MUVERA:Multi-Vector Retrieval via Fixed Dimensional Encodings

APL 김주희

목차

1. Prerequisites
Vector DB vs. Vector Indexing

2. Overview of the Paper
Vector DB vs. Vector Indexing

3. FDE
What is FDE(Fixed Dimensional Encoding)

4. How FDE is Constructed

5. Strengths of MUVERA

6. Limitation of MUVERA
What is FDE(Fixed Dimensional Encoding)

7. Experimental Results

8. Conclusion

1. Prerequisites

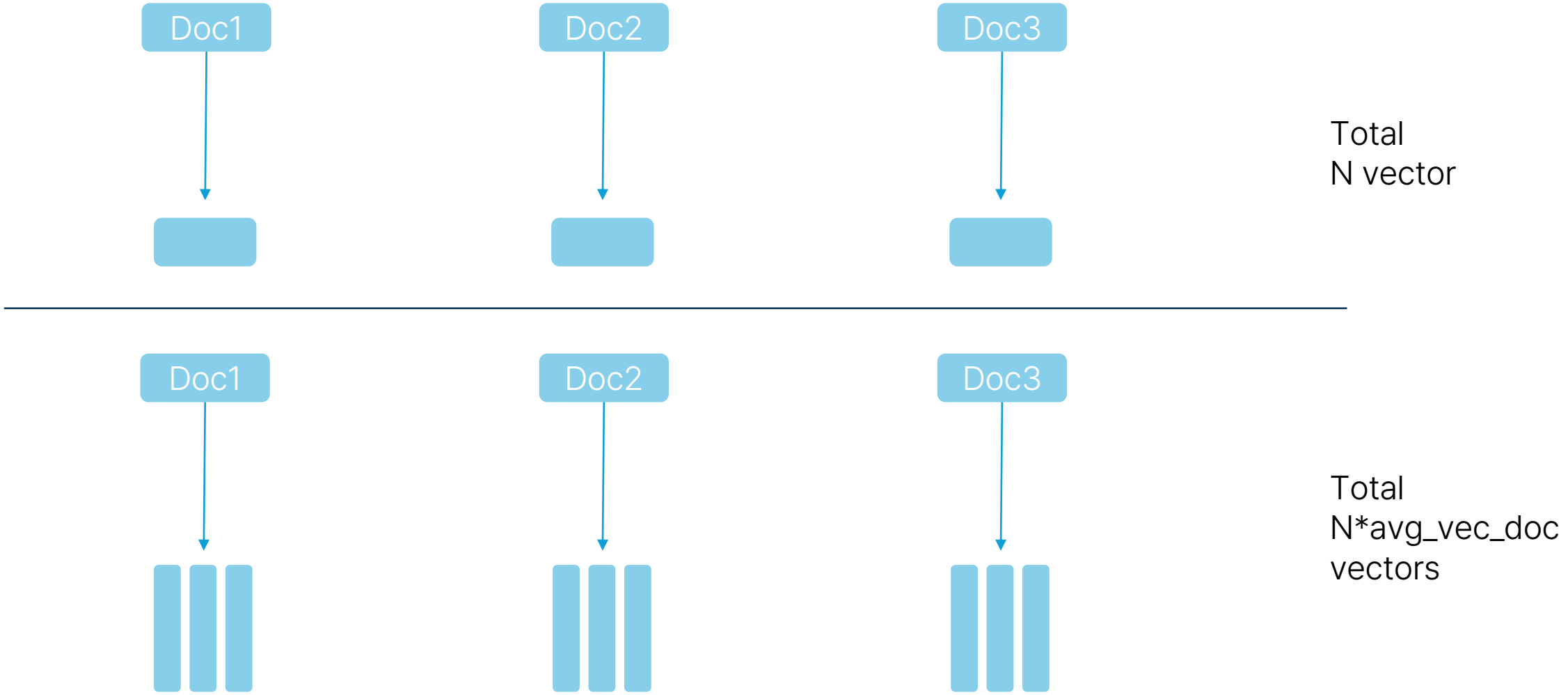
Vector database vs Vector Indexing

- **Vector DB**
  - 벡터 임베딩을 메타데이터와 함께 저장하고 관리하는 시스템
  - CRUD를 지원
  - 하이브리드 검색, 필터링, 권한 제어 등 지원
  - 예) Weaviate, Pinecone, Milvus 등
- **Vector Indexing**
  - 벡터끼리의 유사도 계산을 빠르게 하기 위한 알고리즘
  - 예) FAISS, HNSW, ScANN

