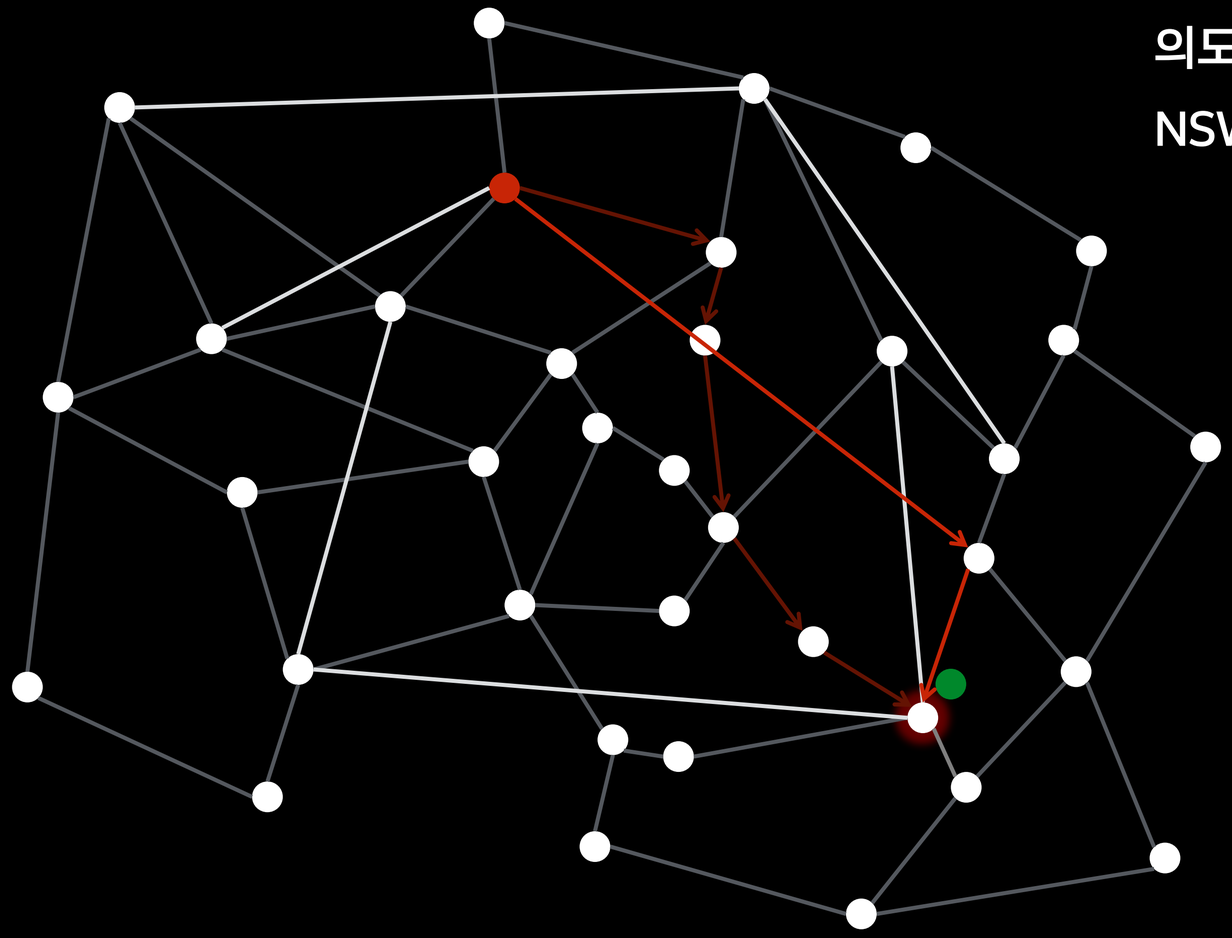
각 vector 마다 적당히 M개의 이웃 연결

※ M은 그래프 구축 시 활용되는 파라미터

random point에서 시작해 greedy 하게 query vector로 접근

1.2.2 ANN Method: graph based (NSW)

● query vector ● DB vector ● starting point



의도적으로 멀리 있는 이웃 연결, 탐색 속도 개선

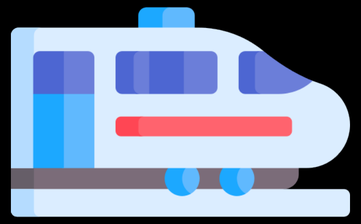
NSW = Navigable Small World

1.2.2 ANN Method: graph based (HNSW)

HNSW = Hierarchical NSW

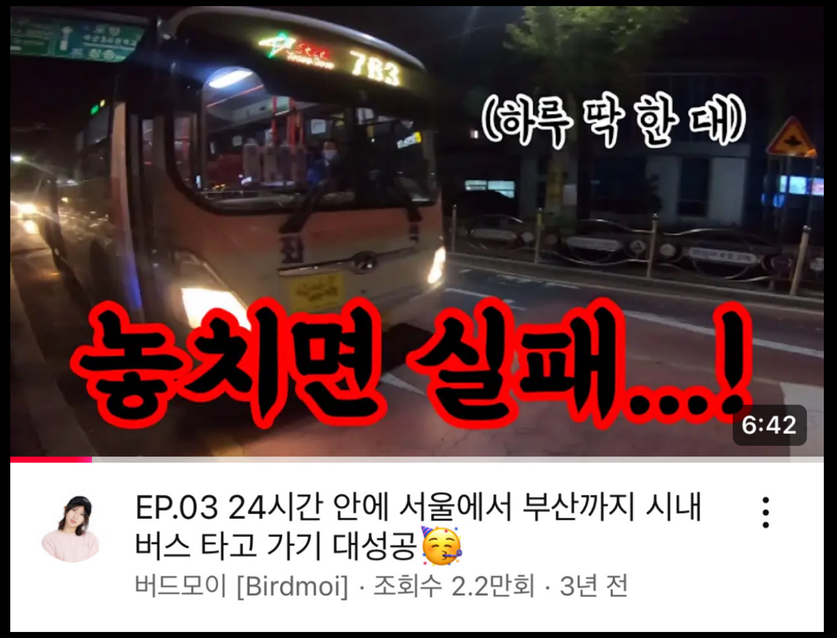
왜 hierarchy가 필요할까?

50분



3시간 20분

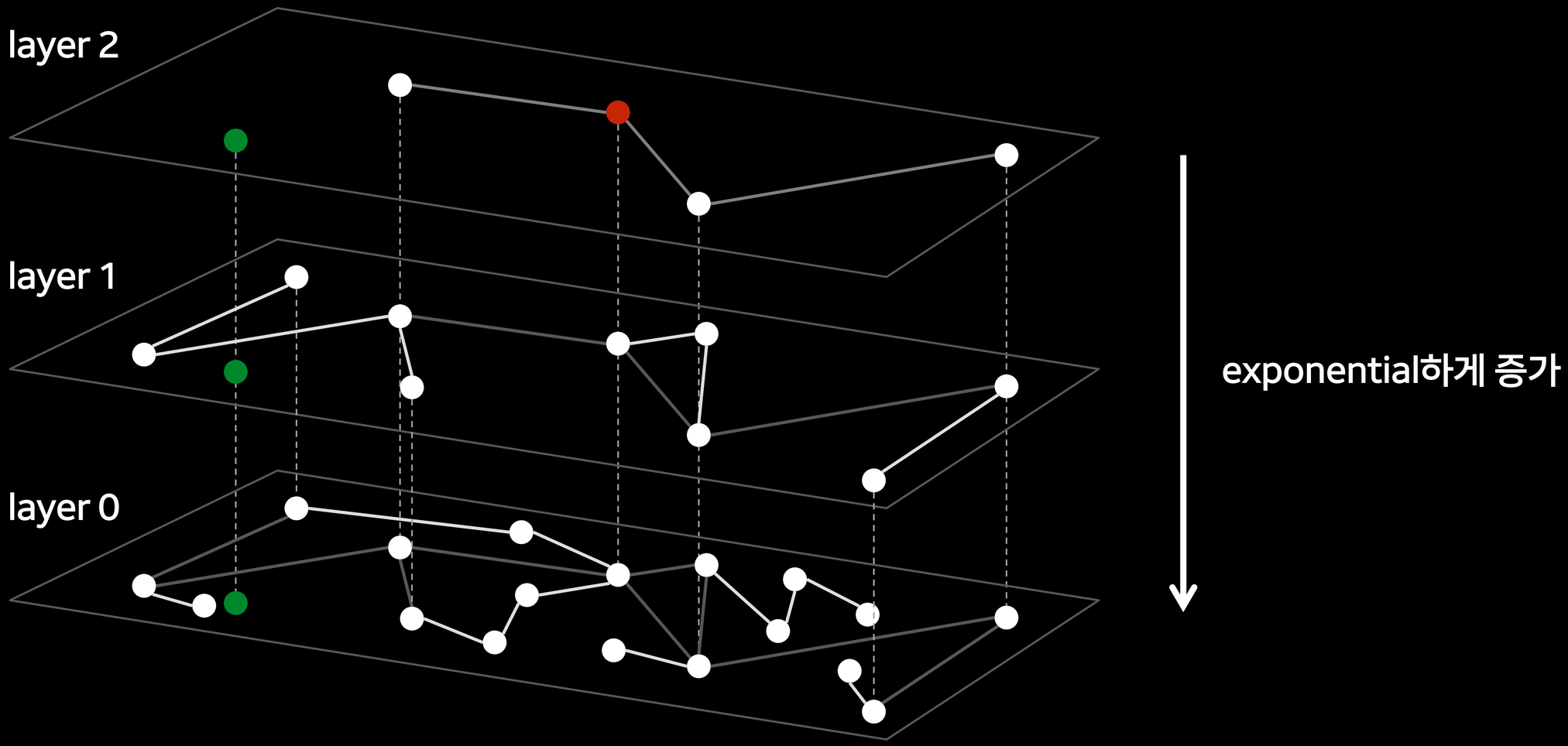
VS



23~28번의 환승, 24시간 챌린지

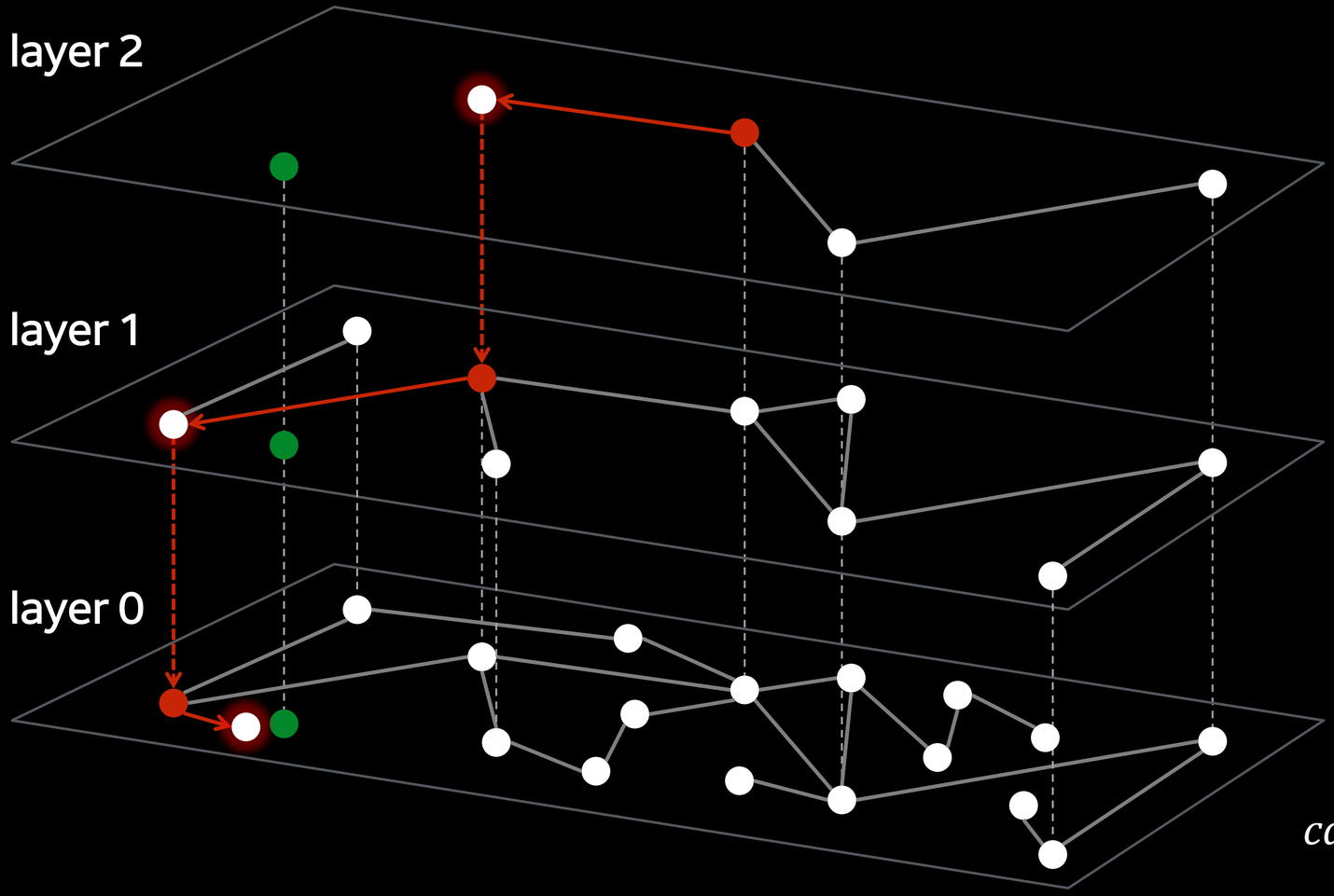
1.2.2 ANN Method: graph based (HNSW)

query vector DB vector starting point



1.2.2 ANN Method: graph based (HNSW)

● query vector ● DB vector ● starting point



topK = 최종적으로 찾아낼 가장 가까운 vector의 수

ef = 결과 벡터를 여러 개 찾기 위해 두는 중간 buffer의 크기

반드시 $topK \leq ef$

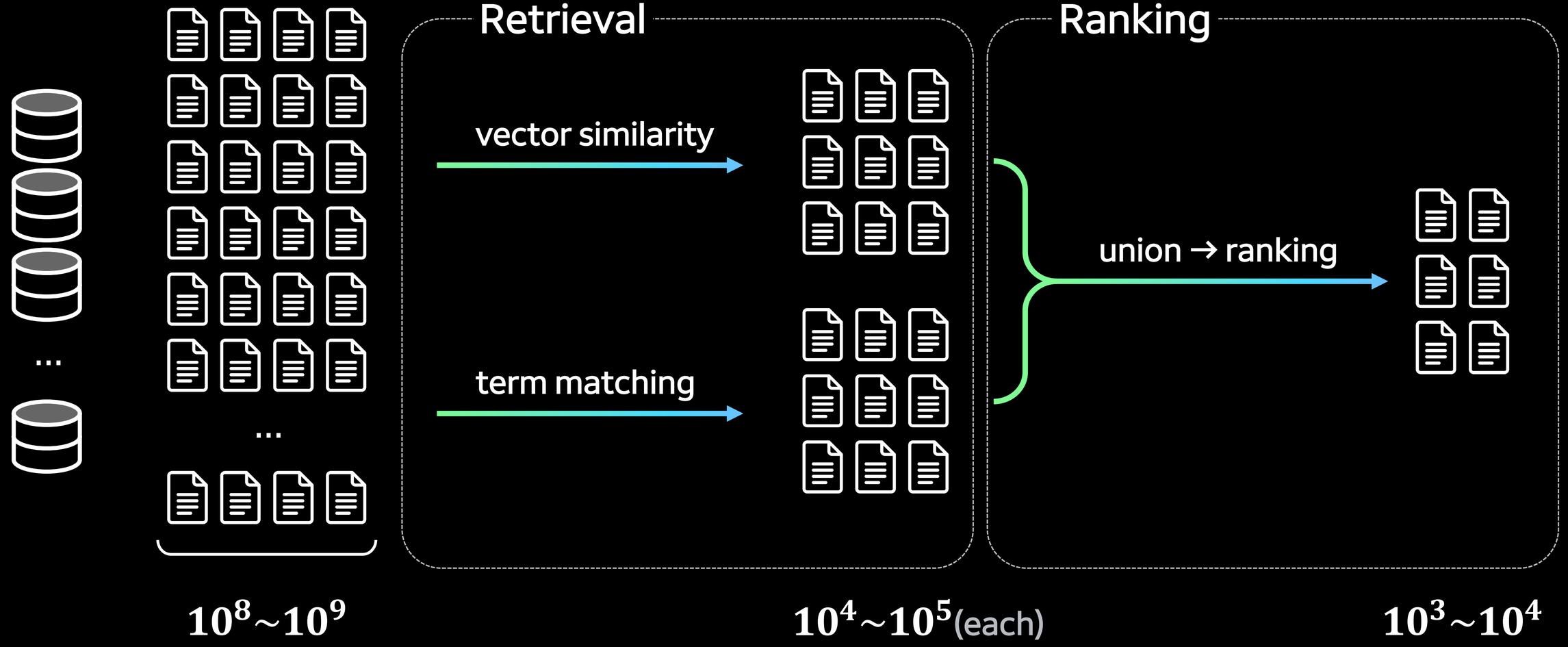
일반적인 경우 $topK \ll ef$ 활용

$case\ when\ topK = 1, ef = 1$

1.2.3 Comparison between IVF and HNSW

	IVF	HNSW
index size	small	big
query speed	slow	fast
build time	faster than HNSW	slow
search accuracy	moderate	high (95% recall@1)

1.3 Nexus++ Search Engine



2. ColBERT 무지출 챌린지

ColBERT 복습부터 무지출 챌린지까지

2.1 Our Scope

✓ ColBERT 무지출 챌린지

ColBERT 복습

Nexus++에 ColBERT를 적용하는 일의 복잡도

ColBERT ‘무지출 챌린지’ 의 의미

엔지니어링 관점에서의 해결 방법

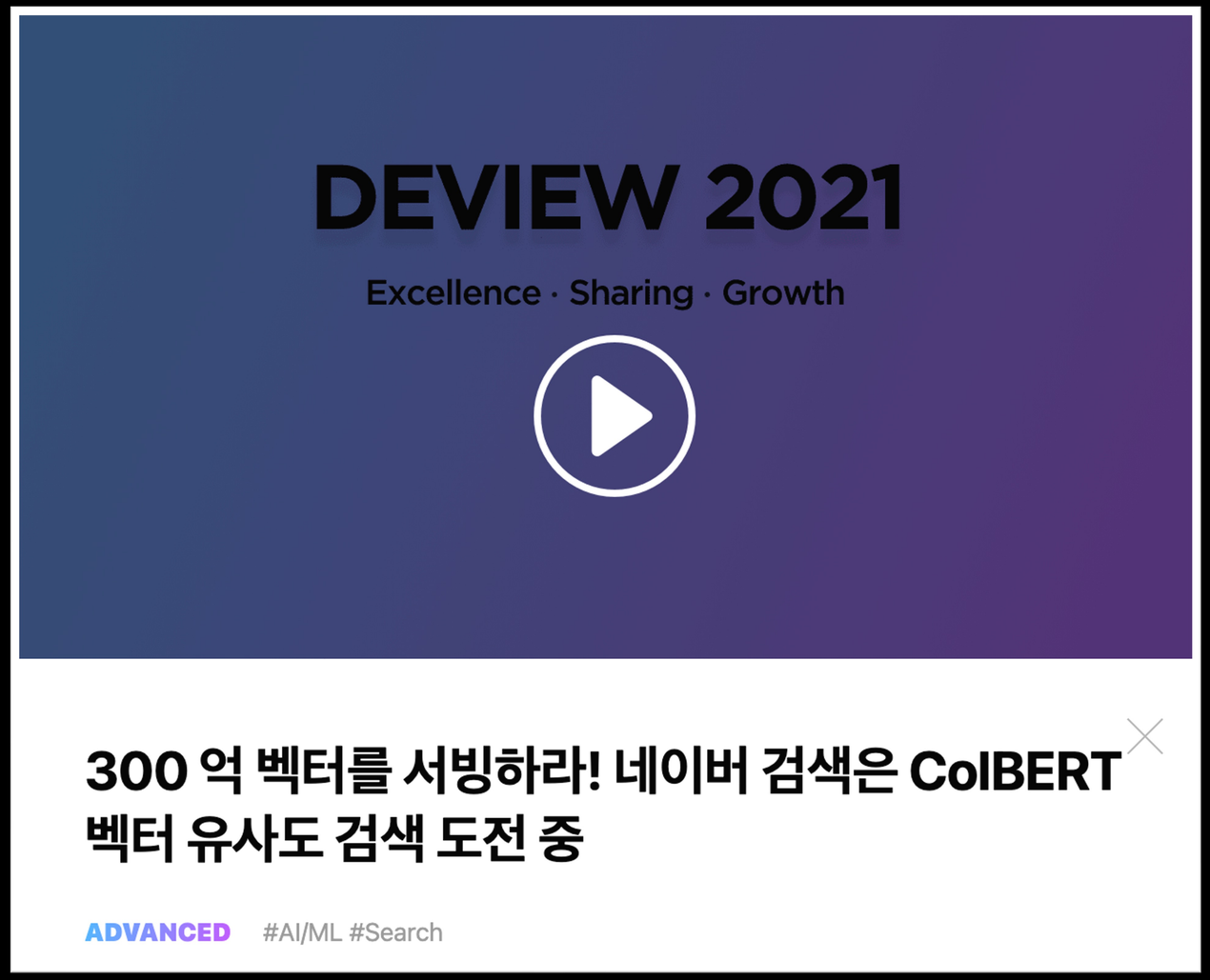
✗ AI model

Let the black box remain as a black box

✗ Scalability

분산 환경을 지원하기 위한 고민과 구현

2.2 Recapturing ColBERT



ColBERT = multi-vector search

※ 고차원 (100차원 가량)