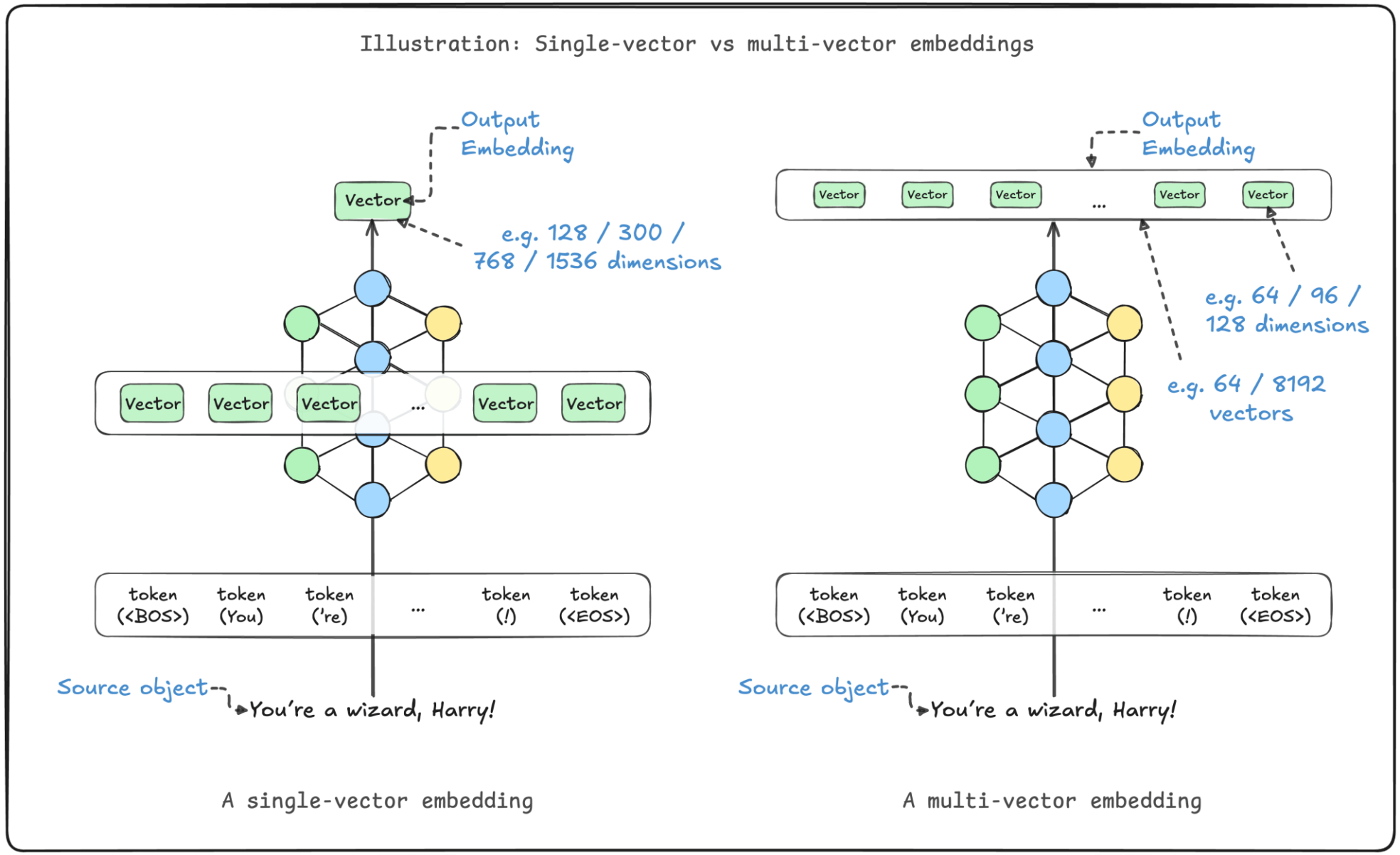
2. Overview of the Paper

- **목적**
  - Multi-vector 문서 검색을 single vector 검색처럼 효율적으로 하는 것
- **Motivation**
  - 메모리 감소
  - Vector search의 CPU overhead
- **Contribution**
  - FDE (Fixed dimensional Encoding) 제안
  - 내적 기반 Chamfer 유사도 근사
  - 빠른 후보 검색 + 정확한 Reranking

2. Overview of the Paper



2. Overview of the Paper



2. Overview of the Paper

- maxSim 유사도 계산
  - 문서 임베딩 D, 쿼리 임베딩 Q

$$\mathrm{sim}(D, Q) = \sum_{q \in Q} \max_{d \in D} q \cdot d$$

3. FDE

What is the FDE?