1 doc → nn vectors

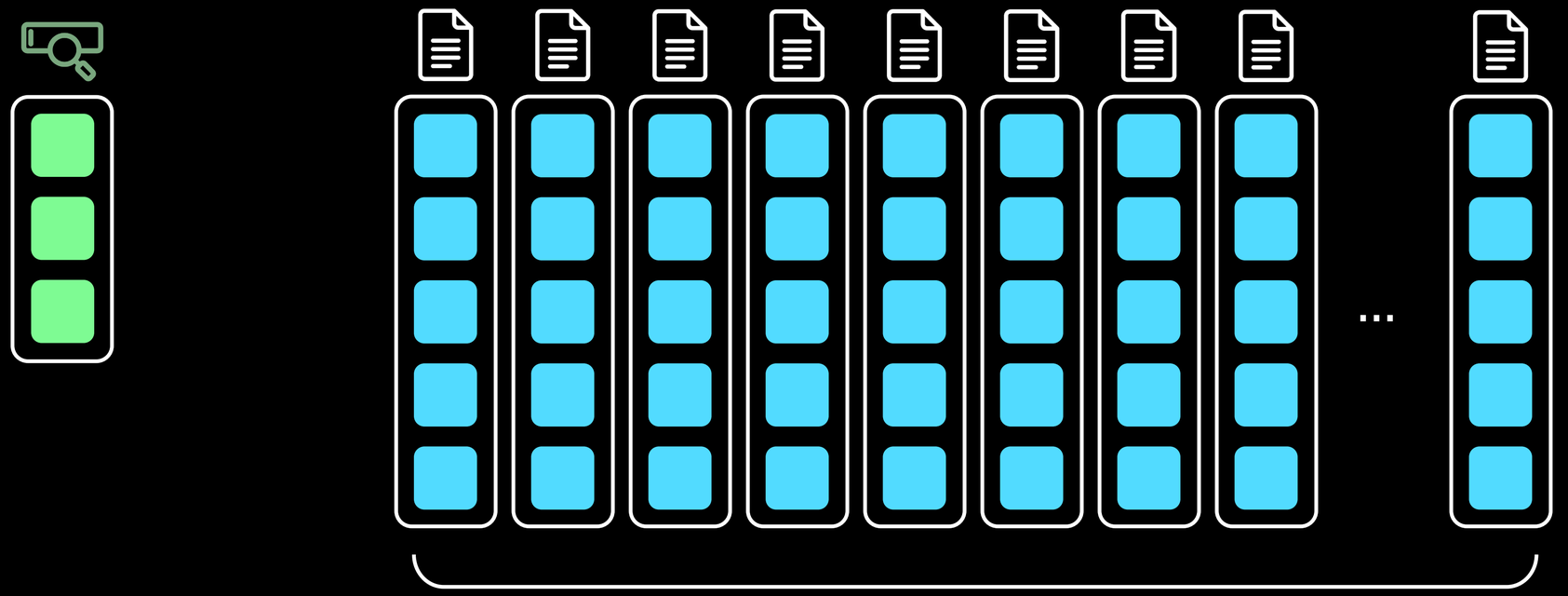
1 query → term-wise vectors

"코엑스 맛집 데이트"

$(vec1, vec2, vec3)$

=

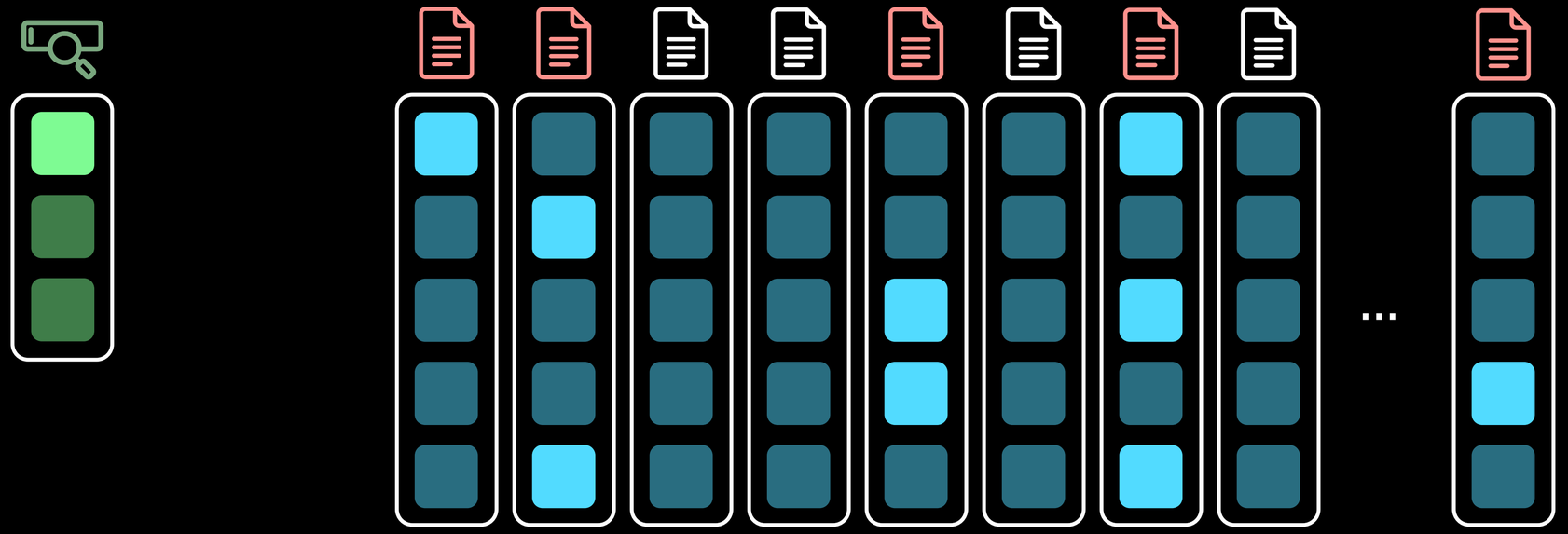
2.2.1 ColBERT Score



실제로는?

(N억 개 규모 문서) ✕ (문서 당 <=100 vectors)
3년 전에 300억 vector 였으니 지금은 500억+

2.2.1 ColBERT Score

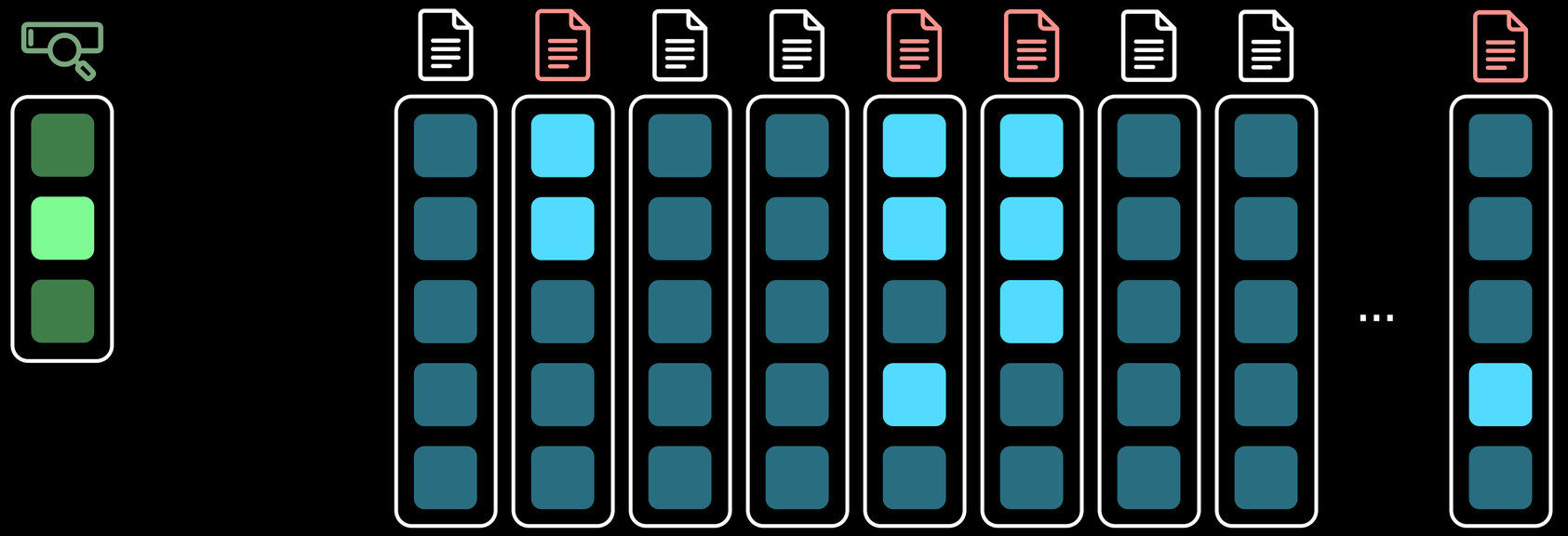


※ 문서 하나 당 5개의 vector

STEP 1. retrieval: element-wise ANN topK

※ 예시에서는 topK=9

2.2.1 ColBERT Score

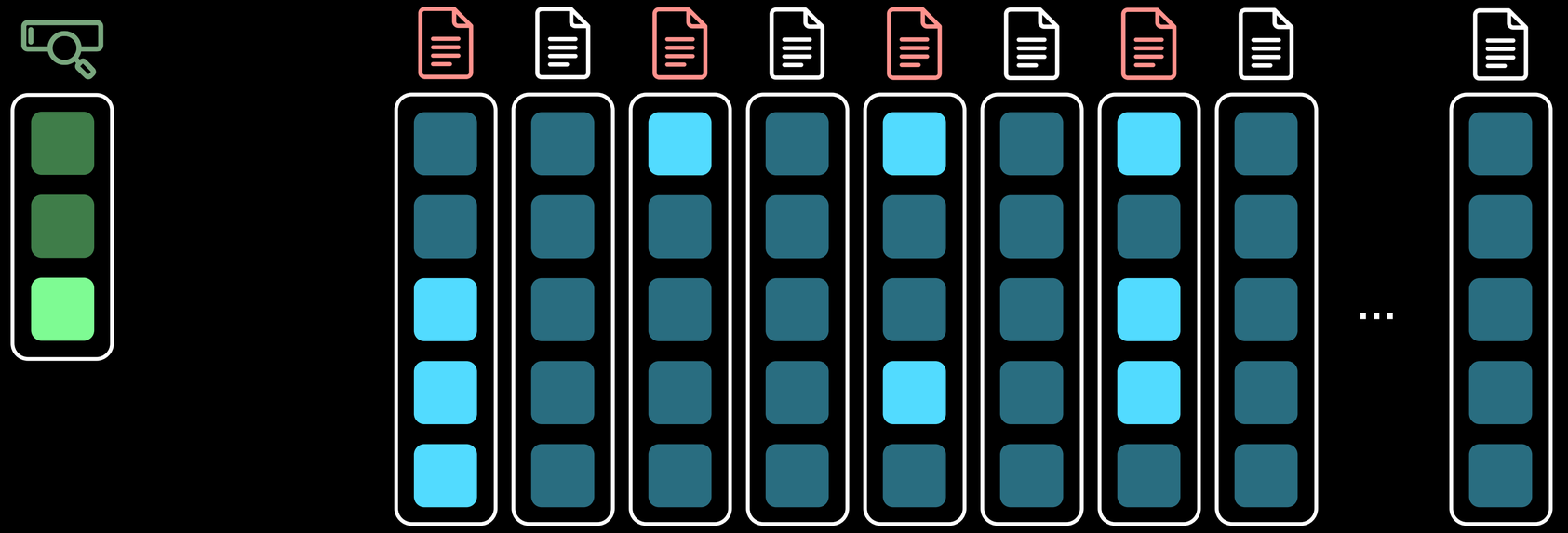


※ 문서 하나 당 5개의 vector

STEP 1. retrieval: element-wise ANN topK

※ 예시에서는 topK=9

2.2.1 ColBERT Score

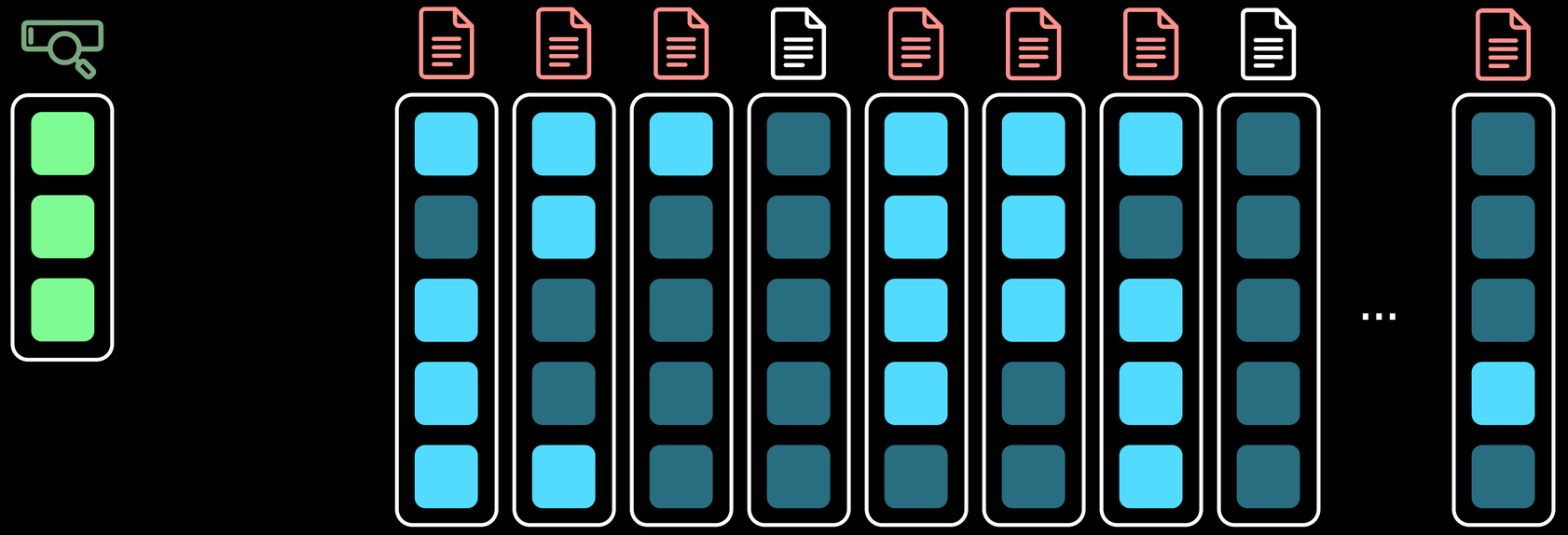


※ 문서 하나 당 5개의 vector

STEP 1. retrieval: element-wise ANN topK

※ 예시에서는 topK=9

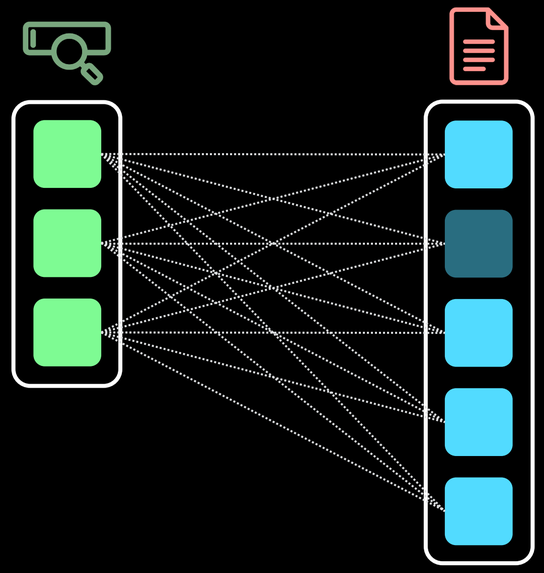
2.2.1 ColBERT Score



STEP 2. retrieval: 후보 문서 취합 (union)

※ 실제로는 < 0.01% 정도의 문서만 매칭

2.2.1 ColBERT Score



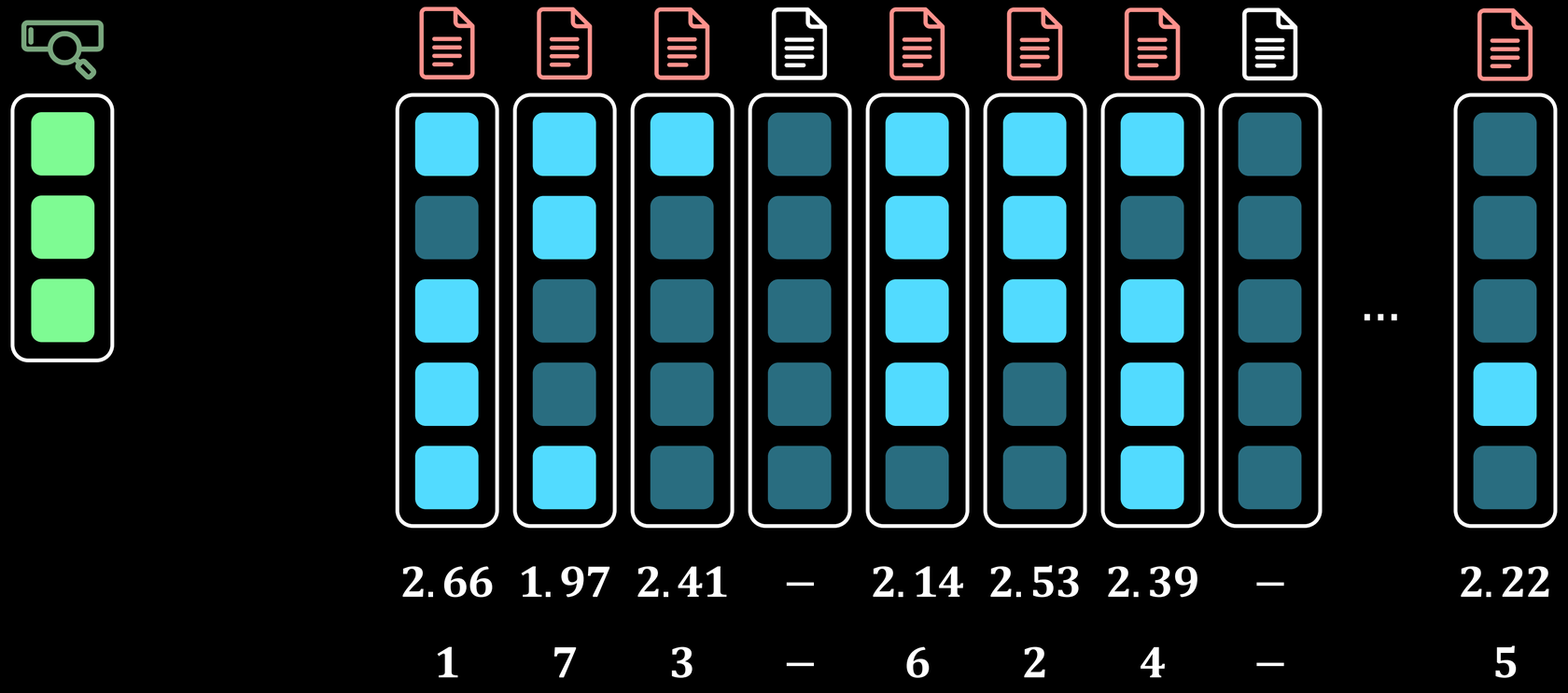
0.85	0.64	0.82
0.61	0.60	0.72
0.72	0.71	0.82
0.76	**0.84**	**0.97**
0.55	0.80	0.90

$$score(q, doc) = 0.85 + 0.84 + 0.97 = 2.66$$

Sum of Maximum Similarity

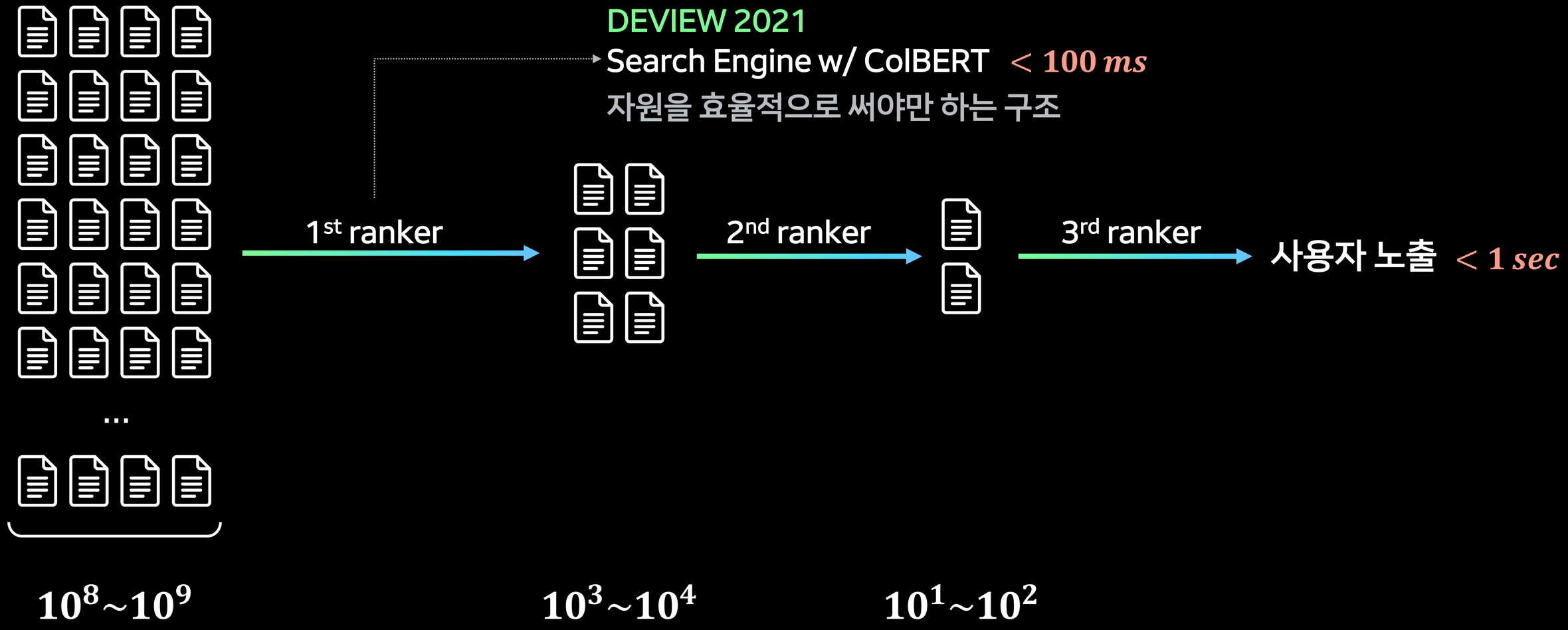
STEP 3. ranking: $(document\ vector) \times (query\ vector)$ interaction을 통한 score 계산