- 여러 벡터로 표현된 문서를 고정된 하나의 벡터로 변환
- 클러스터링 + 차원 축소 + 반복을 통해 하나의 벡터로 압축
- 내적으로 chamfer 유사도 근사 가능

3. FDE

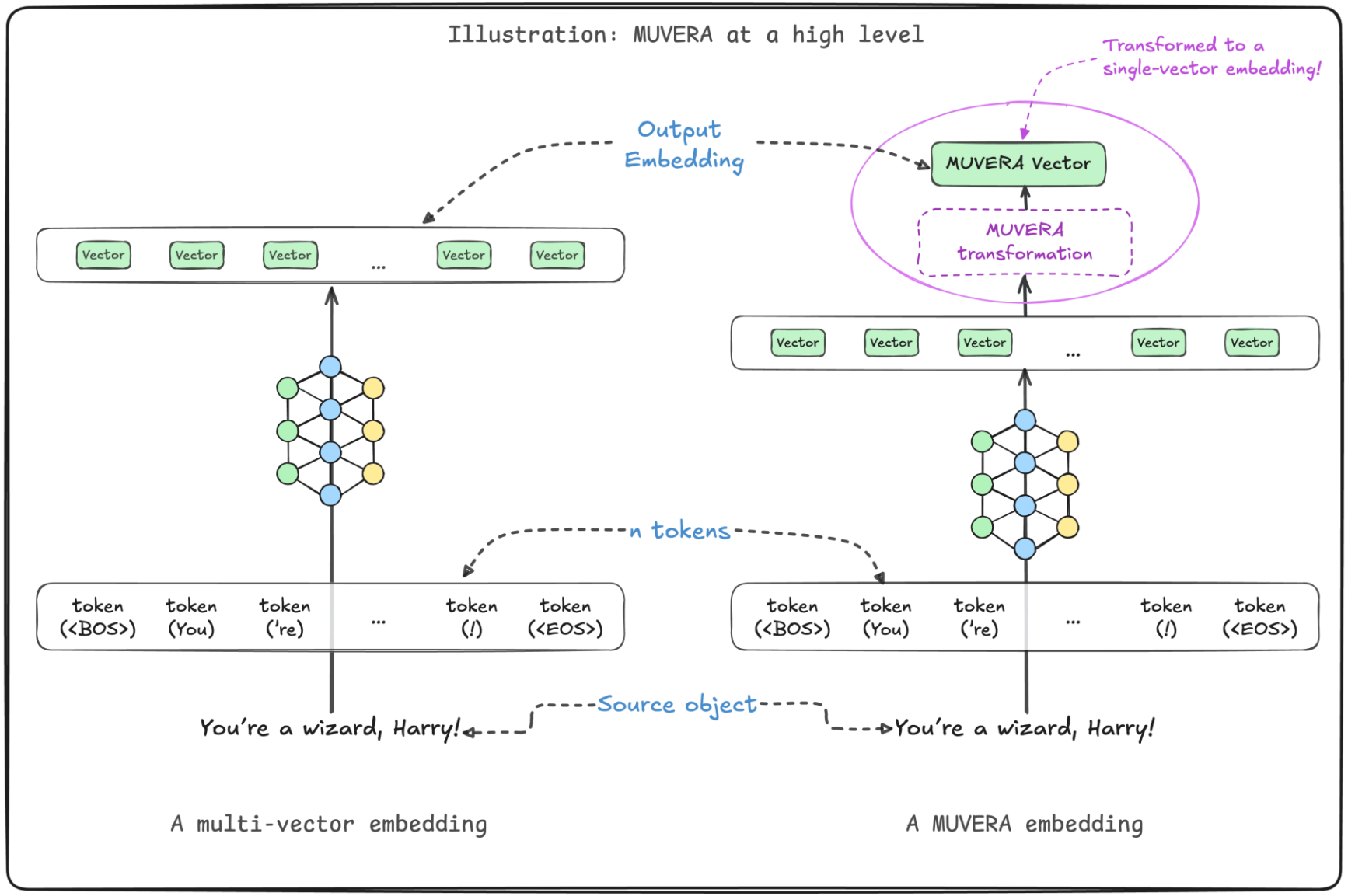


Illustration: MUVERA at a high level
Transformed to a single-vector embedding!
Output Embedding
MUVERA Vector
MUVERA transformation
Vector
Vector
Vector
...
Vector
Vector
n tokens
token (<BOS>)
token (You)
token ('re)
...
token (!)
token (<EOS>)
Source object
You're a wizard, Harry!
You're a wizard, Harry!
A multi-vector embedding
A MUVERA embedding

3. FDE

Key Idea

- Multi-vector embedding D
- Single-vector embedding d_single

$$encode(x_{multi}) \Longrightarrow x_{single}$$
$$x \in D, Q$$

$$maxSim(D, Q) \approx d_{single} \cdot q_{single}$$
$$where$$
$$d_{single} = encode(D) \wedge q_{single} = encode(Q)$$

4. How FDE is Constructed

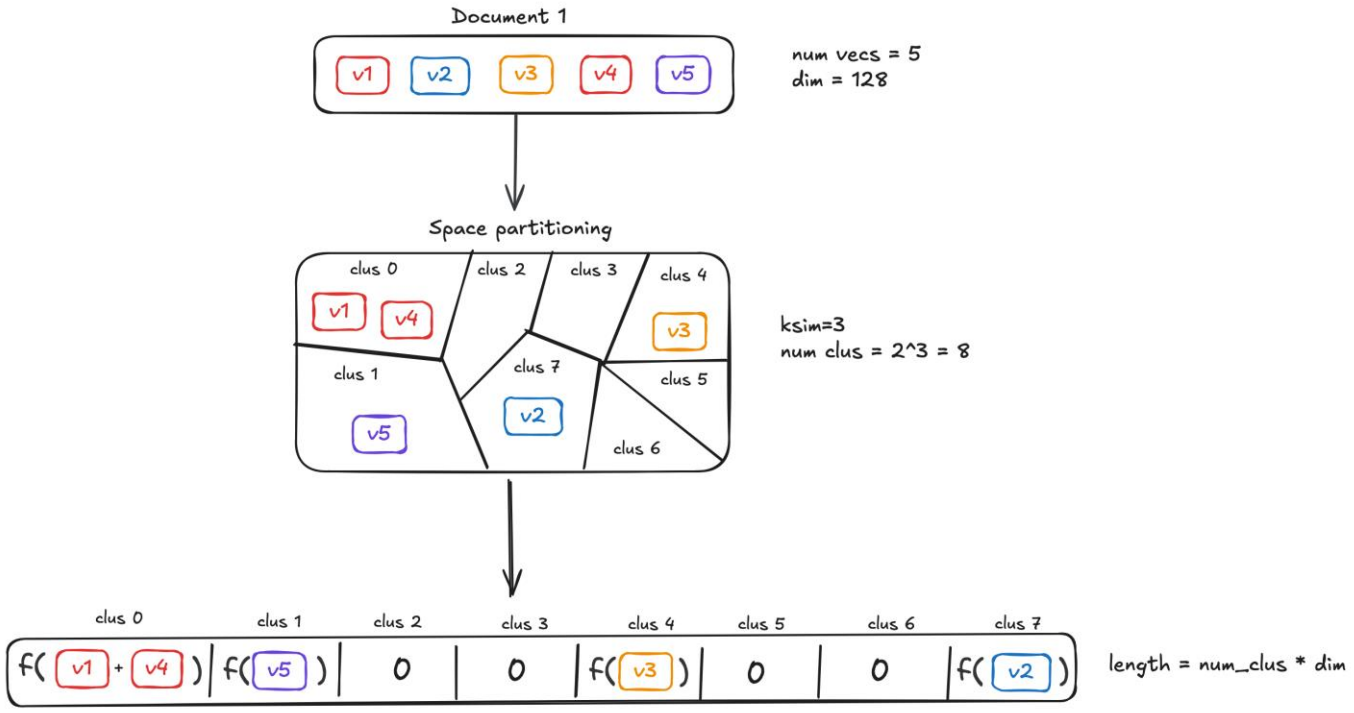
Four Steps

1. Space partitioning
2. Dimensionality reduction
3. Multiple repetitions
4. Final Projection

4. How FDE is Constructed

Space Partitioning

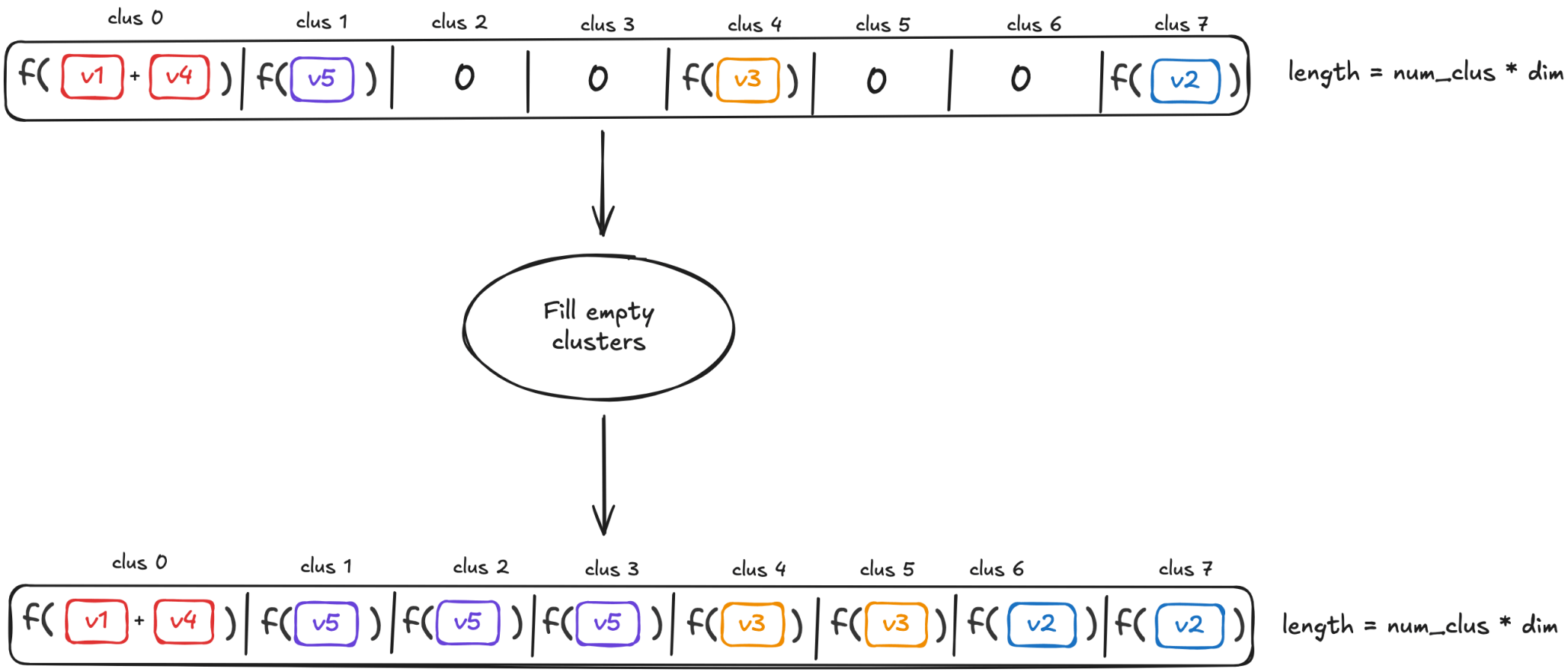
- **SimHash로 벡터 공간을 B개의 클러스터로 분할**
  - 함수 φ(x) 를 알 수 있다면 k번째 클러스터로 들어갈 수 있음
  - 이 때, 클러스터 k는 총 B개 중 하나이므로 k ∈ {1, 2, ..., B} 로 표현 가능
  - g1, . . . , gksim ∈ Rd , and set φ(x) = (1(⟨g1, x⟩ > 0), . . . , 1(⟨gksim , x⟩ > 0)), where 1(·) ∈ {0, 1} is the indicator function