2.2.1 ColBERT Score

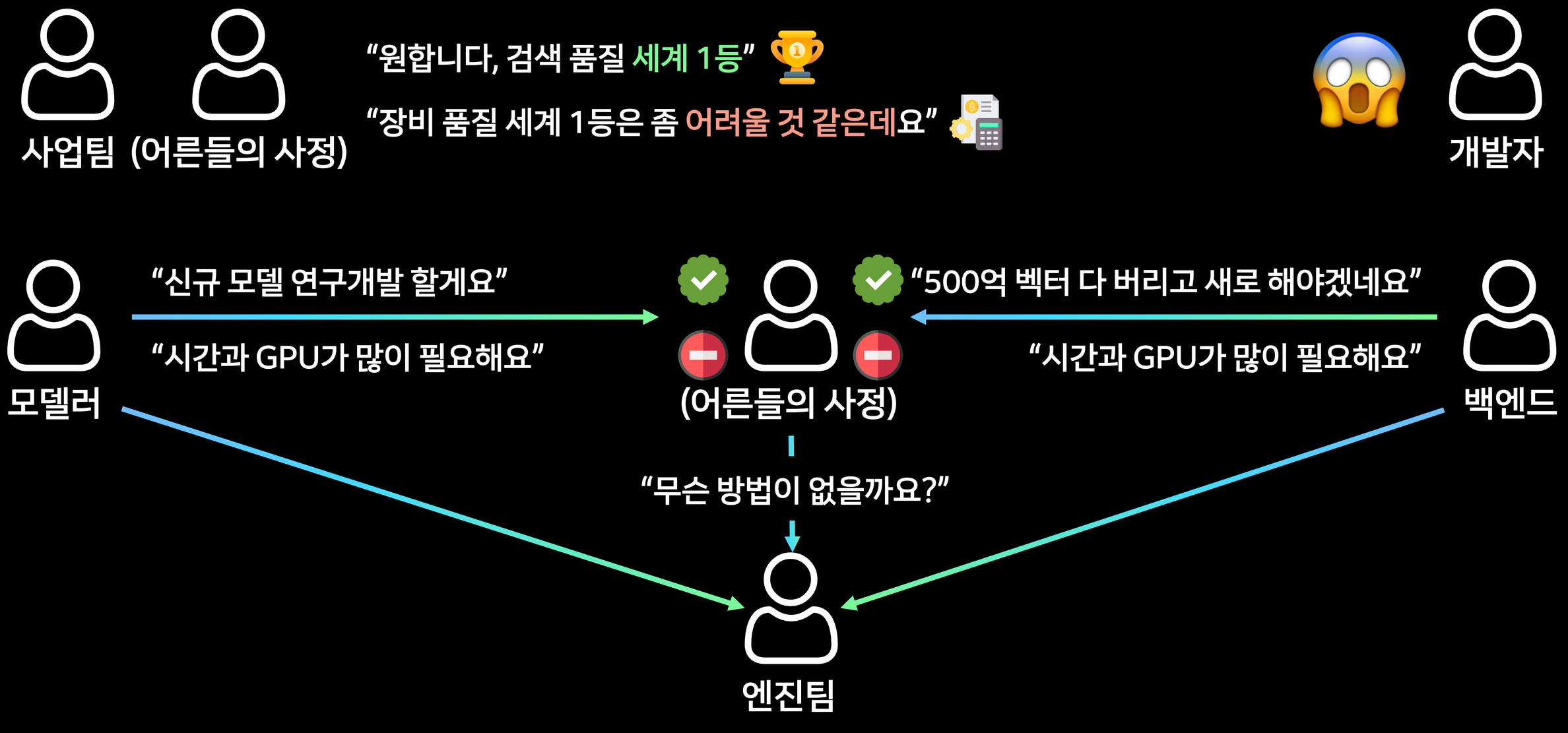


STEP 4. ranking: score를 기반으로 순위 매기기

2.3 The Complexity of Applying ColBERT



2.4 So What's the Problem?



2.5 그렇게 등장한 “무지출 챌린지”

엔진 개발자

“모델 변경으로 인한 overhead를 줄이면서”
“검색 품질은 상승시키고”
“같은 장비에서 더 많은 문서를 서빙할 수 있는 방법이 있을까?”

1st ranker로서의 엔진의 강점에 주목,
Retrieval과 Ranking 간의 관계를 보자

2.6 Solution Overview

학계 동향은 어떨까?

We present XTR, ContXextualized Token Retriever: a simplified and efficient method for multi-vector retrieval, through re-thinking the role of token retrieval. The key insight of XTR is that the token retrieval in multi-vector models should be trained to retrieve the most salient and informative document tokens, so that the score between a query and document can be computed using only the retrieved information, just like how single-vector retrieval models work. By doing so, the gathering

[Lee et al., NeurIPS 2023]

employs a simple combination of distillation from a cross-encoder and hard-negative mining (§3.2) to boost quality beyond any existing method, and then uses a *residual compression* mechanism (§3.3)

[Santhanam et al., NAACL 2022]

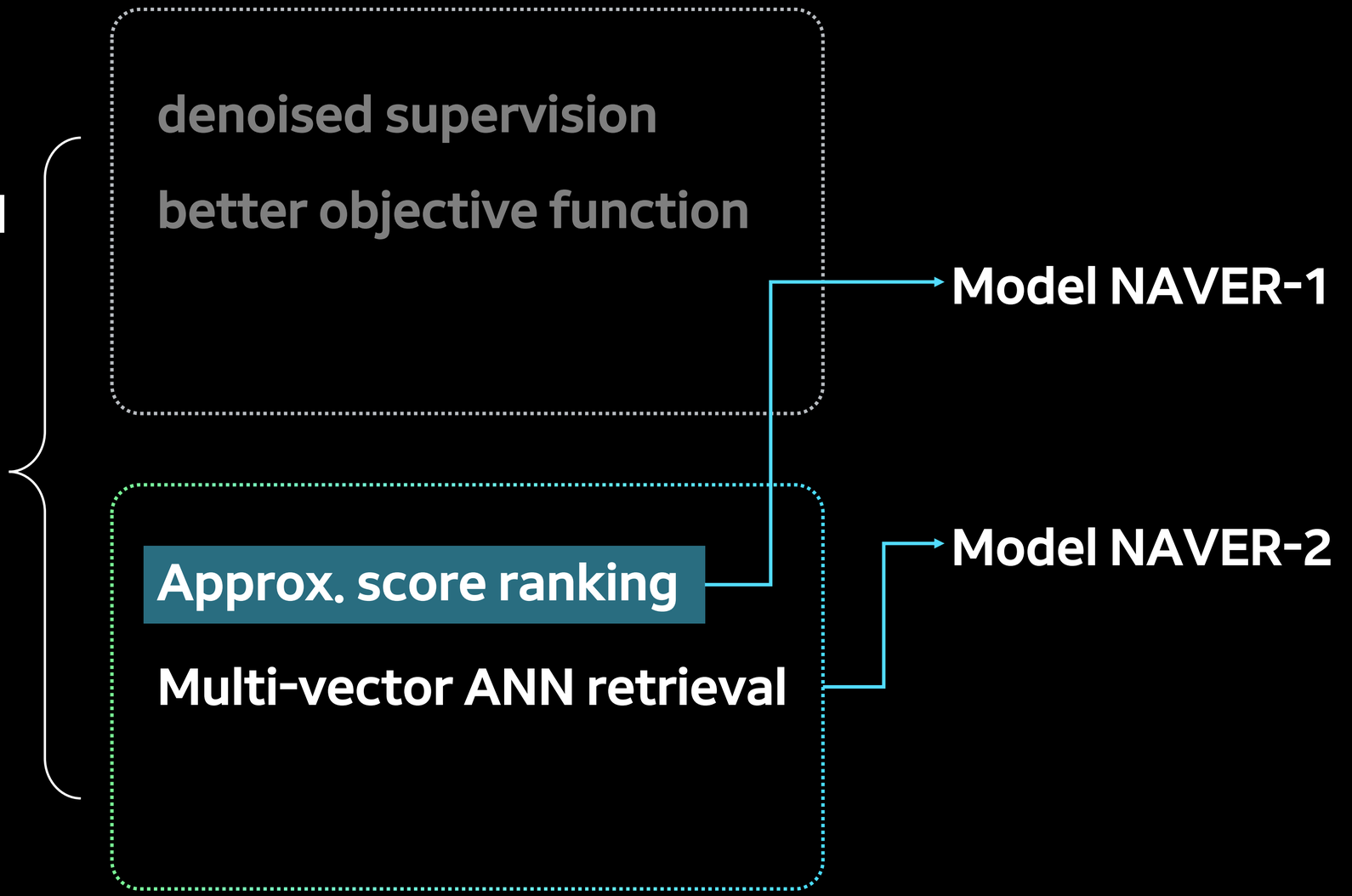
모델 재학습 포함

2.6 Solution Overview

Approximate Score
[Macdonald and Tonellotto, CIKM 2021]

ColBERTv2
[Santhanam et al., NAACL 2022]

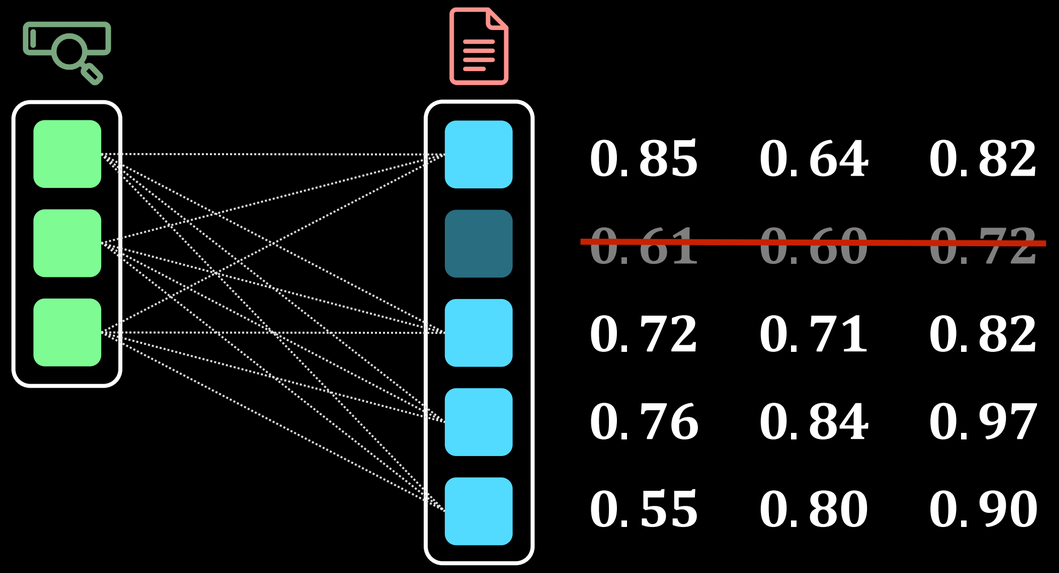
XTR
[Lee et al., NeurIPS 2023]



3. 첫번째 무지출 챌린지 솔루션, NAVER-1 모델

Retrieval을 위해 계산한 score로 ranking까지 진행해보는 건 어떨까?

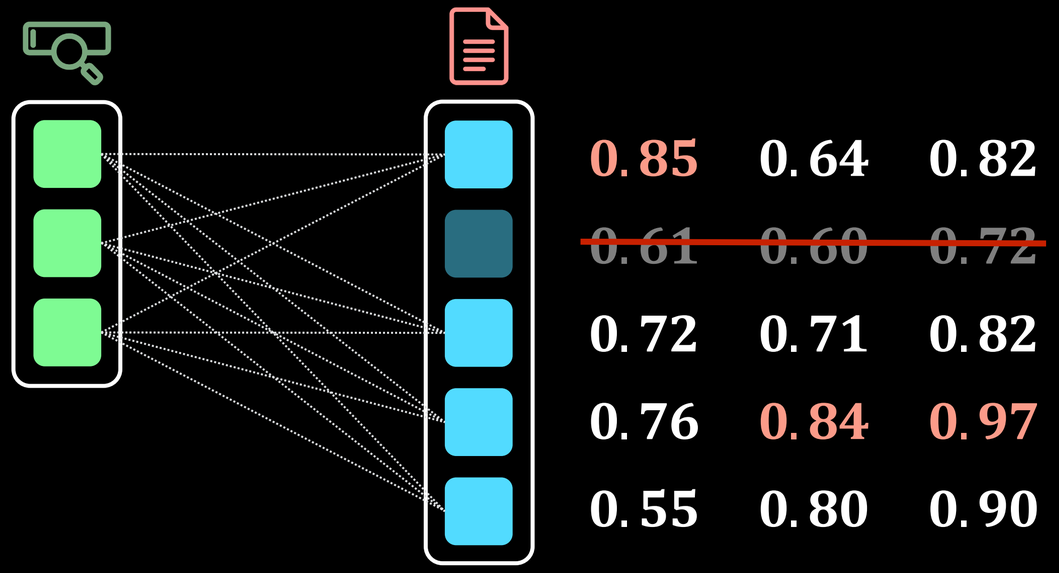
3.1.1 Motivation: ColBERTv2 approx. score ranking



Retrieval 된 vector만 보자

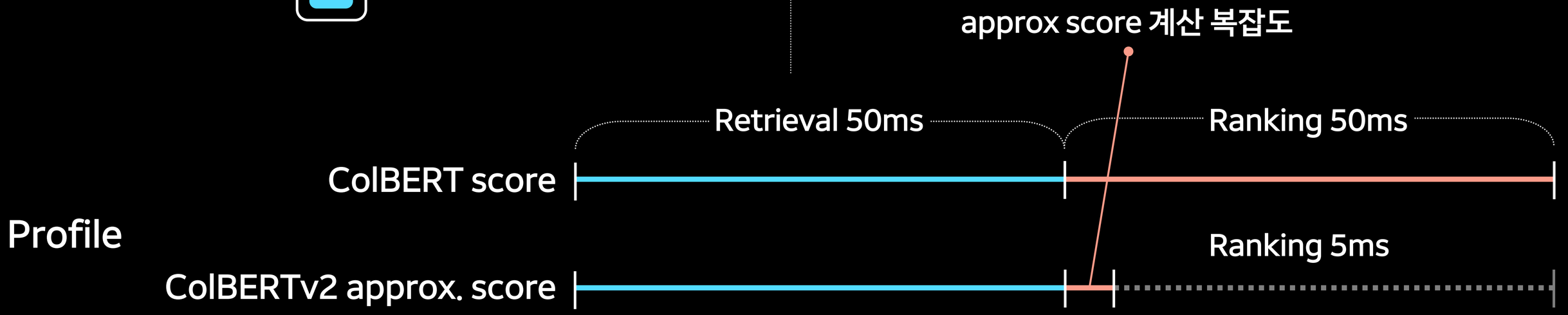
retrieval 되지 않은 vector들은 similarity가 낮아,
대부분의 경우 max 연산에서 탈락할 것이라는 직관

3.1.1 Motivation: ColBERTv2 approx. score ranking



0.85	0.64	0.82
~~0.61~~	~~0.60~~	~~0.72~~
0.72	0.71	0.82
0.76	0.84	0.97
0.55	0.80	0.90

$$approx\ score(q, doc) = 0.85 + 0.84 + 0.97 = 2.66$$

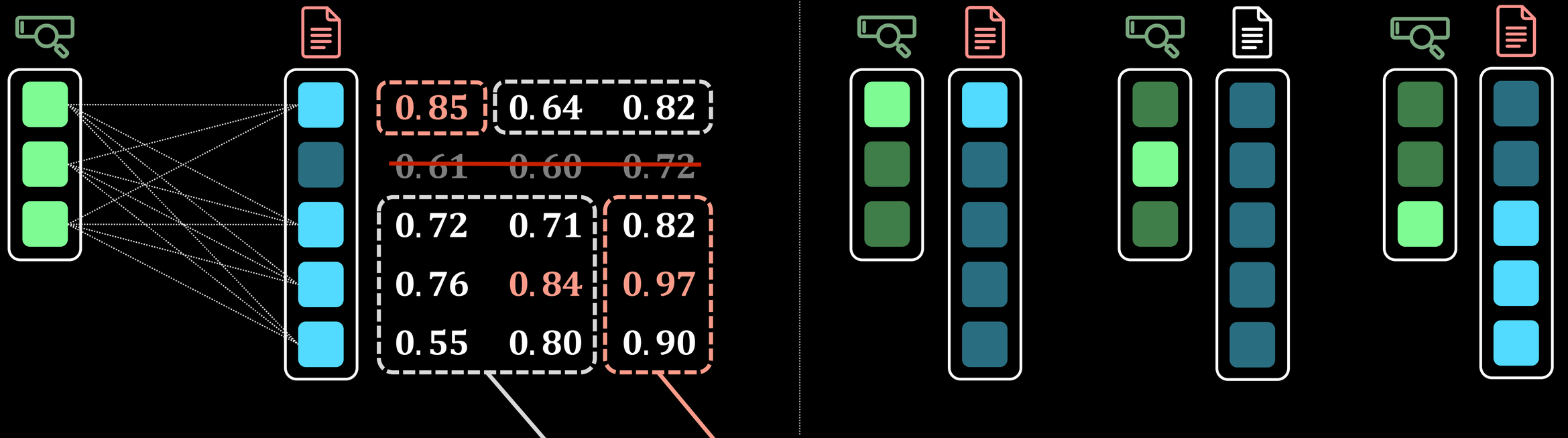


3.1.2 Experimental Results

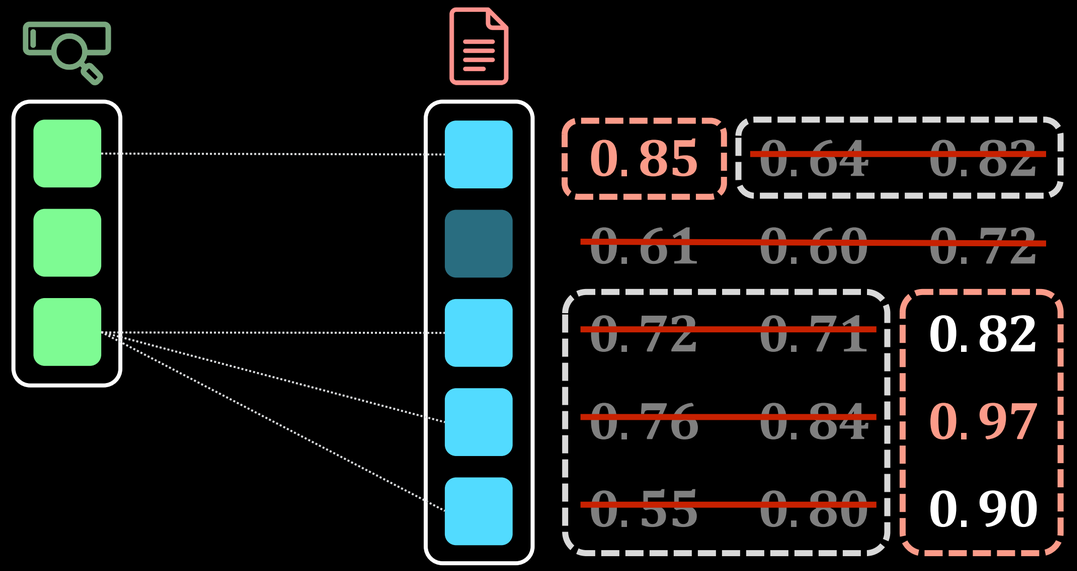
ColBERT score 대비 ColBERTv2 approx.score 품질

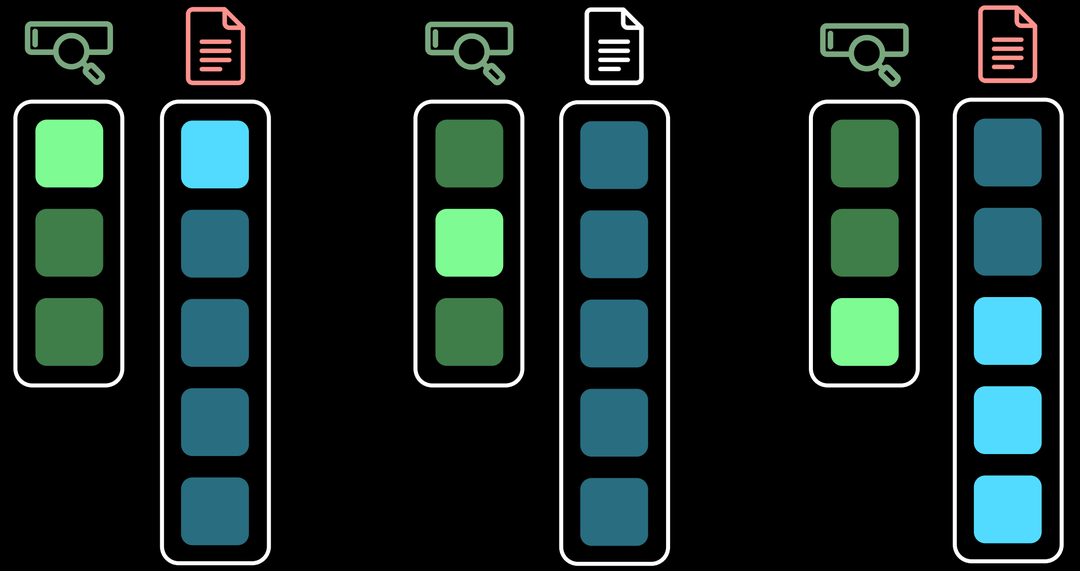
$(topK, ef)$	model	$ndcg@5$	$ndcg@10$	$mrr@10$	$recall@5$	$recall@10$	$precision@5$
$(320, 800)$	ColBERT	0.2385	0.2640	0.3760	0.2171	0.2901	0.1585
	ColBERTv2 approx. score	0.2264	0.2484	0.3617	0.2040	0.2689	0.1499
	유사도 (%)	94.92	94.09	96.19	93.96	92.69	94.57
$(640, 1600)$	ColBERT	0.2385	0.2640	0.3760	0.2171	0.2901	0.1585
	ColBERTv2 approx. score	0.2314	0.2544	0.3675	0.2095	0.2769	0.1535
	유사도 (%)	97.02	96.36	97.73	96.49	95.45	96.84