4. How FDE is Constructed

Space Partitioning

- Fill empty cluster



4. How FDE is Constructed

Space Partitioning

- **최종 단일 벡터**
  - 문서에서 클러스터별 vector d_k 만들기
  - 쿼리 벡터는 합만 구함
  - D1부터 db까지 이어붙여 하나의 벡터로 만듦

$$d_k = \frac{1}{|D_k|} \sum_{d \in D, \phi(d)=k} d \qquad q_k = \sum_{q \in Q, \phi(q)=k} q$$

$$d_{\text{single}} = [d_1, d_2, \ldots, d_B]$$

4. How FDE is Constructed

Dimensionality Redcution & Multiple repetitions

- D_k를 줄이기
  - S∈R d proj ×dim :±1로 구성된 랜덤 행렬(JL 렘마 근거)
  - D_proj: 축소하고 싶은 목표 차원 예)16, 32
  - $\psi(dk)$: 압축된 d_k 벡터

$$\psi(d_k) = \frac{1}{\sqrt{d_{\text{proj}}}} \cdot S \cdot d_k$$

$$d_{\text{single}} = [\psi(d_1), \psi(d_2), \ldots, \psi(d_B)]$$

5. Strengths of MUVERA

- 기존 Multi-vector 방식보다 훨씬 빠른 검색 속도 제공
- Dot product 기반이라 GPU최적화가 가능함
- 다양한 벡터 수와 구조에 유연하게 대응할 수 있음