3.2.1 Further Improvements



3.2.1 Further Improvements 1: XTR basic



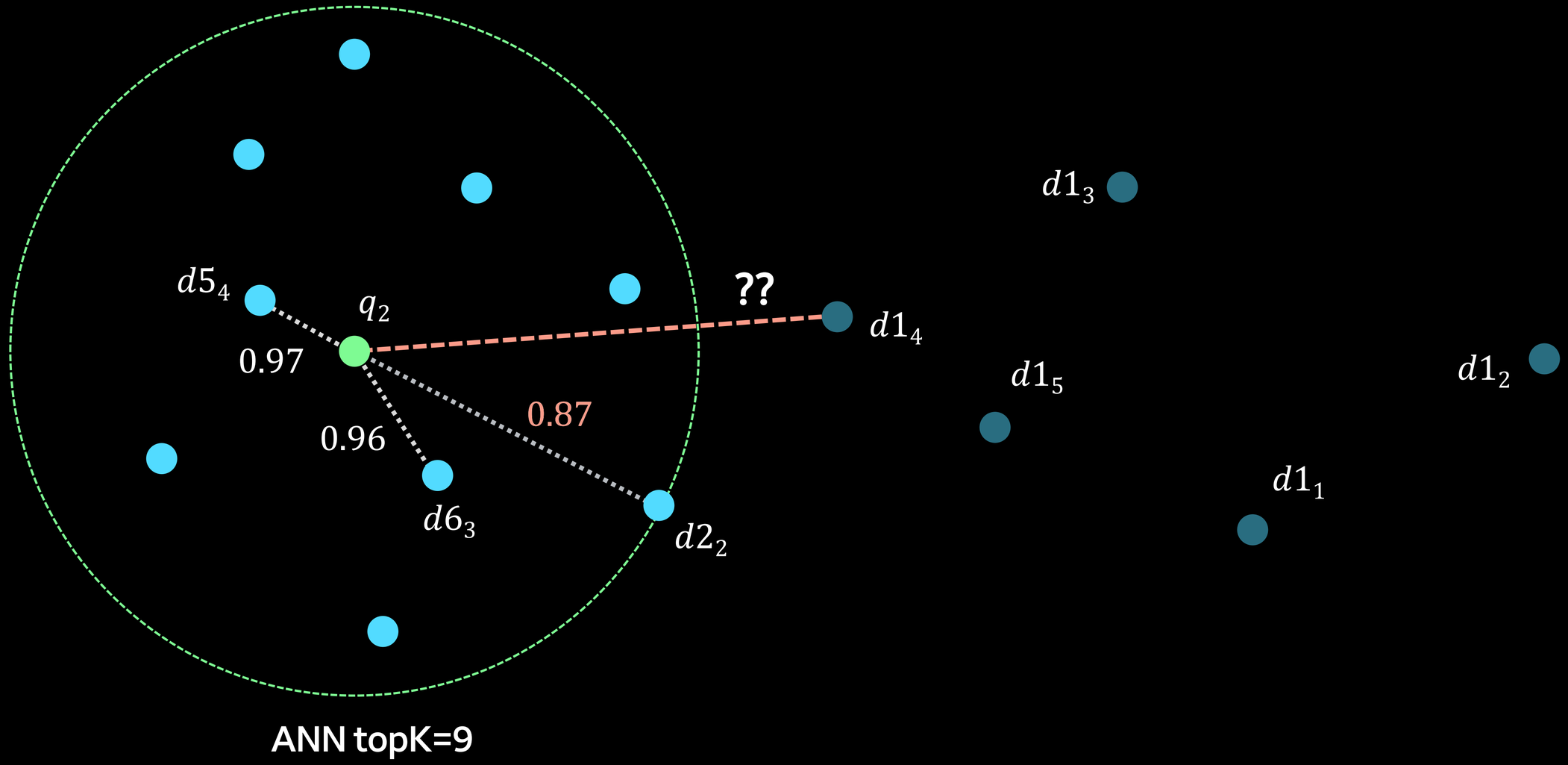


conteXtualized Token Retriever
[Lee et al., NeurIPS 2023]

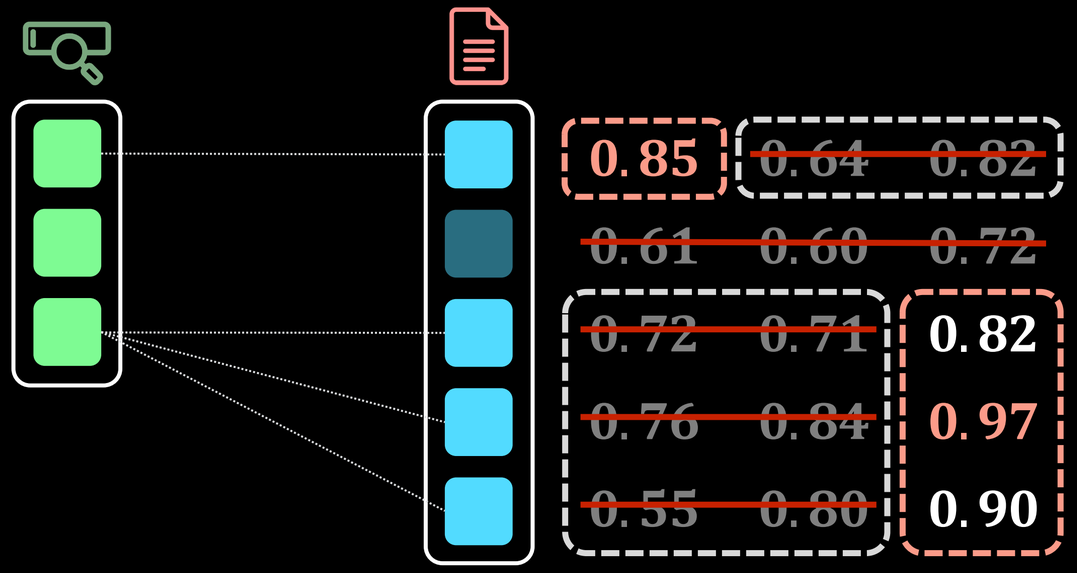
$$XTR\ basic\ score(q, doc) = 0.85 + null + 0.97 = NA$$

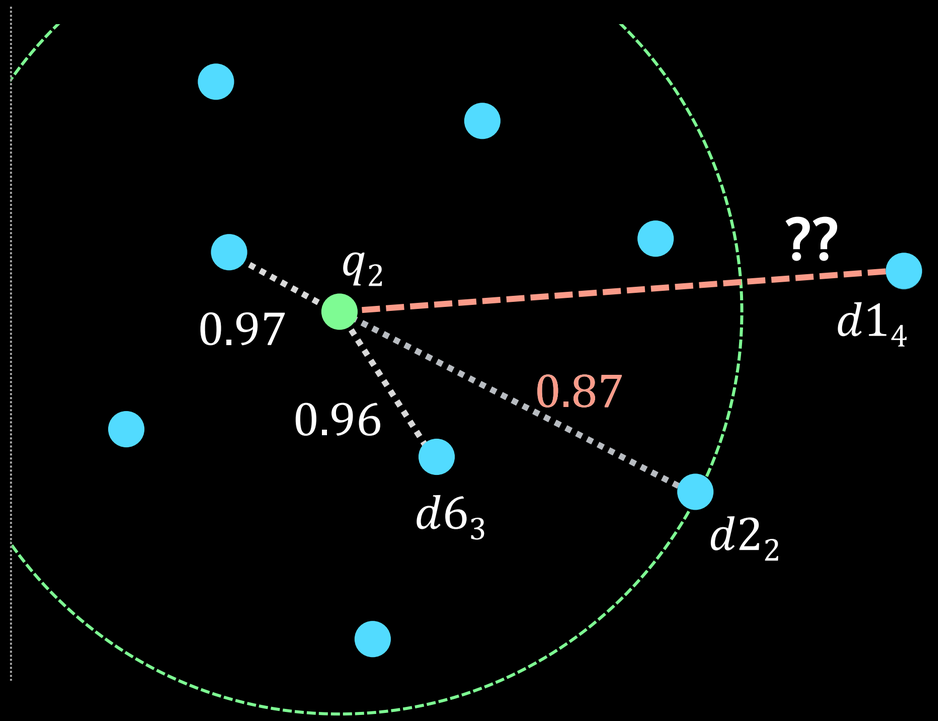
$$\Rightarrow 0.85 + \frac{(0.85 + 0.97)}{2} + 0.97 = 2.73$$

3.2.2 Further Improvements 2: MSI (Missing Similarity Imputation)



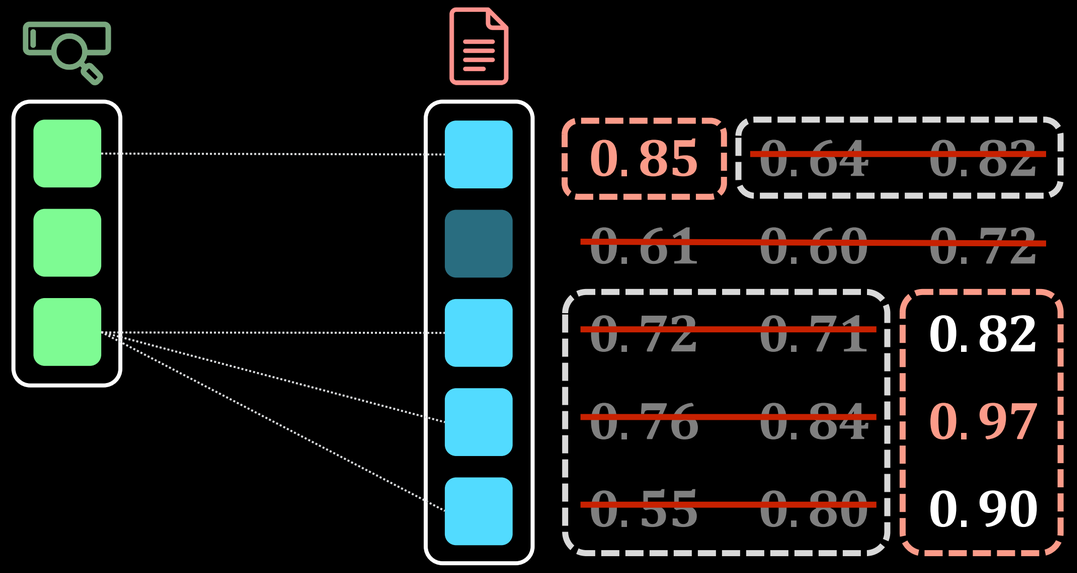
3.2.2 Further Improvements 2: MSI (Missing Similarity Imputation)

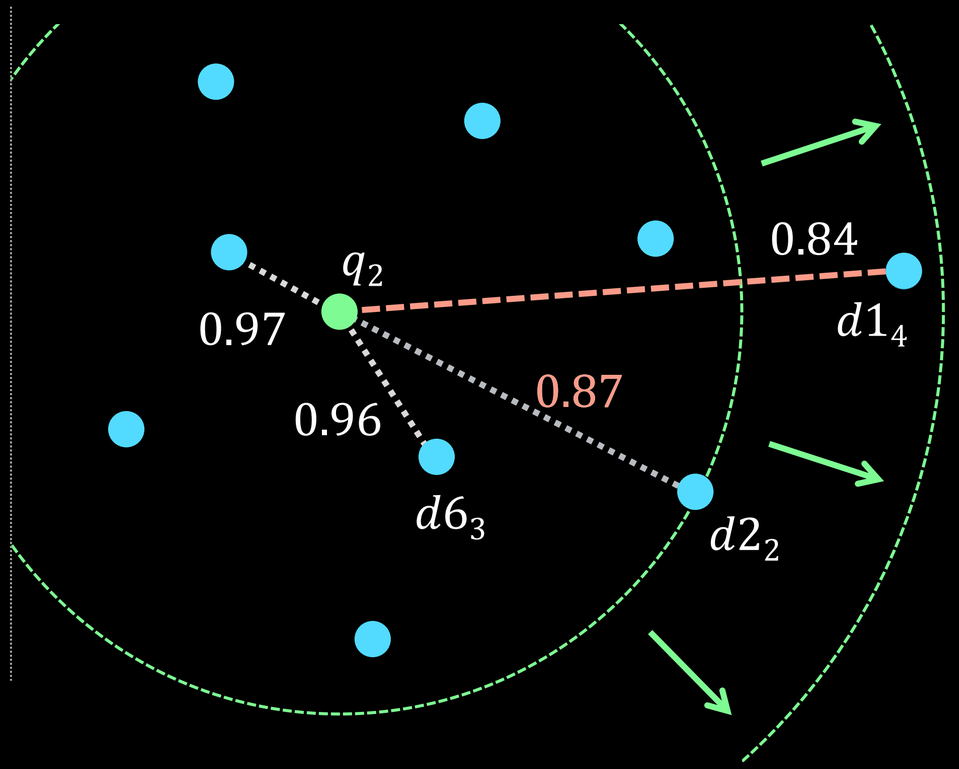




$$\textbf{XTR MSI score}(q, doc) = 0.85 + 0.87 + 0.97 = 2.69 \geq \textbf{ColBERT score}$$

3.2.3 Further Improvements 3: MSI++



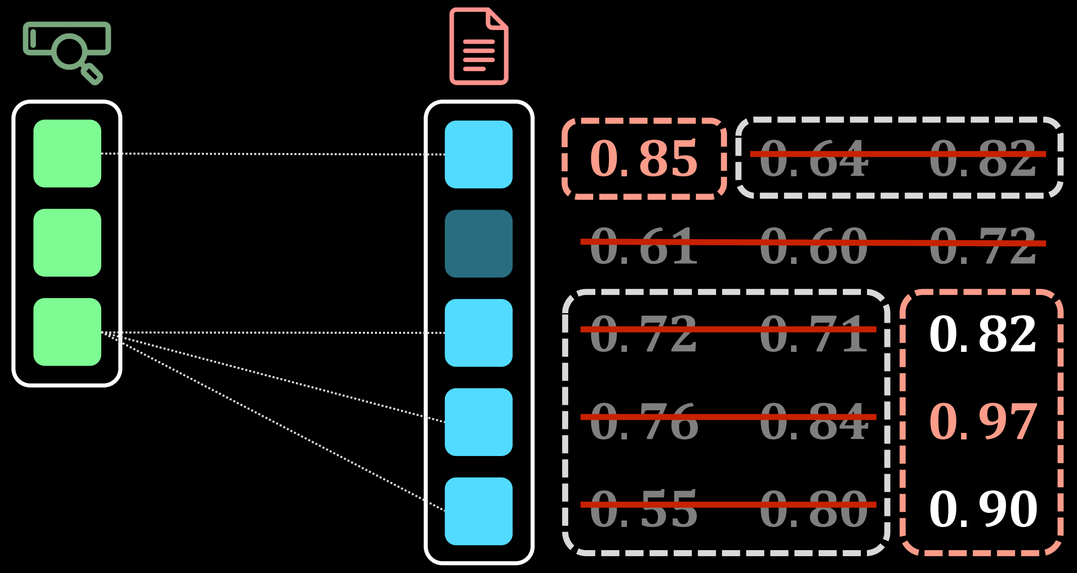


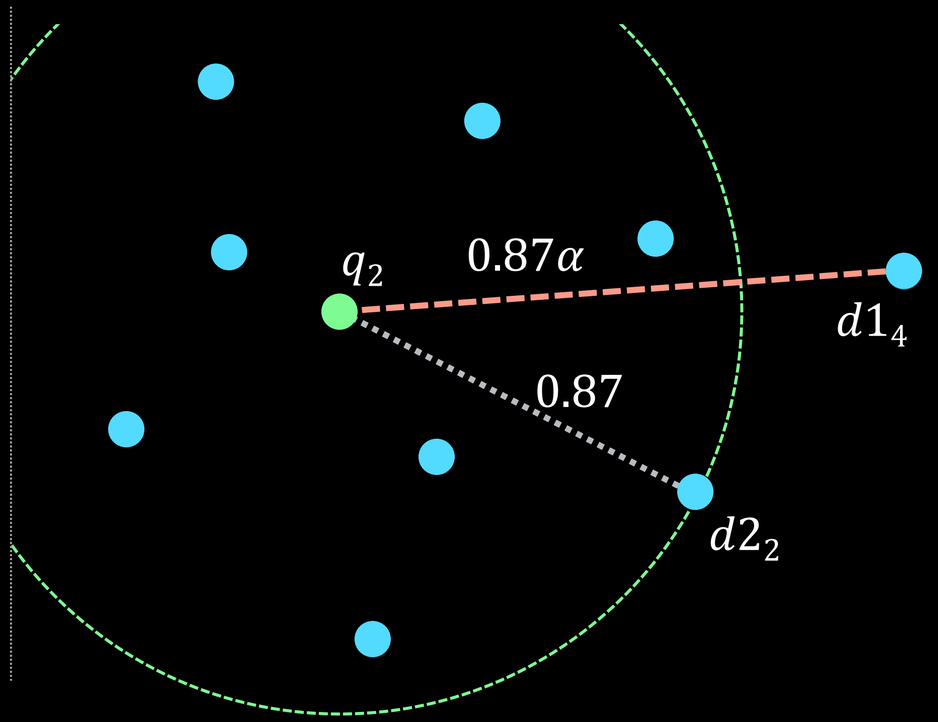
$$\textbf{XTR MSI score}(q, doc) = 0.85 + 0.87 + 0.97 = 2.69 \geq \textbf{ColBERT score}$$

$score_{XTR\_MSI} \rightarrow score_{ColBERT}, topK \rightarrow (\#vector)$

품질 튜닝 포인트 1. $topK \rightarrow ef$

3.2.3 Further Improvements 3: MSI++





$$XTR\ MSI\ advanced\ score(q, doc) = 0.85 + 0.87\alpha + 0.97 \approx ColBERT\ score$$

$0 \leq \alpha \leq 1$, tighter upper bound를 위한 penalty rate

품질 튜닝 포인트 2. $0 < \alpha < 1$ 인 적절한 값 선택

3.2.4 Experimental Results

Scoring 방식에 따른 최대 $RPS(request\ per\ sec)$ 및 응답시간

	$topK$	ef	최대 $\boldsymbol{RPS}$	평균 응답시간 (ms)
ColBERT	40	100	289.32	86.58
	-	-	-	-
ColBERTv2 approx. score	40	100	484.03	43.09
	-	-	-	-
XTR basic	40	100	490.10	41.33
	-	-	-	-
XTR MSI	40	100	489.51	41.45
	-	-	-	-

3.2.4 Experimental Results

Scoring 방식에 따른 최대 $RPS(request\ per\ sec)$ 및 응답시간

	$topK$	ef	최대 $\boldsymbol{RPS}$	평균 응답시간 (ms)
ColBERT	40	100	289.32	86.58
	100	100	185.57	125.00
ColBERTv2 approx. score	40	100	484.03	43.09
	100	100	458.38	44.21
XTR basic	40	100	490.10	41.33
	-	-	-	-
XTR MSI	40	100	489.51	41.45
	-	-	-	-

3.2.4 Experimental Results

Scoring 방식에 따른 최대 $RPS(request\ per\ sec)$ 및 응답시간

	$topK$	ef	최대 $\boldsymbol{RPS}$	평균 응답시간 (ms)
ColBERT	40	100	289.32	86.58
	100	100	185.57	125.00
ColBERTv2 approx. score	40	100	484.03	43.09
	100	100	458.38	44.21
XTR basic	40	100	490.10	41.33
	100	100	473.42	43.97
XTR MSI	40	100	489.51	41.45
	100	100	473.09	42.82

XTR MSI++

3.2.4 Experimental Results

	$ndcg@5$	$ndcg@10$	$mrr@10$	$recall@5$	$recall@\mathbf{10}$	$precision@5$
BM25	0.1244	0.1629	0.1484	0.0970	0.1663	0.0649
ColBERT	0.2097	0.2331	0.3357	0.1919	0.2581	0.1392
XTR MSI++	0.1974	0.2178	0.3209	0.1789	0.2378	0.1304

※ $(topK, ef) = (100, 100)$ ※ $\alpha = 0.7$

3.3 Advantages

성능

ranking 비용 제거, 응답속도 50% 향상

품질

성능 저하 없는 approx. score 정확도 향상 방법 2가지

1. Retrieval candidate 증가: $topK \ll ef \rightarrow topK = ef$
2. XTR ranking score 튜닝: MSI 도입, $\alpha = 0.7$

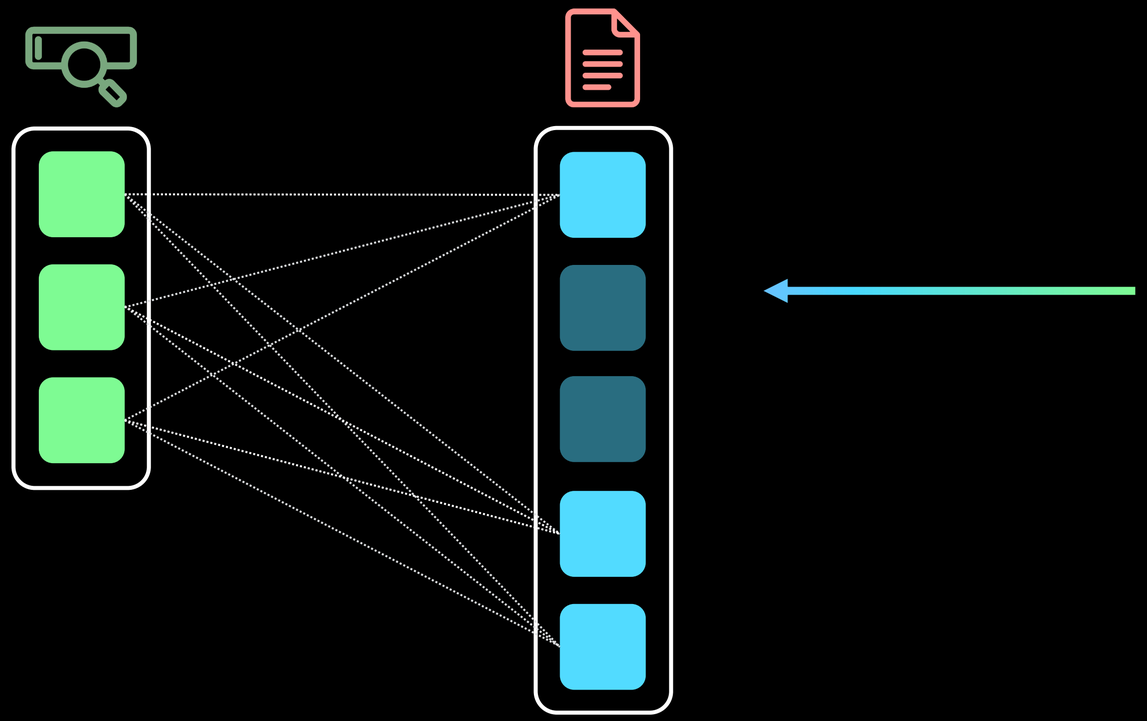
ColBERT 대비 95% 수준의 품질 유지

4. 두번째 무지출 챌린지 솔루션, NAVER-2 모델

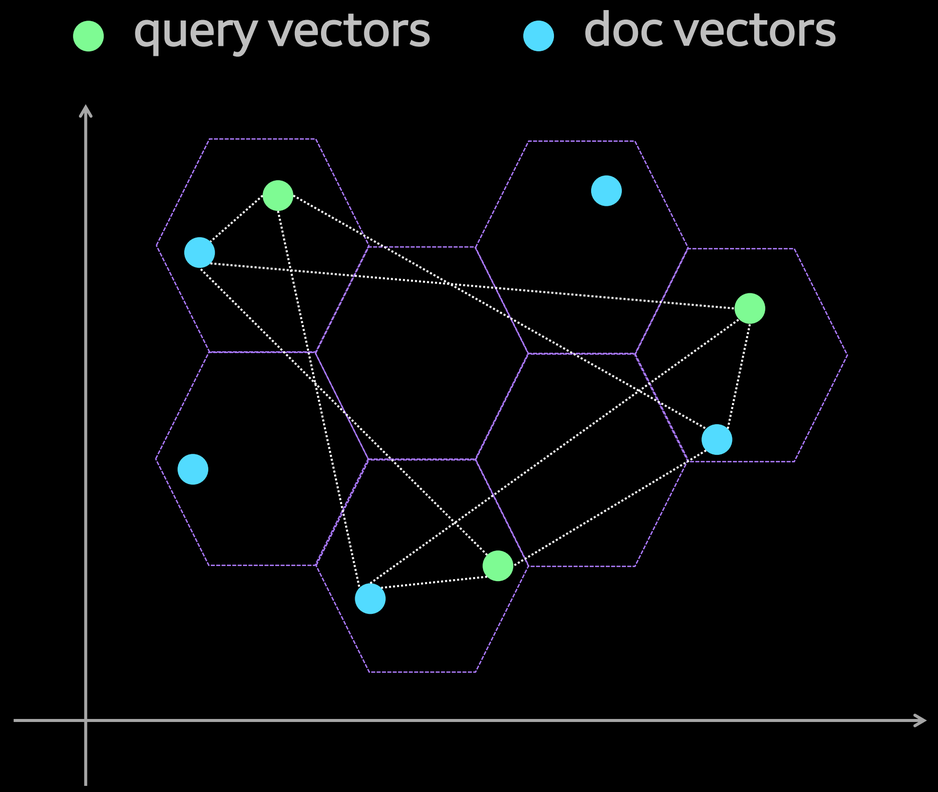
Multi-vector 검색에 딱 맞는 retrieval 방식은 무엇일까?

4.1.1 Motivation: ColBERTv2

approx. score



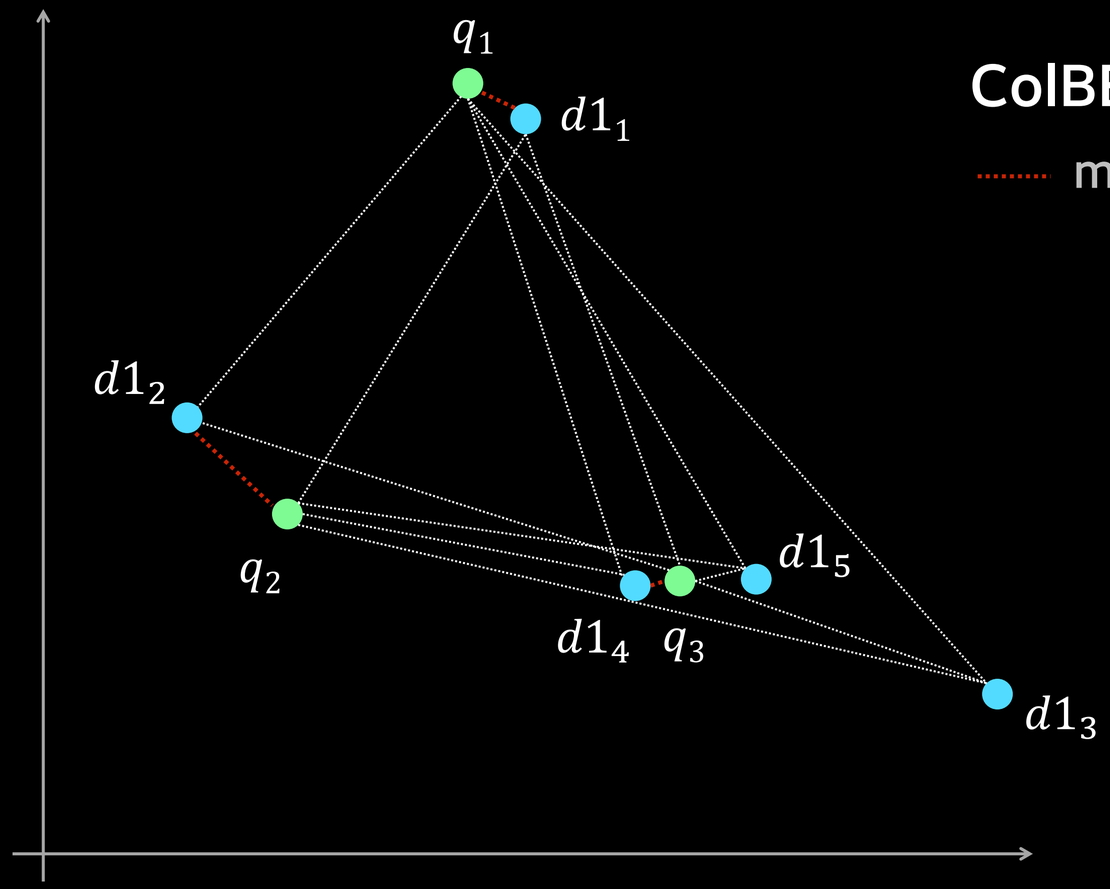
multi-vector 맞춤형 retrieval



ColBERTv2 [Santhanam et al., NAACL 2022]

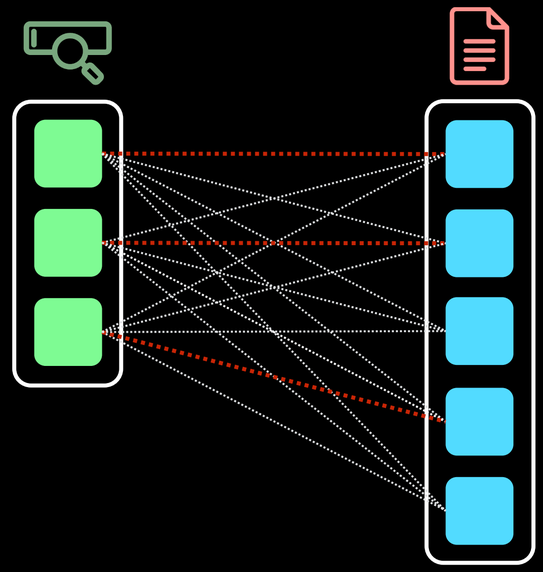
4.1.2 Revisiting ColBERTv2 approx. score

query vectors doc1 vectors



ColBERT score

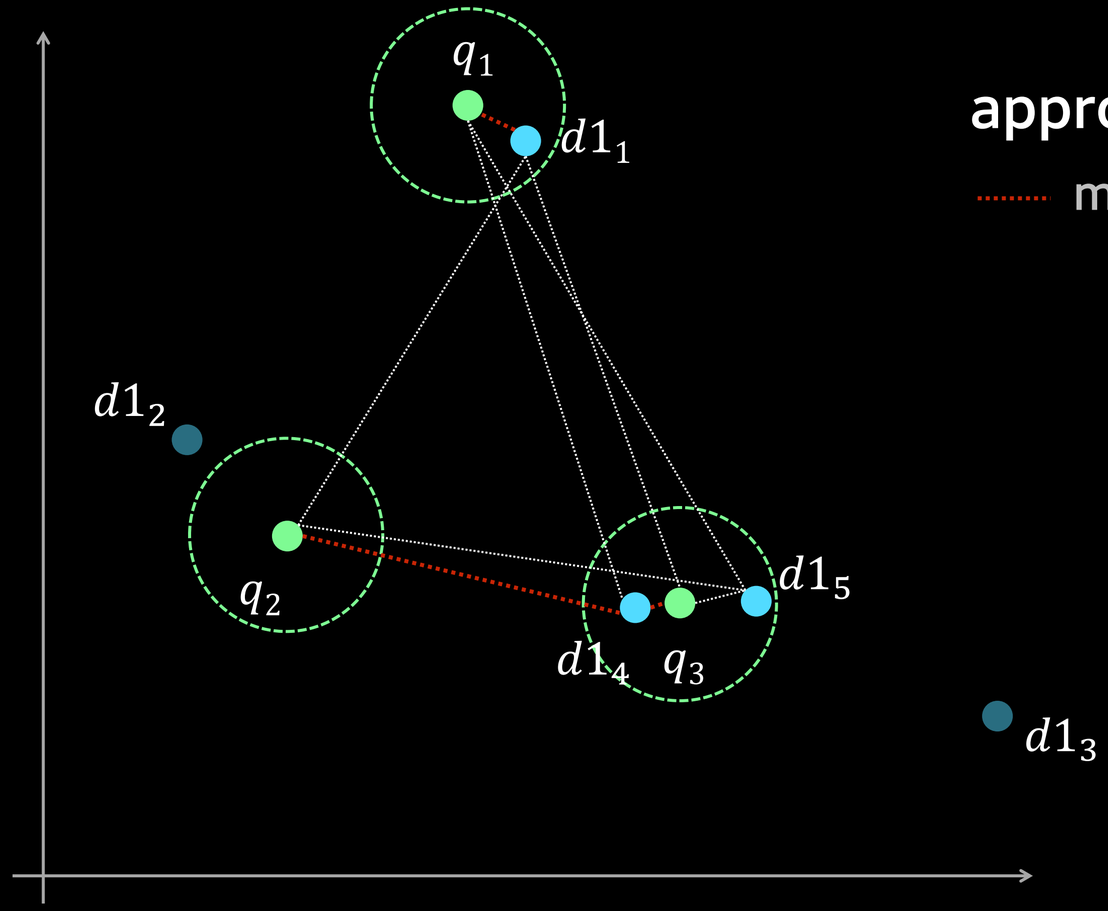
maximum similarity



※ 문서당 최대 100개의 vector, e.g., $d_1, d_2, \ldots, d_{100}$

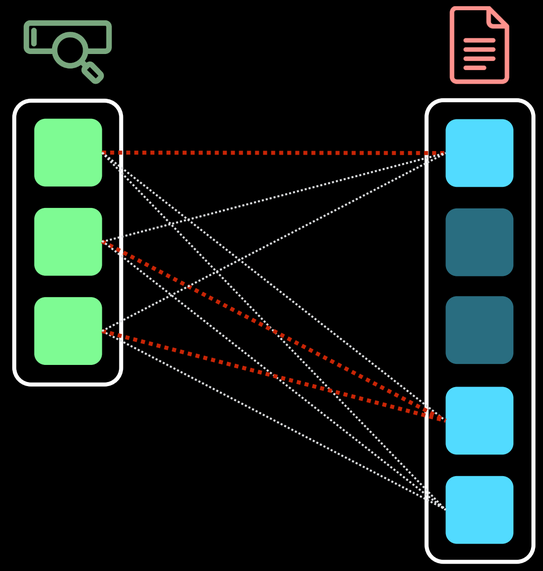
4.1.2 Revisiting ColBERTv2 approx. score