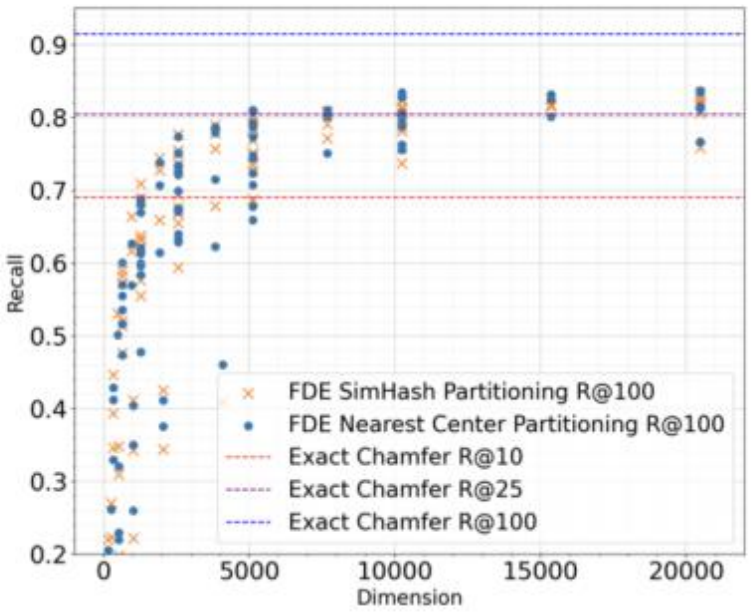
6. Limitation of MUVERA

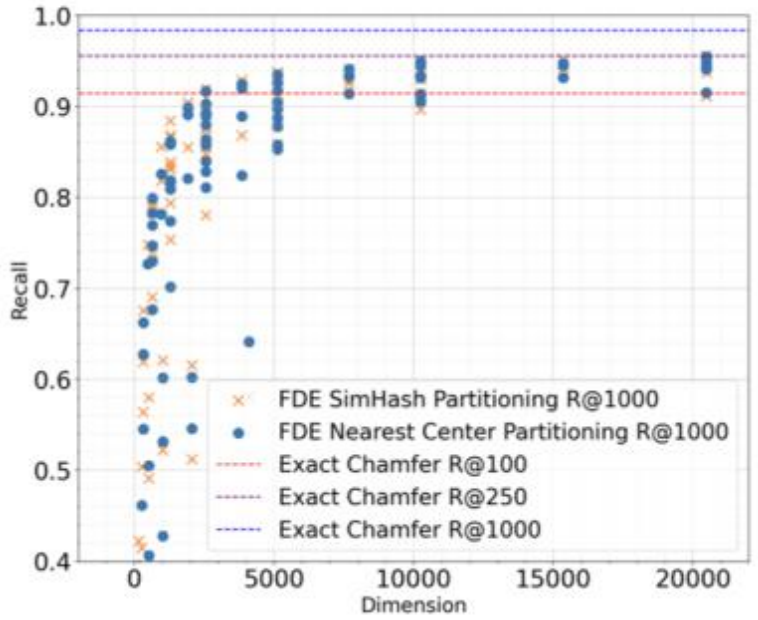
- 근사 방식이기 때문에 정확도 손실 가능성 존재
- 하이퍼파라미터 설정이 복잡함
- 빈 클러스터 보정이 완벽하지 않음
- 일부 데이터셋에서 다른 방식보다 성능이 낮음

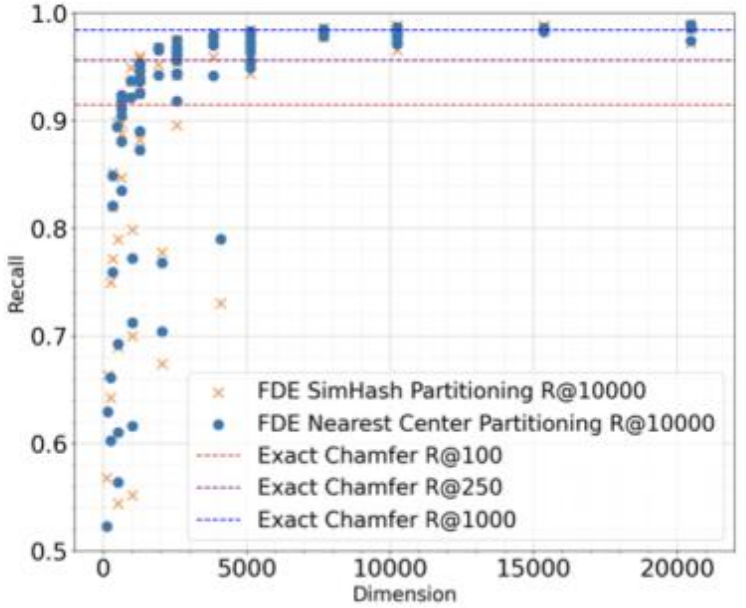
7. Experimental Results

- **비교 대상**
  - MaxSim (Chamfer): 정확하지만 느림
  - PLAID: 빠른 multi-vector 근사 방법
  - Dense Retrieval: single-vector만 쓰는 방식
- **데이터셋**
  - BEIR (MS MARCO, NQ, TREC-COVID 등 다수)
  - LoTTE (Long-form QA)
  - MS MARCO (웹검색 쿼리 기반)
- **평가지표**
  - Recall@k
  - 검색 속도 (query per second)
  - 메모리 사용량