query vectors doc1 vectors single vector ANN - topK

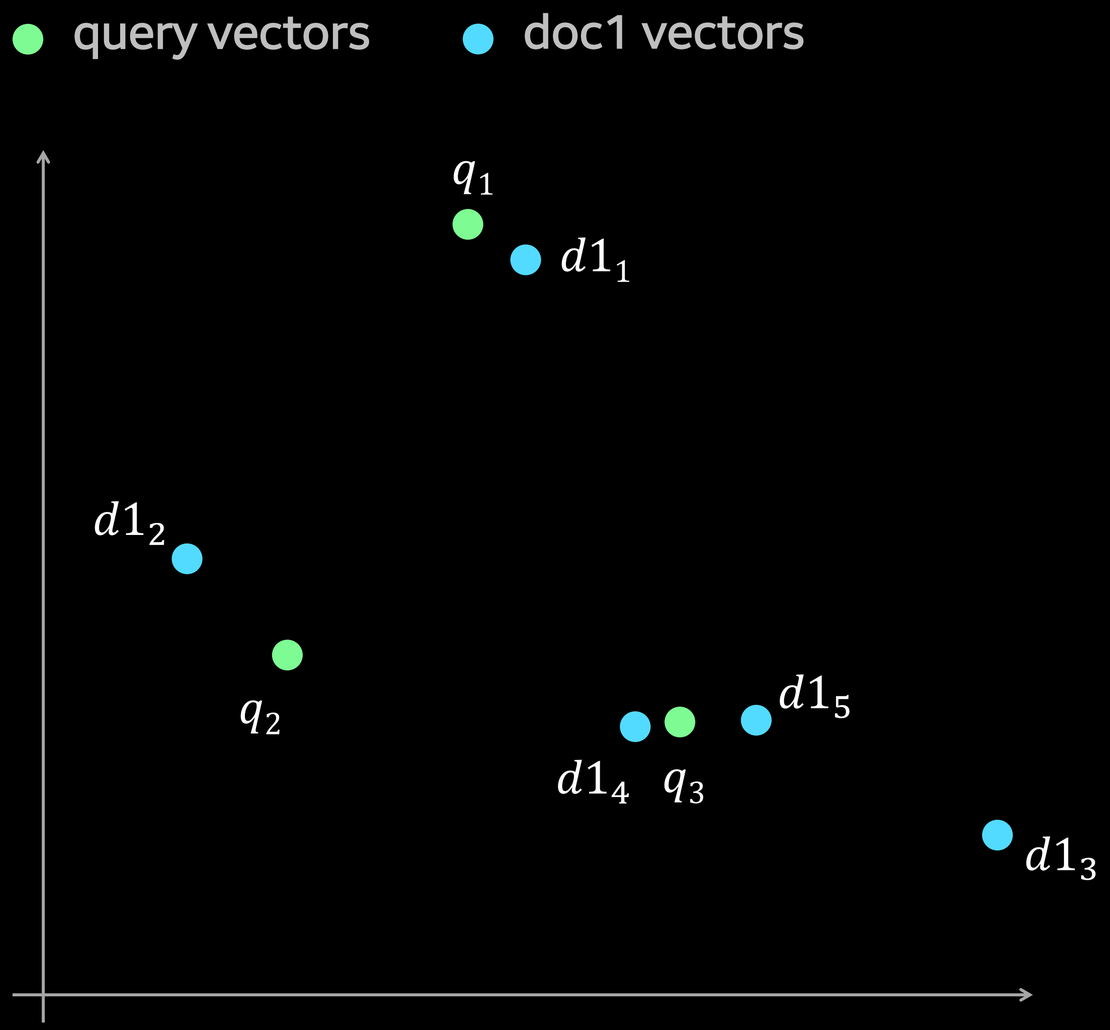


approx. score

maximum similarity



4.1.3 Multi-vector 맞춤형 retrieval이란?



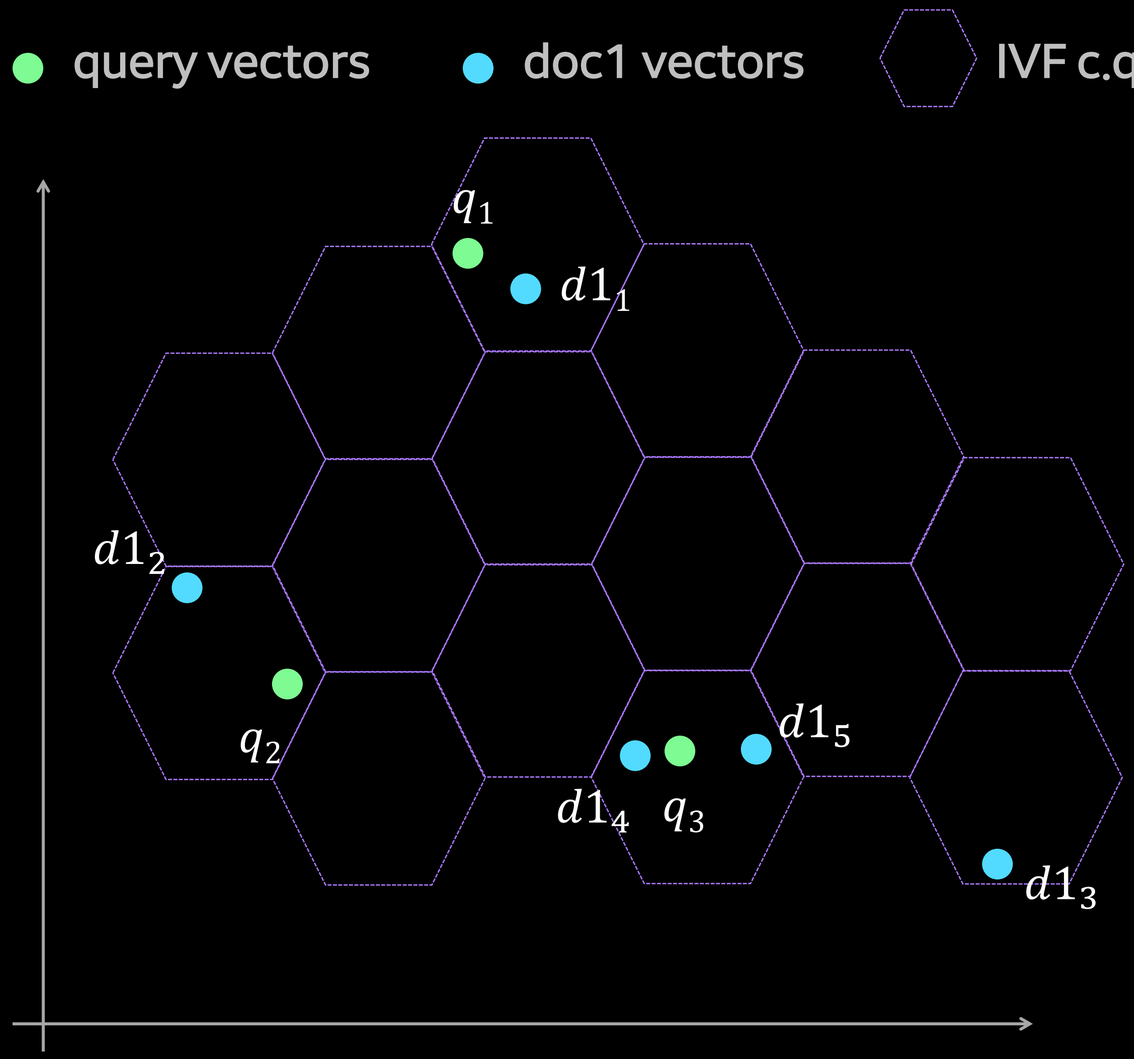
Retrieval

관련 없는 문서들을 **빠르게** 솎아내
ranking 후보군을 결정하는 과정

Ranking

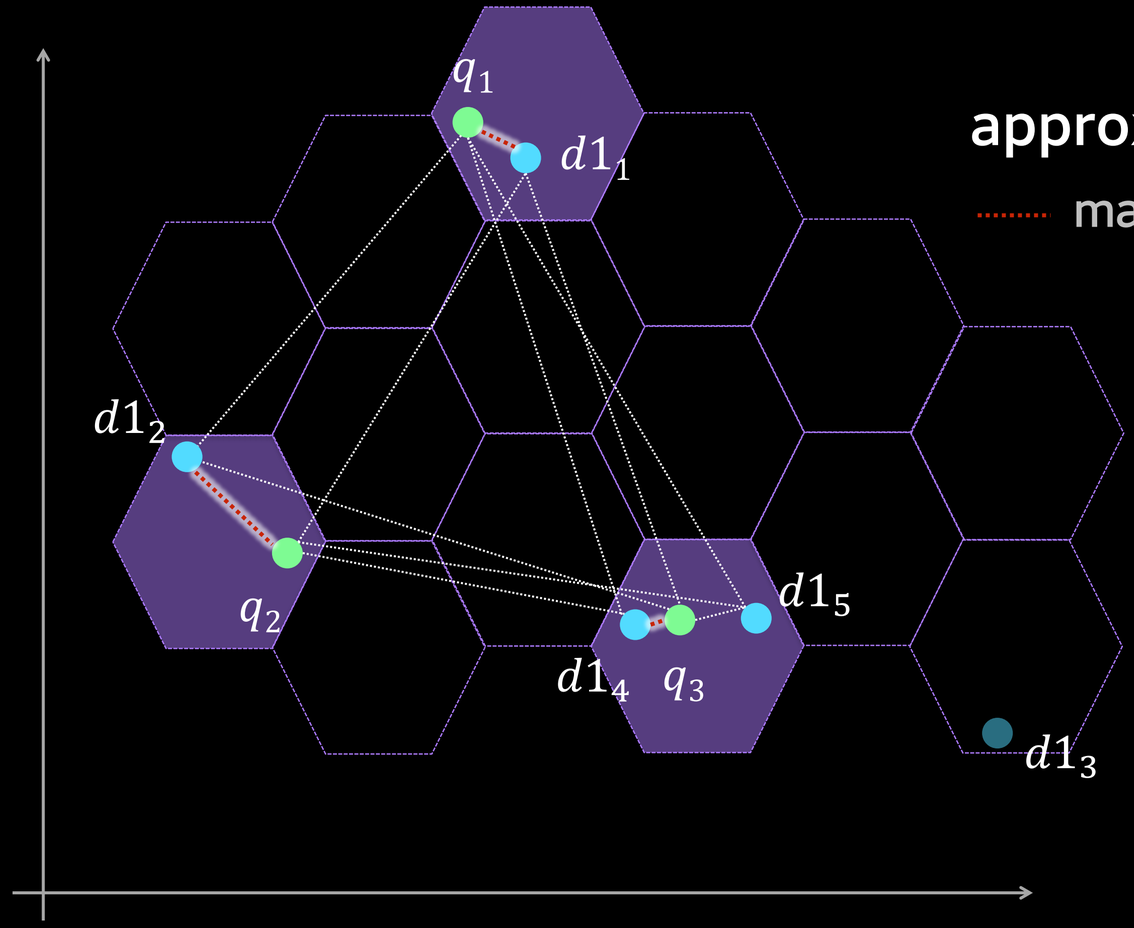
후보 문서들을 **정교하게** 순위화하는 과정

4.1.3 Multi-vector 맞춤형 retrieval이란?



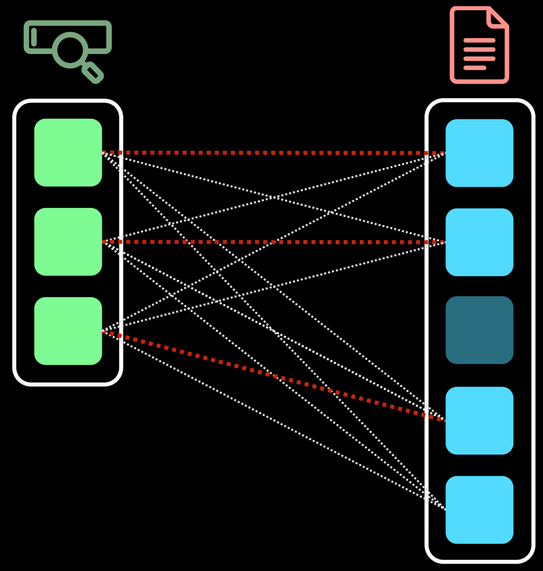
4.1.3 Multi-vector 맞춤형 retrieval이란?

query vectors doc1 vectors IVF c.q.



approx. score topM

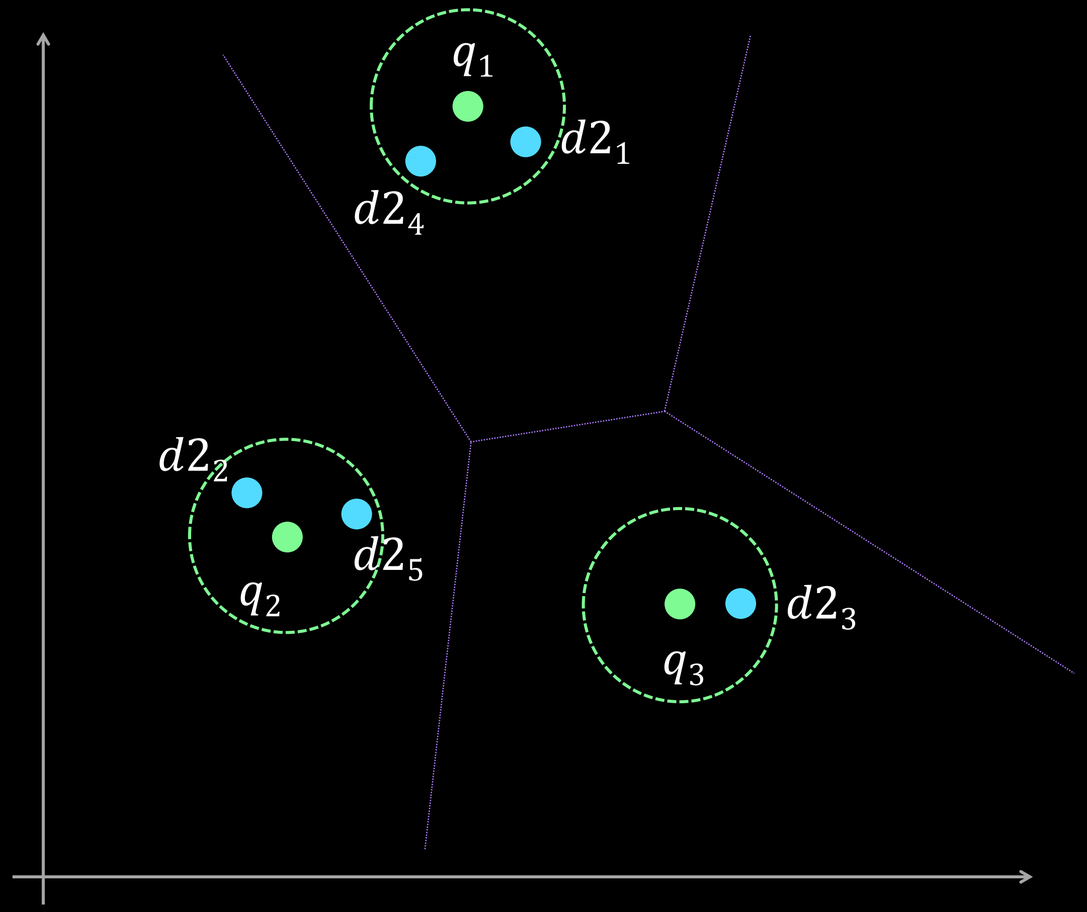
maximum similarity



multi-vector 맞춤형 retrieval
= ranking과 얼라인 되는 retrieval

4.1.4 “맞춤형 retrieval”이 왜 맞춤형일까?

query vectors doc2 vectors IVF c.q. single vector ANN - topK

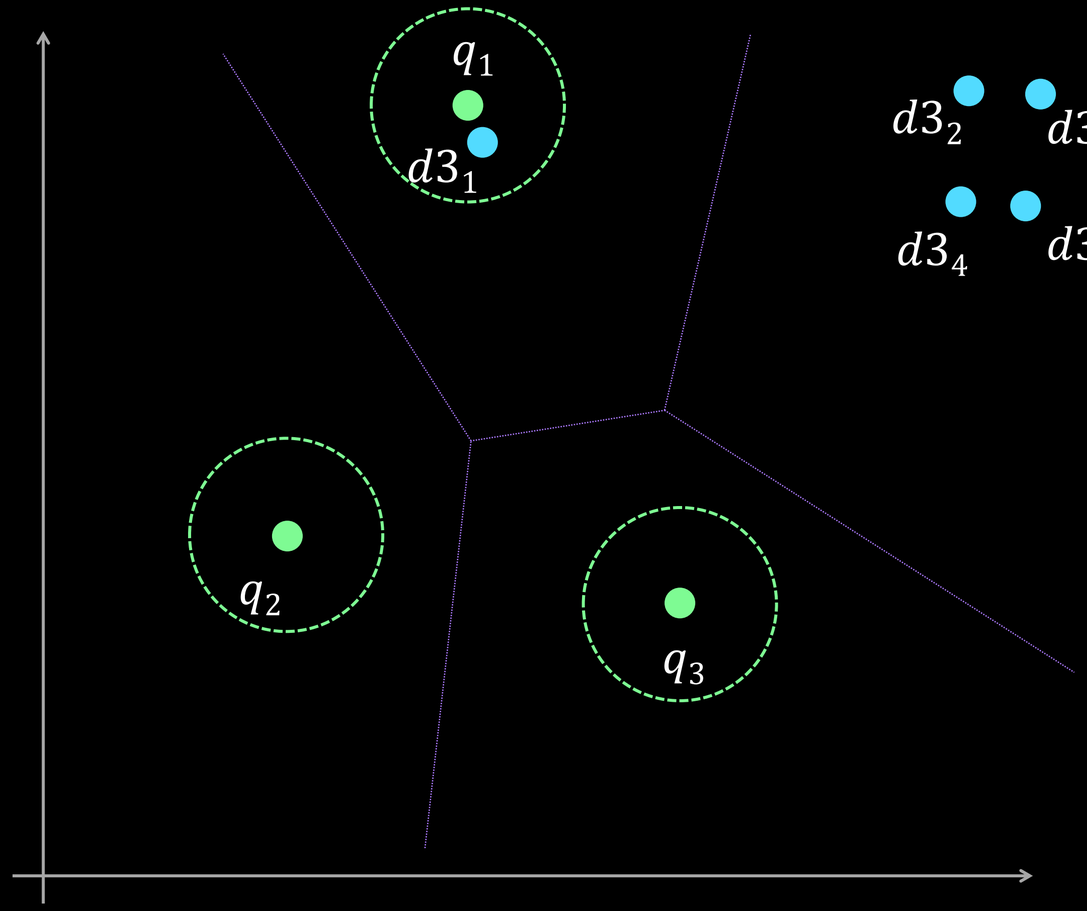


매우 적합한 doc2 retrieval 여부

맞춤형	✓
ColBERT	✓

4.1.4 “맞춤형 retrieval”이 왜 맞춤형일까?

query vectors doc3 vectors IVF c.q. single vector ANN - topK



$d3_2$ $d3_3$

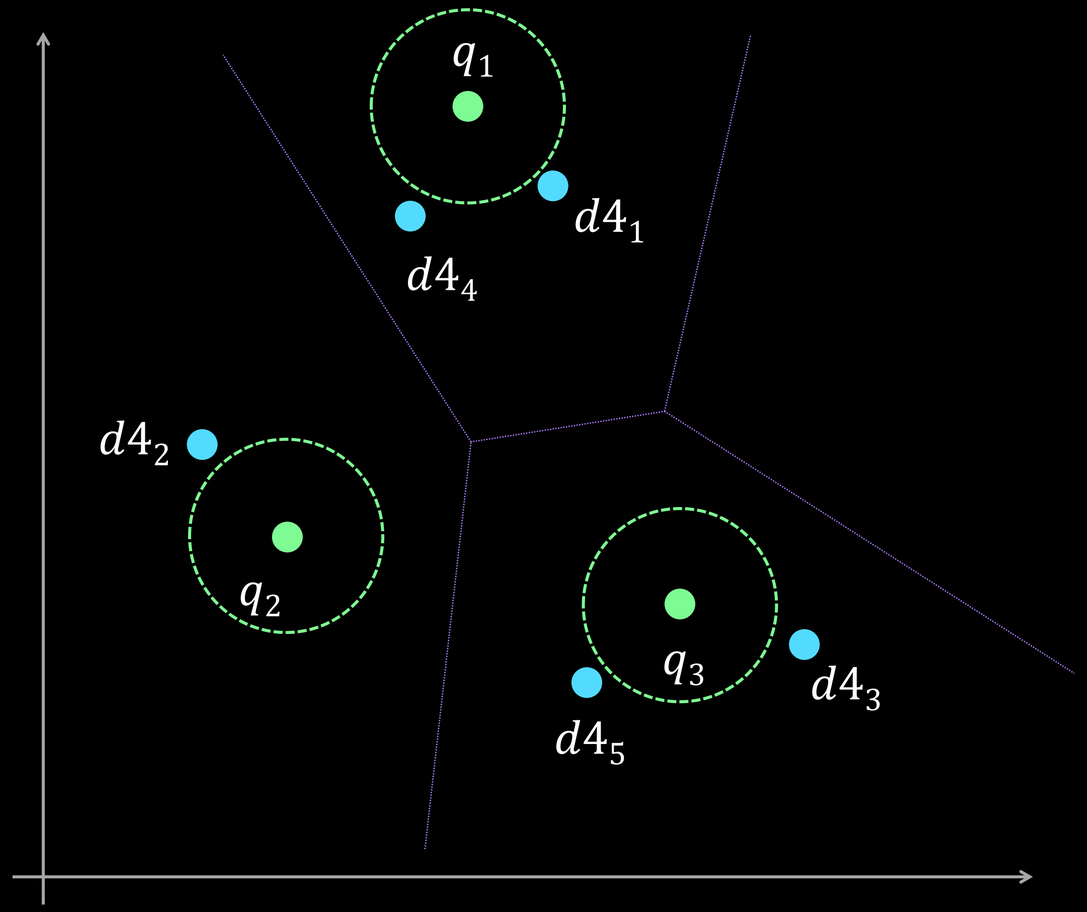
$d3_4$ $d3_5$

안 적합한 doc3 retrieval 여부

맞춤형	✗ (approx. score cut)
ColBERT	✓

4.1.4 “맞춤형 retrieval”이 왜 맞춤형일까?

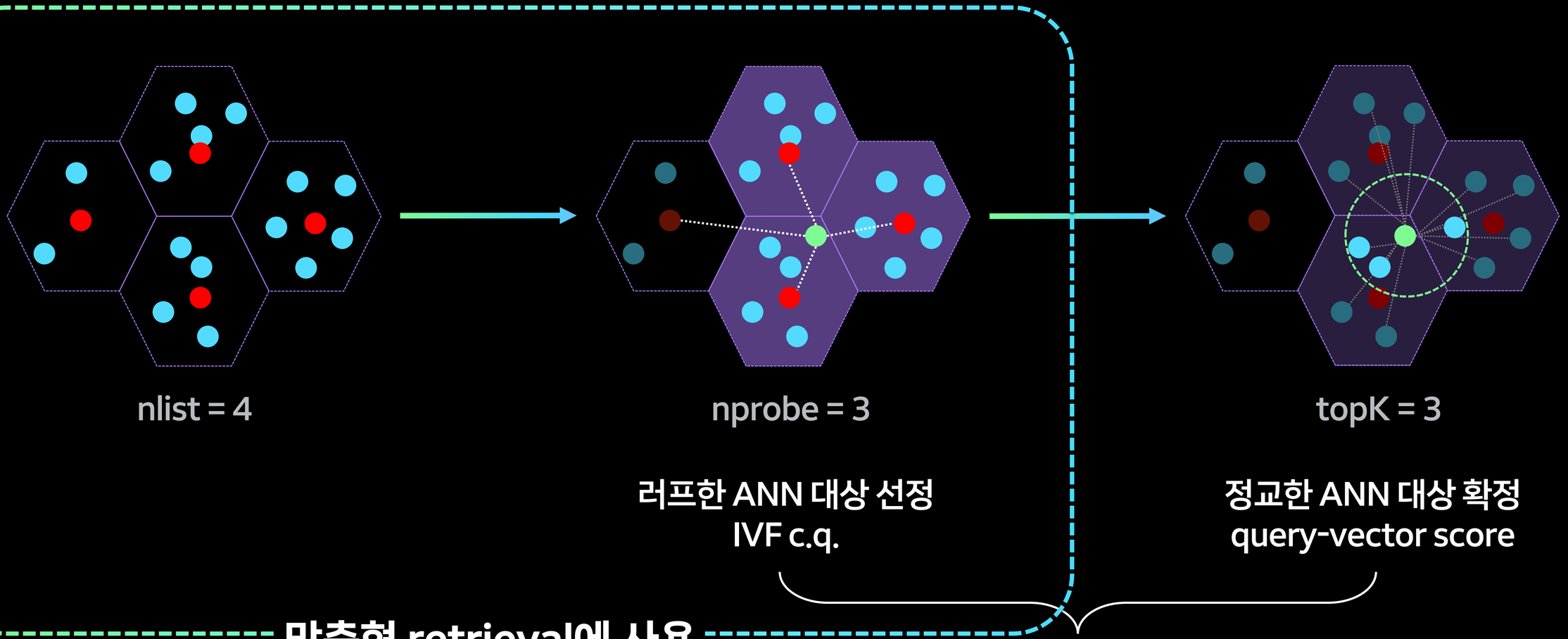
query vectors doc4 vectors IVF c.q. single vector ANN - topK



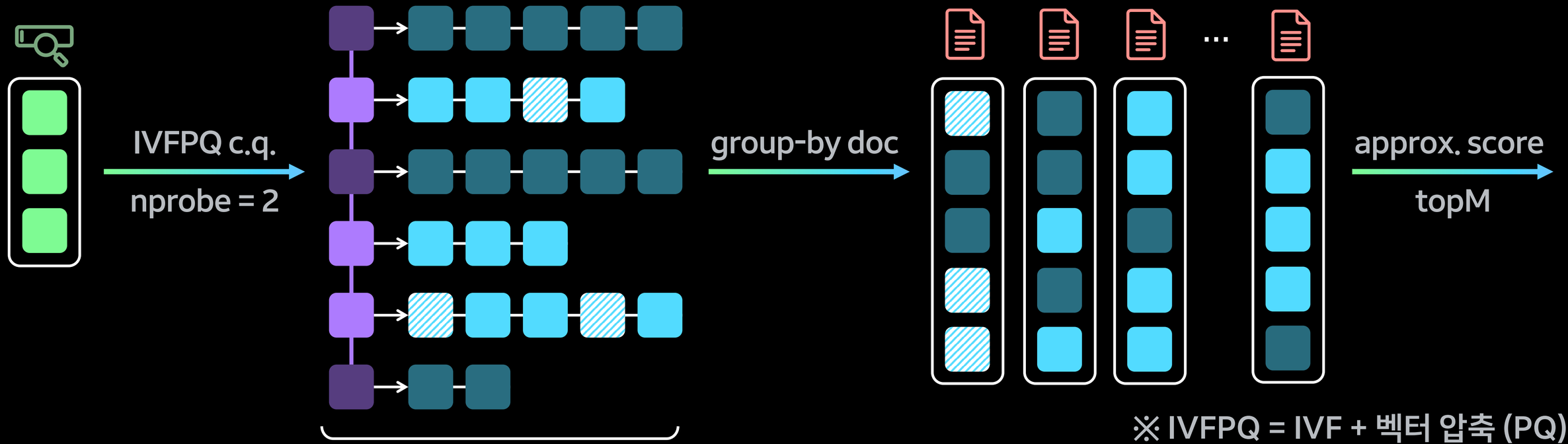
적합한 doc4 retrieval 여부

맞춤형	✓ 모든 query에 전부 가까워야 뽑힘
ColBERT	✗ 하나의 query라도 가까우면 뽑힘

4.1.5 IVF – 맞춤형 retrieval에 적합한 ANN



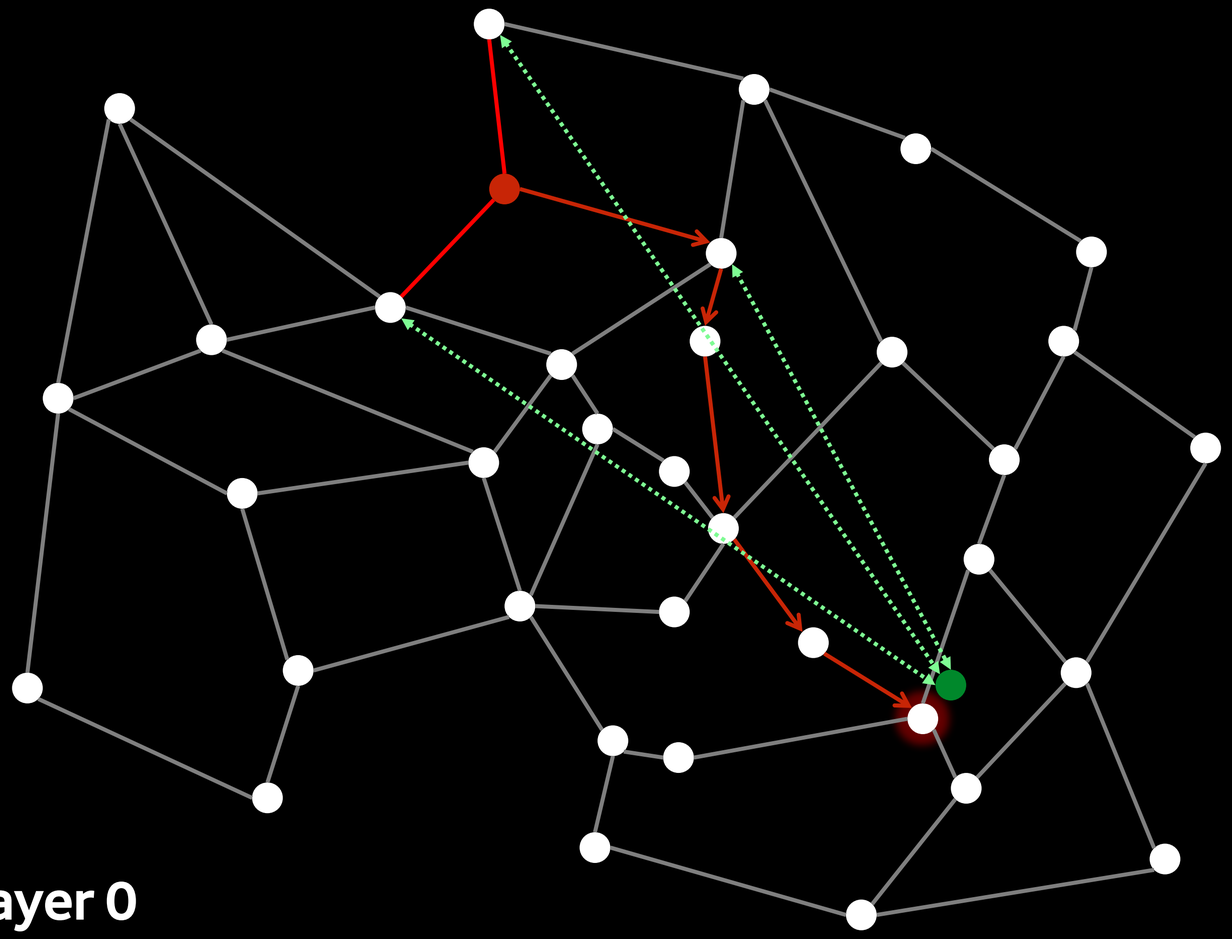
4.1.6 ColBERTv2 retrieval in detail



multi-vector 맞춤형 retrieval = IVFPQ c.q. + approx. score topM

4.1.7 왜 HNSW는 안 될까?

query vector DB vector starting point



layer 0

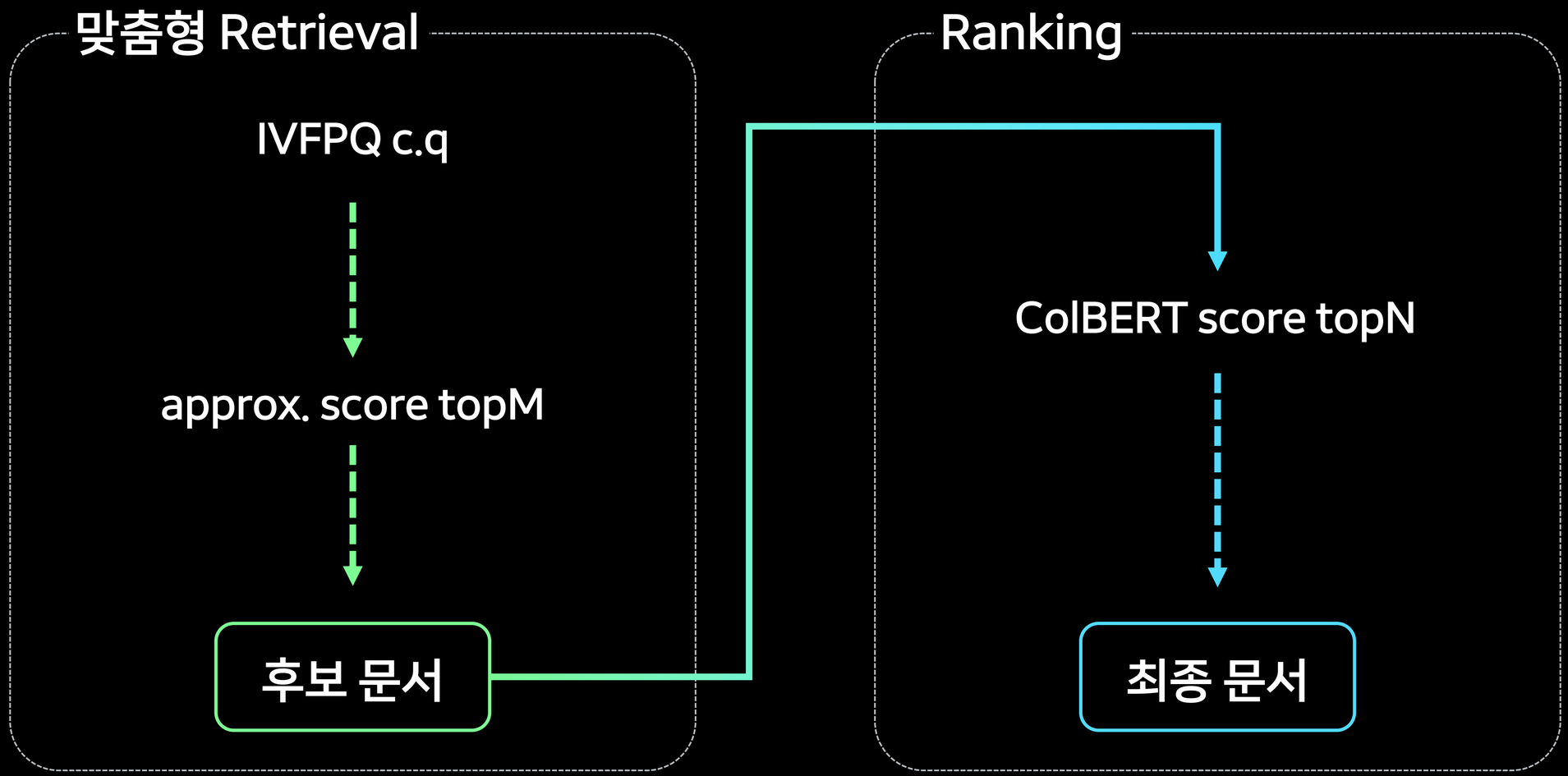
러프한 ANN 대상 선정

정교한 ANN 대상 확정

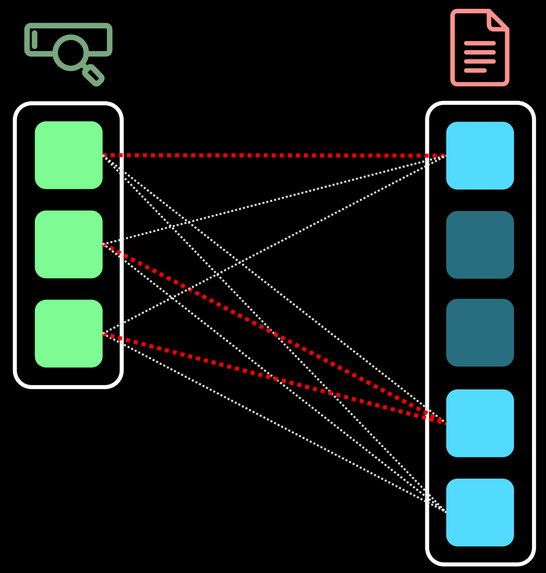
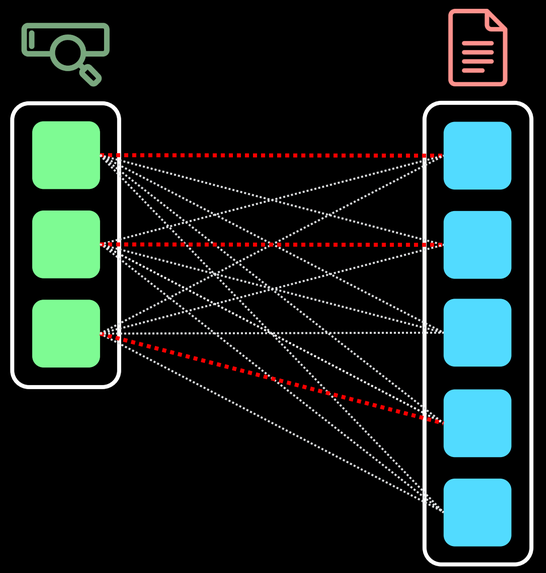
분리 불가

맞춤형 retrieval을 위해 IVFPQ 전환

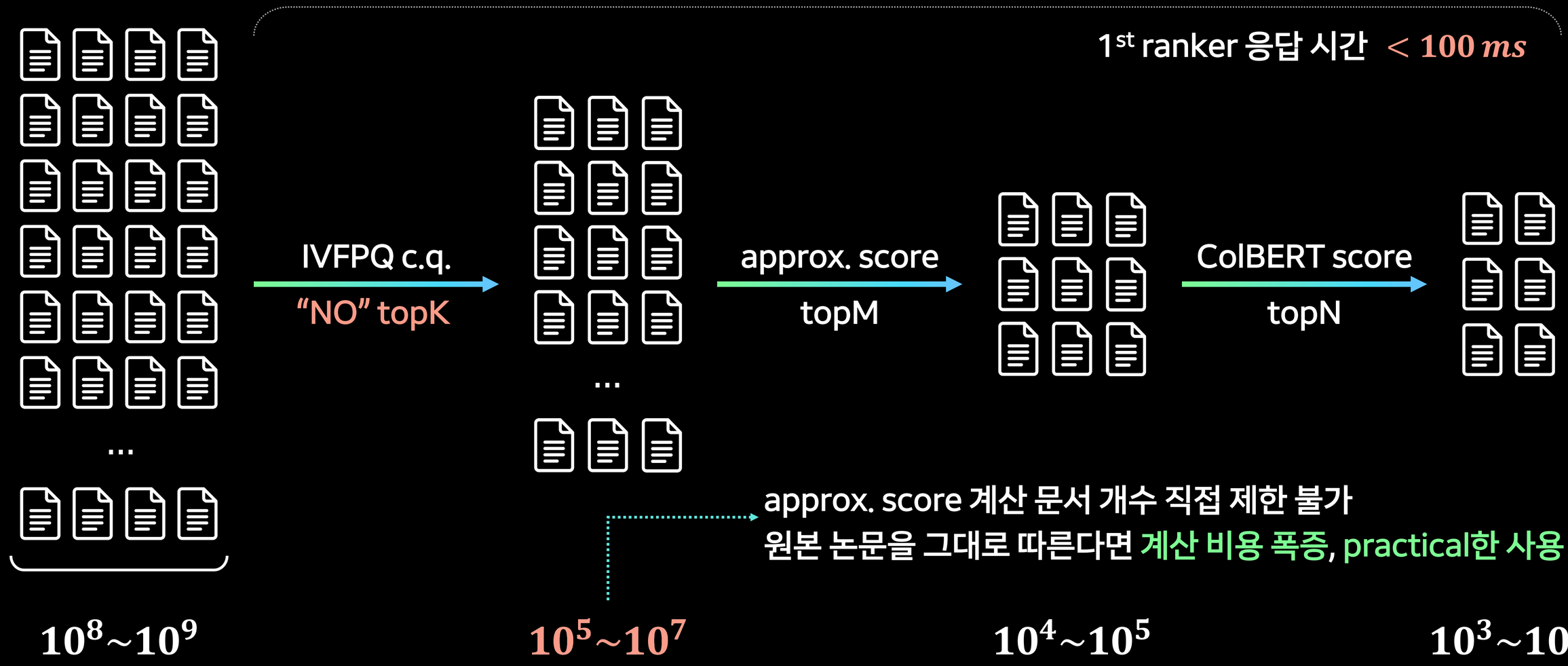
4.1.8 NAVER-2 prototype



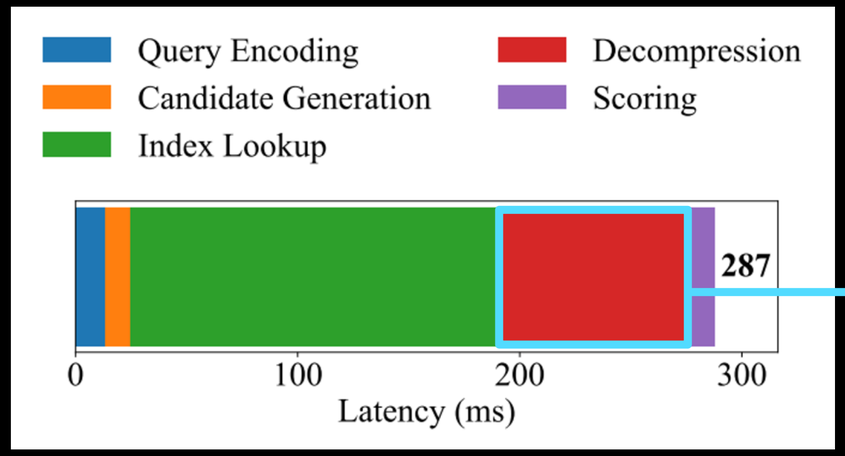
4.2.1 이걸로 끝인 줄 알았는데...

	approx. score	ColBERT score
		
문서 별 score 계산 복잡도	$dimension \times (\#query) \times 3$	$dimension \times (\#query) \times 5$
계산 대상 문서 수		

4.2.2 왜 문제가 되는가: b/c we are 1st ranker



4.2.3 해결 방안1: 벡터 압축 기능 삭제



ColBERTv2 profile

ColBERTv2	NAVER-2
IVFPQ	IVF
벡터 압축 'PQ' 사용	벡터 압축 미사용
색인 사이즈 절감	ColBERT(HNSW) 대비 색인 사이즈 2배 감소

PQ로 인한 decompression이 쿼리 응답 시간의 30% 차지

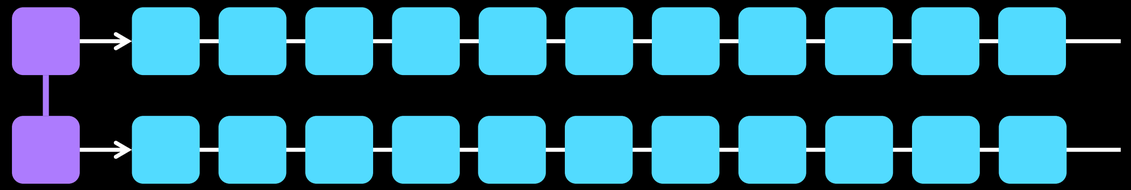
Practicality 위해 IVFPQ 대신 IVF 선택

Image from [Santhanam et al., CIKM 2022]

4.2.4 해결 방안2: finer granularity 선택

ColBERTv2

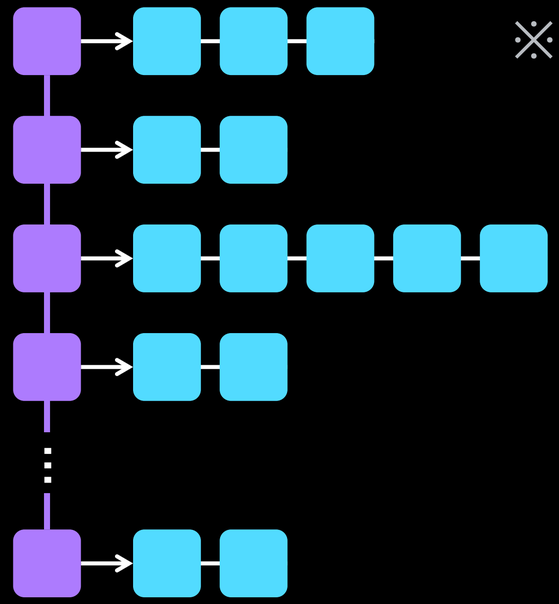
$nlist(= \#cell) \propto \sqrt{(\#vector)}$



NAVER-2

$nlist \gg \sqrt{(\#vector)}$

IVF 권장 설정보다 조밀하게 cluster 분할
approx. score 계산 비용 폭증 방지



※ e.g., $nlist = (\#vector) \times 0.006$

4.2.5 finer granularity의 side-effect

문제

Query 처리 속도 저하

IVF c.q. 탐색 방식: ENN

Index 생성 속도 저하

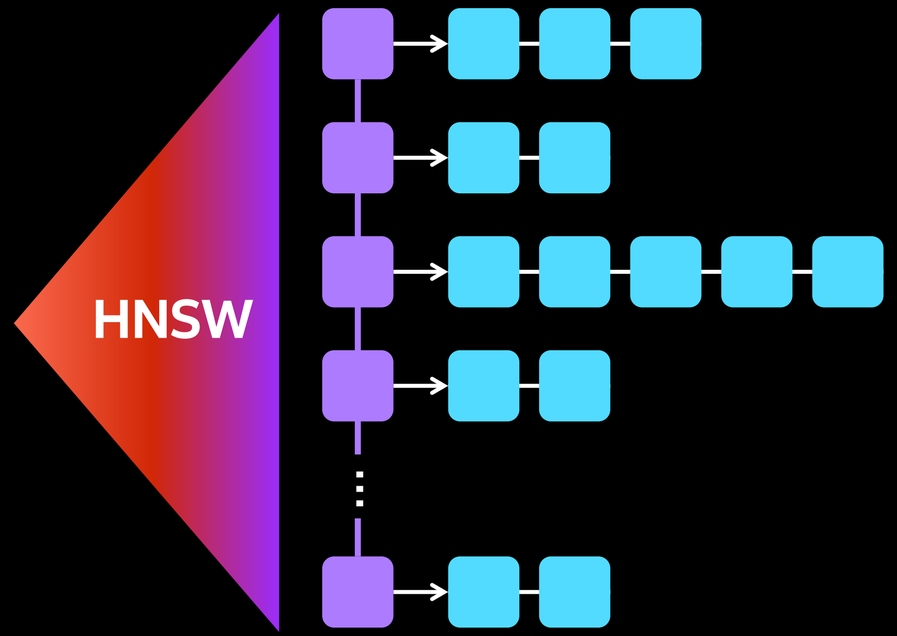
Index time 거리 계산 overhead

해결 방안

IVF-HNSW 활용

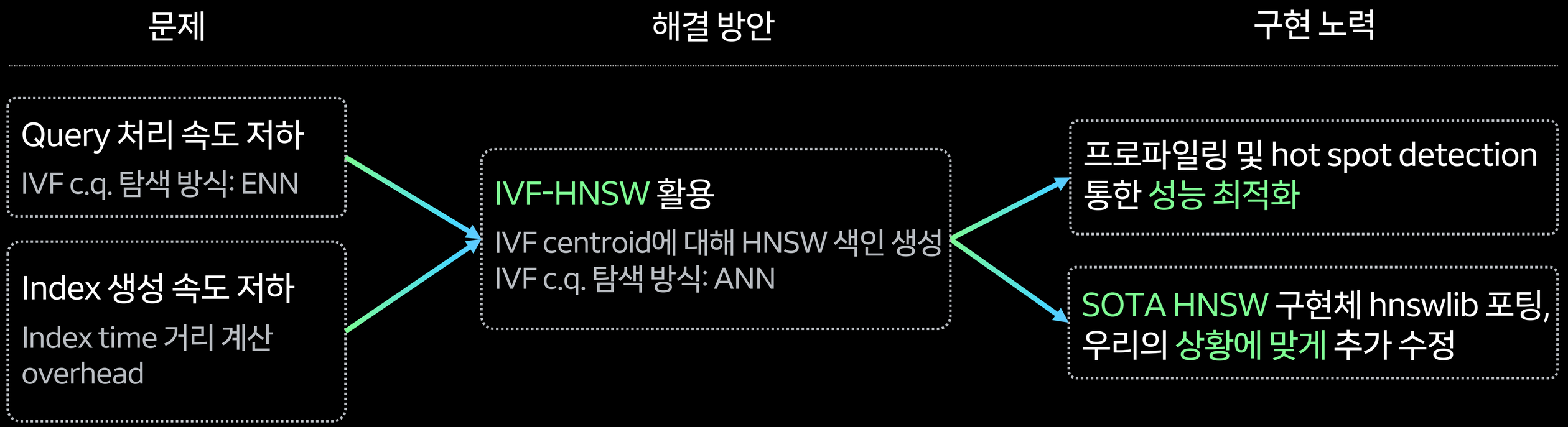
IVF centroid에 대해 HNSW 색인 생성

IVF c.q. 탐색 방식: ANN



Practicality 위해 IVF-HNSW 선택

4.2.5 finer granularity의 side-effect



Practicality 위해 IVF-HNSW 선택
극한의 성능을 위한 인하우스 구현

4.2.6 HNSW vs. IVF-HNSW

	ColBERT w/ HNSW	NAVER-2 w/ IVF-HNSW
ANN 색인 사이즈	$2 \times (\#vector)$	$(\#vector) + 2 \times nlist$
ANN retrieval complexity	$log(\#vector)$	$log(nlist)$
ANN retrieval result (approx. score 계산 비용)	$topK$	affordable w/ finer granularity

※ $nlist(= \#cell) \gg \sqrt{(\#vector)}$

4.3.1 NAVER-2 vs. ColBERT (색인)

NAVER-2의 IVF-HNSW

nlist	색인 사이즈 (GiB)	색인 생성 시간 (s)
$22K$	6.88	97.25
$44K$	6.89	143.69
$88K$	6.91	283.79
$176K$	6.96	629.53

ColBERT의 HNSW

색인 사이즈 (GiB)	색인 생성 시간 (s)
14.35	622.2

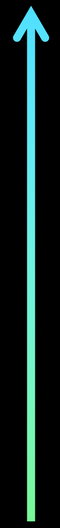
※ $(\#vector) = 30M$

4.3.2 NAVER-2 vs. ColBERT (응답 시간)

NAVER-2 평균 응답 시간 (ms)

※ $nlist = 176K$

$nprobe$ \ $topM$	120	240	360	480
1	23.49	38.38	46.00	48.26
8	34.43	50.63	67.74	85.24
16	46.74	61.28	79.31	98.20
24	57.29	76.94	94.12	107.92
32	72.14	86.91	106.79	119.79