7. Experimental Results







7. Experimental Results

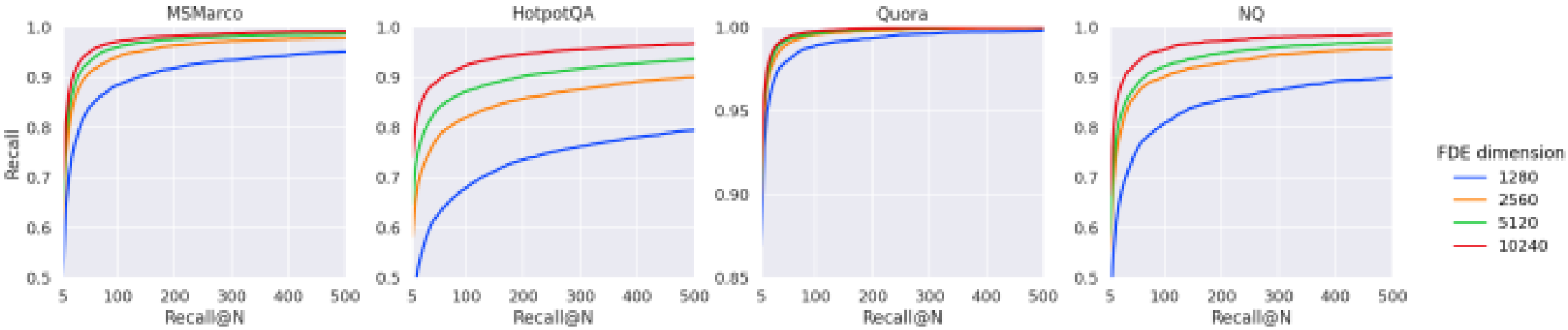


Figure 4: Comparison of FDE recall versus brute-force search over Chamfer similarity.

7. Experimental Results

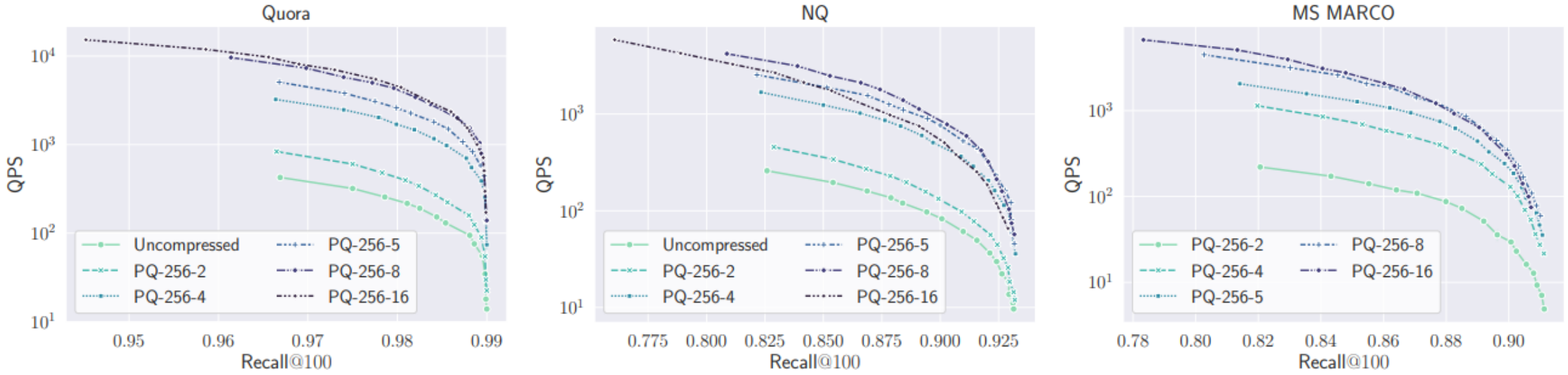


Figure 6: Plots showing the QPS vs. Recall@100 for MUVERA on a subset of the BEIR datasets. The different curves are obtained by using different PQ methods on 10240-dimensional FDEs.