ColBERT 평균 응답 시간

86.58

4.3.3 NAVER-2 vs. ColBERT (품질)

※ $nlist = 176K$

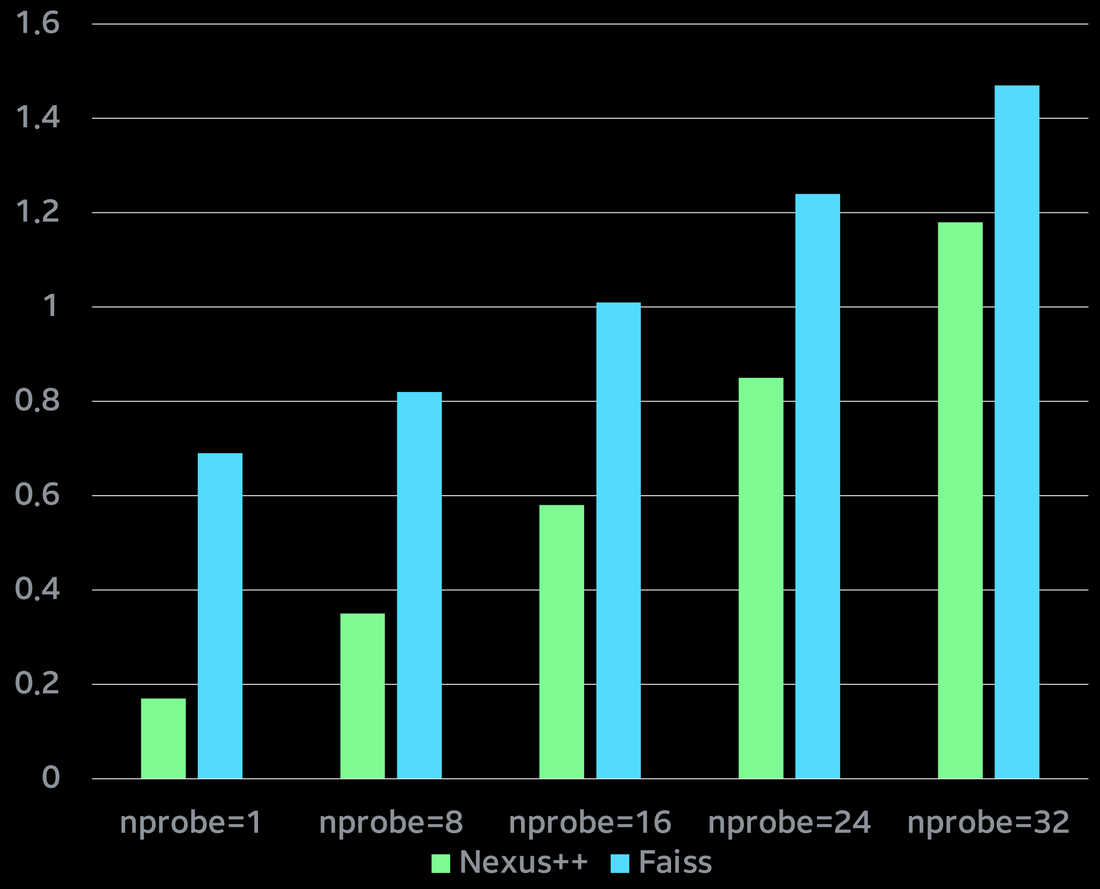
		$topM$			
		120	240	360	480
$nprobe$	1	-392	-388	-386	-386
	8	-228	-169	-118	-84
	16	-123	-27	15	49
	24	-61	26	83	111
	32	-8	68	124	151

검색 결과 평균 ColBERT score

NAVER-2 > ColBERT

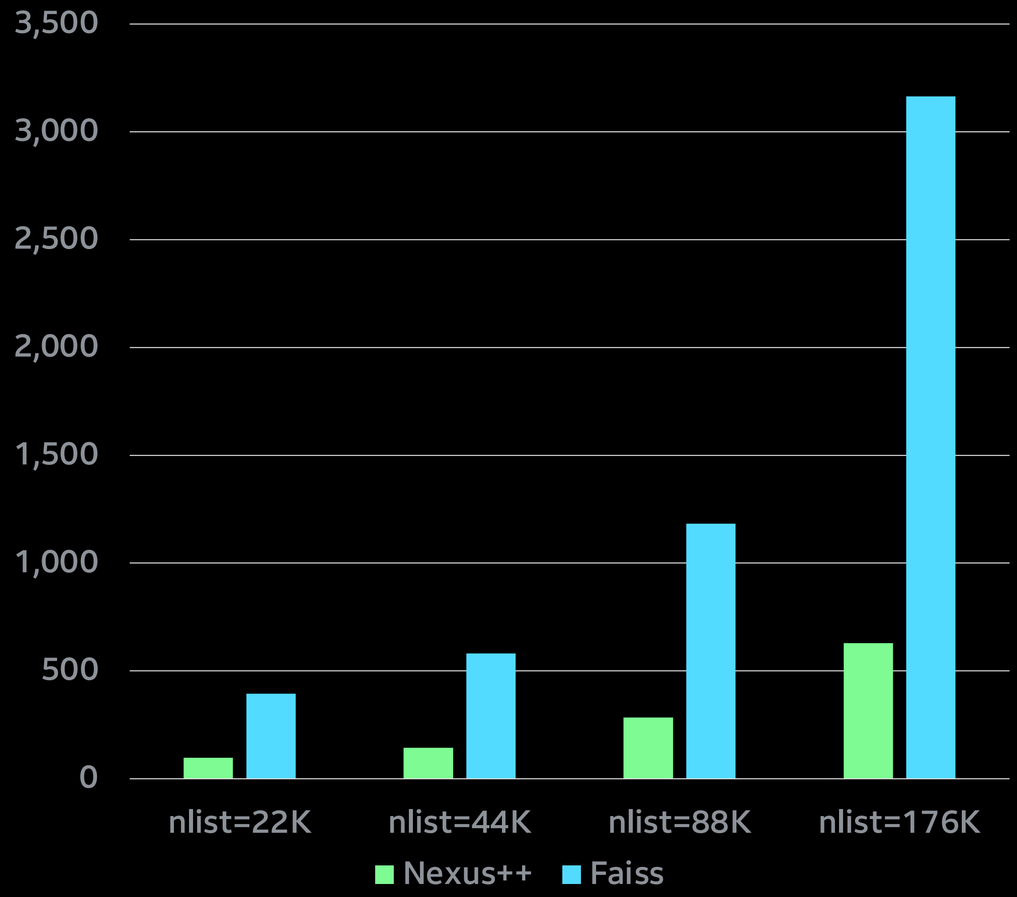
4.3.4 Nexus++ vs. Faiss

인덱스 retrieval 응답시간 (ms)



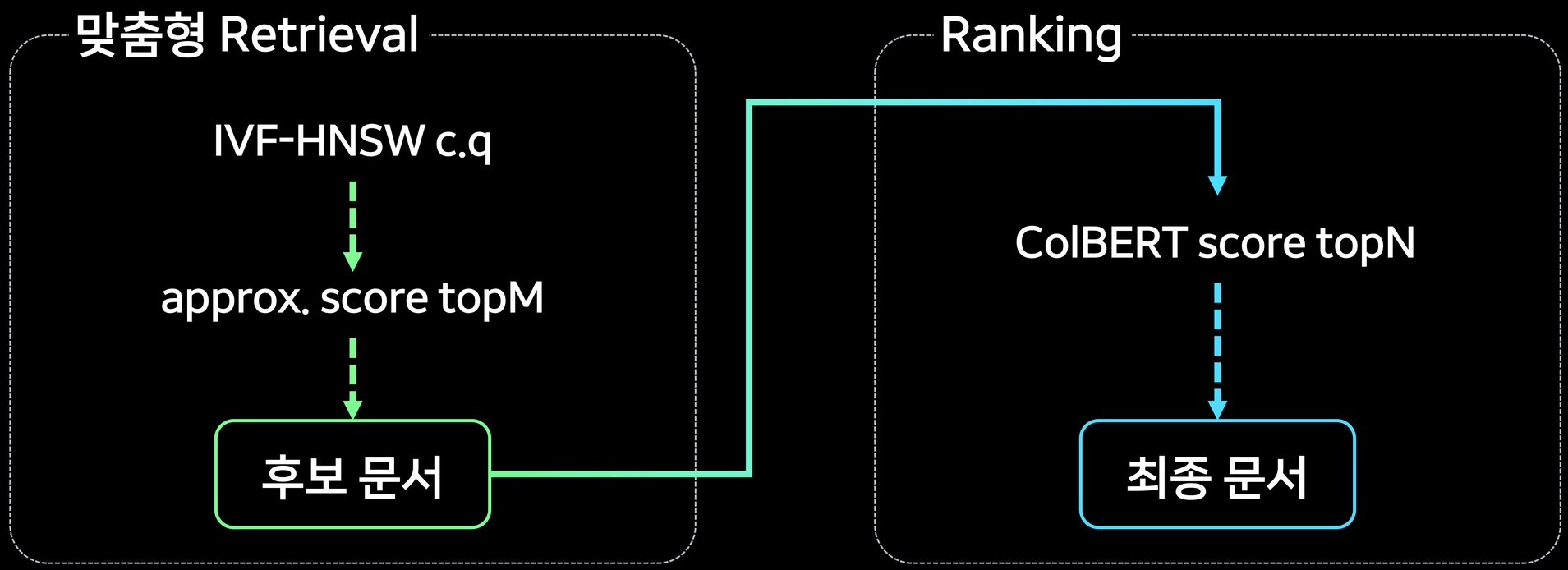
※ $nlist = 176K$

인덱스 생성 시간 (s)



※ $(\#vector) = 30M$

4.3.5 NAVER-2 모델 요약



색인 크기 IVF-HNSW 사용, HNSW 대비 색인 크기 2배 감소

속도-품질 trade-off query time 파라미터인 nprobe, topM을 서비스 필요에 따라 선택 가능

5. Solution Comparison & Future Work

ColBERT vs. NAVER-1 vs. NAVER-2 vs. …

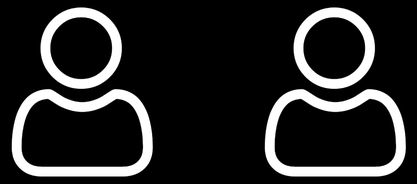
5.1 Our Model Comparison

	ColBERT	NAVER-1	NAVER-2
색인 종류 및 사이즈	HNSW, 1	HNSW, 1	IVF-HNSW, 0.49
Retrieval 방식	single vector ANN	single vector ANN	맞춤형 ANN
Ranking 방식	ColBERT score	approx. score	ColBERT score
qps	1	2.02	0.72 – 3.69
품질	-	acceptable degradation*	adjustable†

* ndcg@5, mrr@10 기준 95%

† 비교 score 기준 -392 ~ +151

5.2 So What’s the Solution?



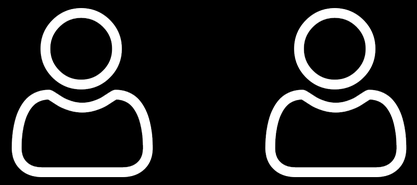
백엔드 (어른들의 사정)

“벡터 데이터와 색인은 그대로,
검색 API 속도가 빨라지는 것만 원해요”

“장비 감축이 가능할까요?”

“NAVER-1”

엔진팀



모델러 (어른들의 사정)

“총 벡터 개수를 2배로 늘리고 싶어요”
(꿈의 1000억 벡터로 서빙 희망)

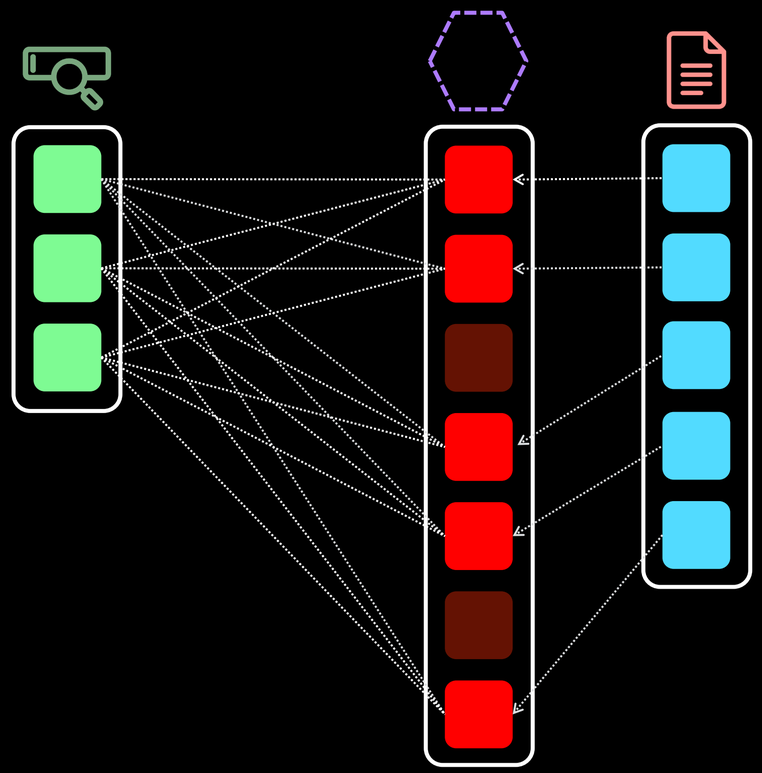
“장비 증설은 어렵습니다”

“NAVER-2”

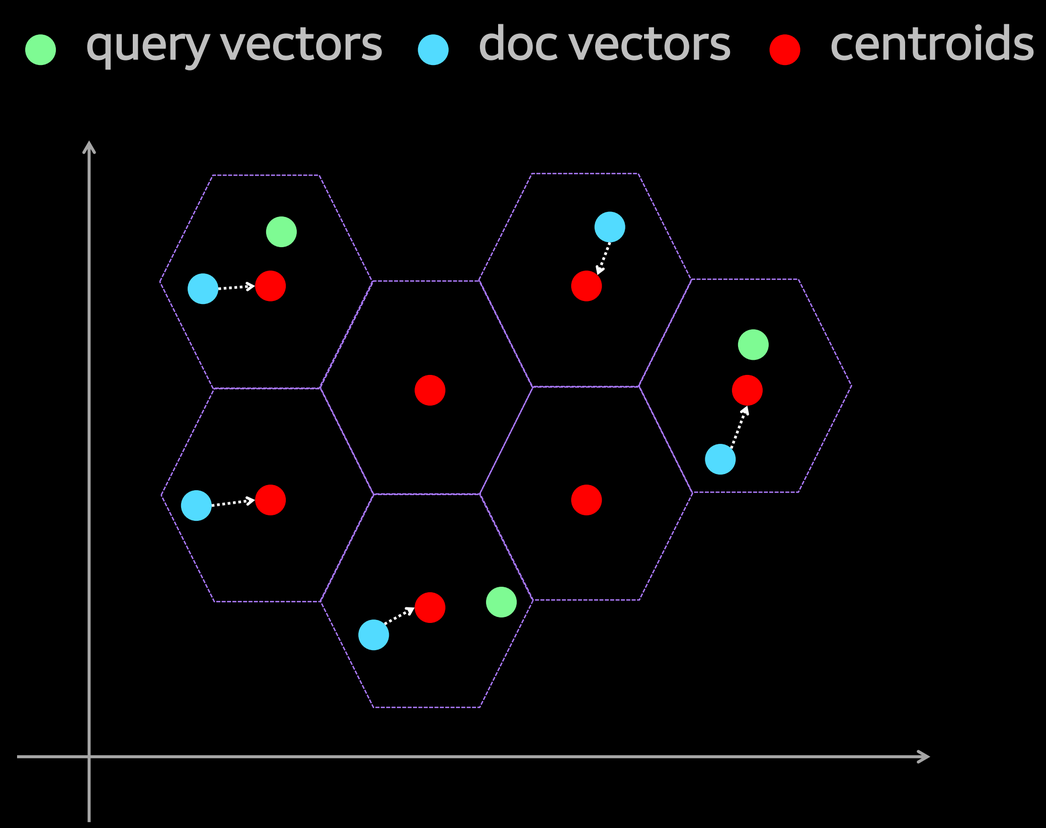
엔진팀

5.3.1 What's Next: short term

centroid interaction score

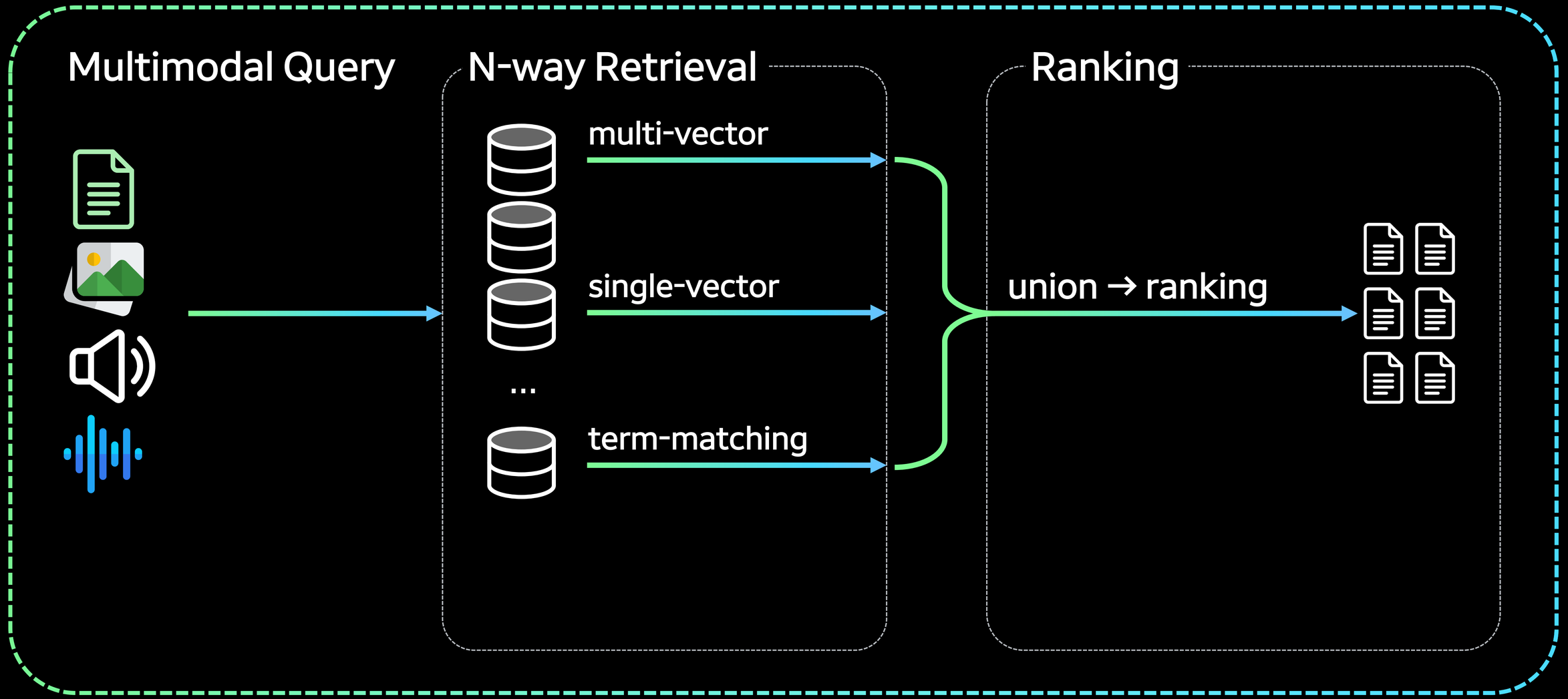


또다른 맞춤형 retrieval

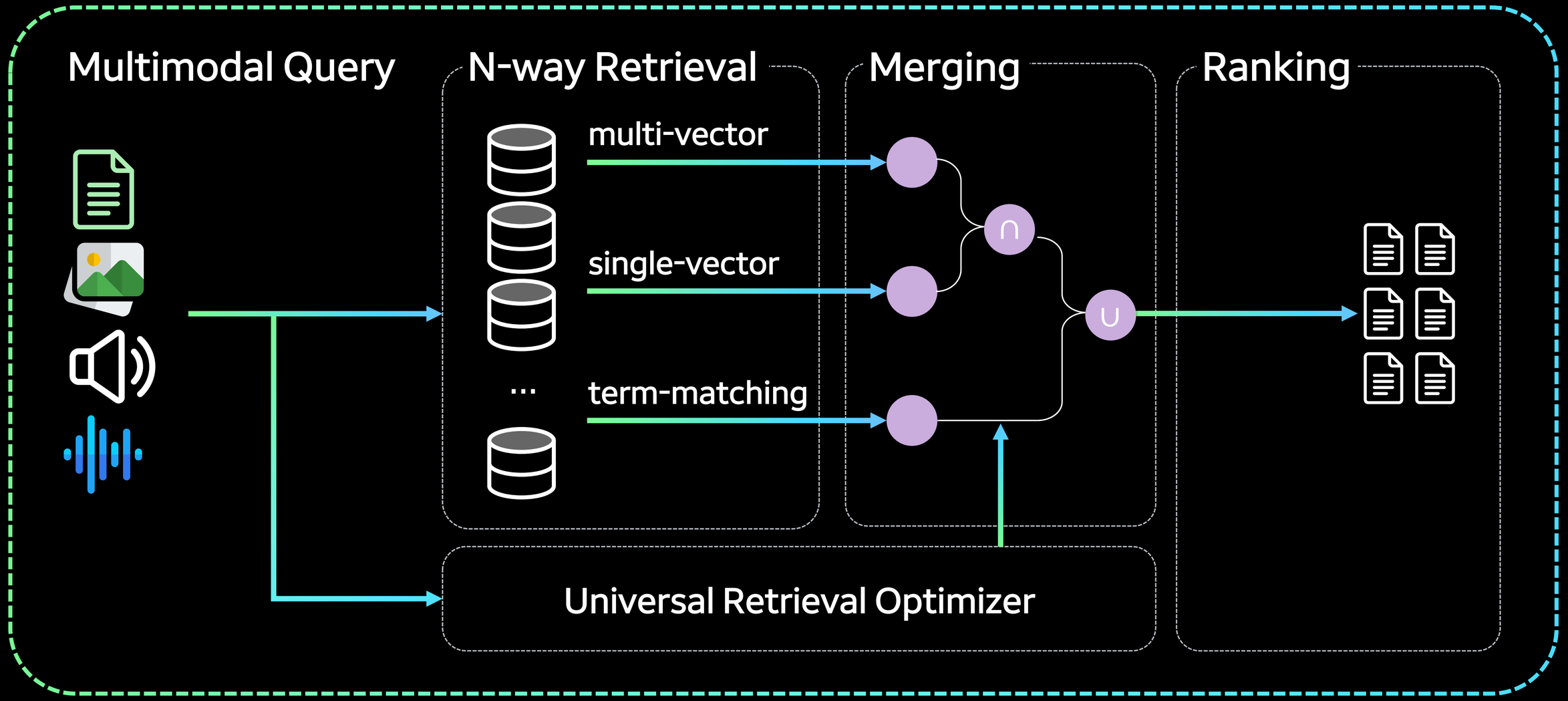


PLAID [Santhanam et al., CIKM 2022]

5.3.2 What's Next: long term



5.3.2 What's Next: long term



Thank you

Q&A