7. Experimental Results

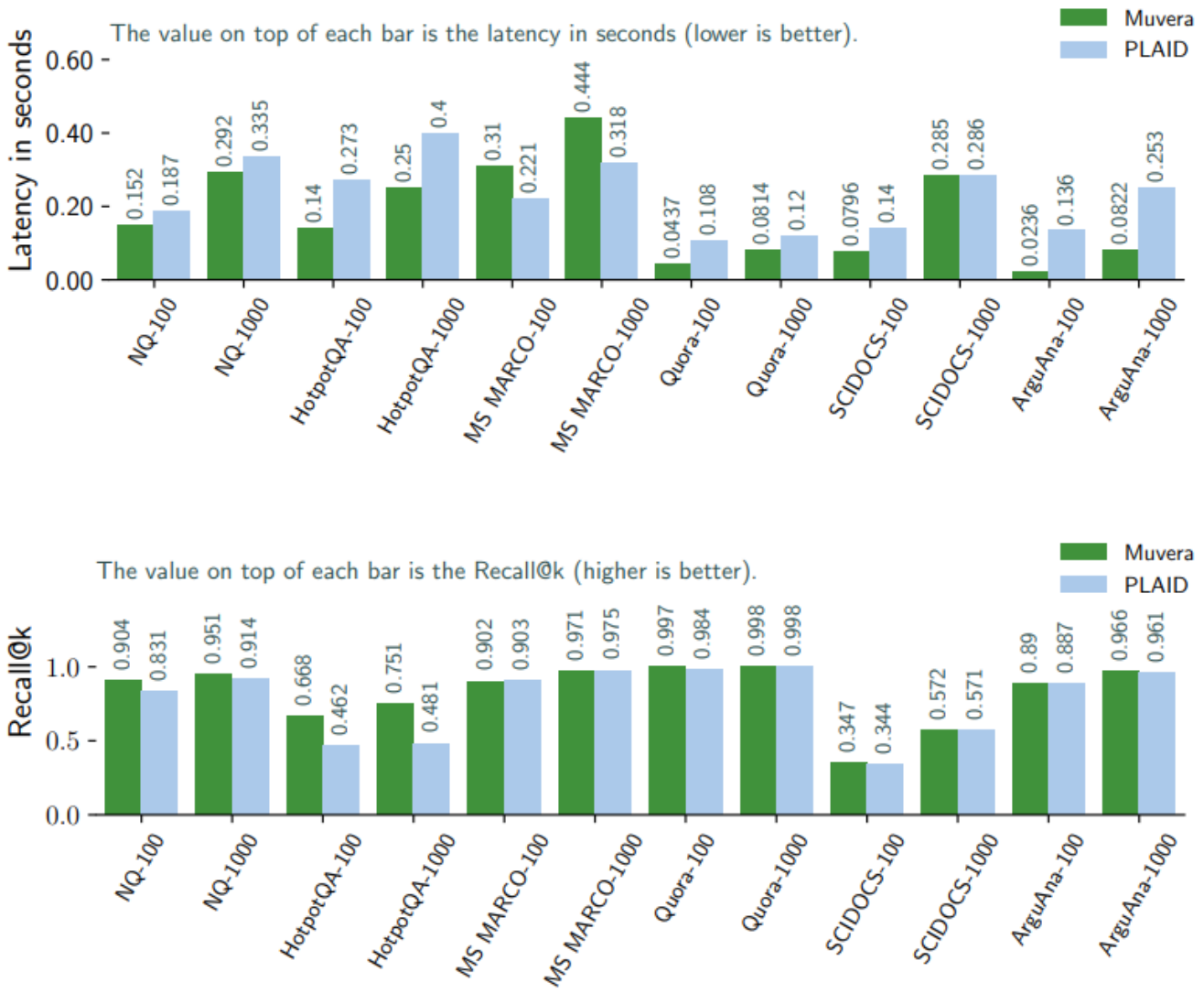


Figure 7: Bar plots showing the latency and Recall@k of MUVERA vs PLAID on a subset of the BEIR datasets. The x-tick labels are formatted as dataset-k, i.e., optimizing for Recall@k on the given dataset.

8. Conclusion

- MUVERA는 효율성과 정확도의 균형을 맞춘 최신 vector 검색 방식
- 실무 적용 시 튜닝이 필요하지만 가치가 충분함
- 하이브리드 방식 등으로 향후 발전 가능성이 있음

9. Experimental

- Dataset: https://huggingface.co/datasets/google-research-datasets/go_emotions
- 실험 구성: muvera의 ksim, dproj, reps를 다르게 설정하여 메모리와 속도를 실험