

강주란
홍석범

RAG 구성을 위한 ElasticSearch 활용

Retrieval
Augmentation
Generation

INDEX

목차

1. RAG와 ElasticSearch

A. RAG를 위한 Database

2. 텍스트 기반 검색

A. 색인

B. 쿼리

3. Vector 검색

A. Text 기반 검색과의 차이점

B. 거리를 측정하는 계산식

C. ANN

4. 하이브리드 검색

A. Text 와 Vector의 조합

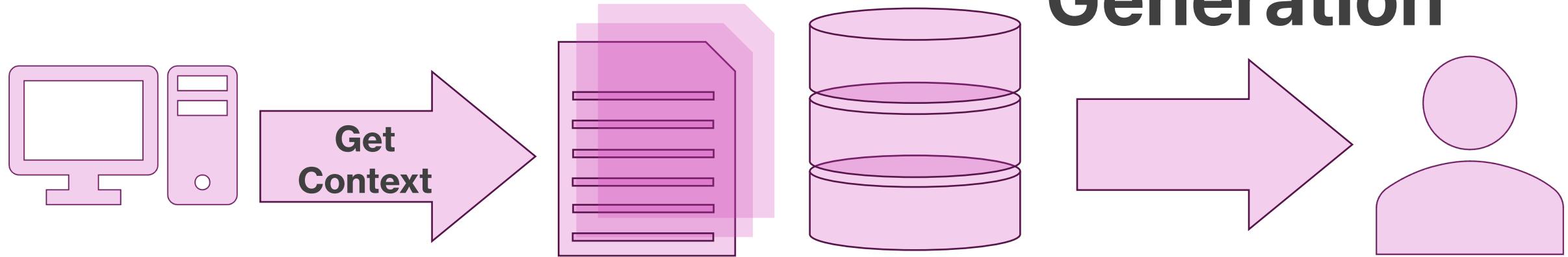
1. RAG와 Elasticsearch

A. RAG를 위한 Database

RAG and Elasticsearch

RAG와 ElasticSearch

RAG 를 위한 Database



Retrieval Augmented

검색 모델로 정보 검색 후 벡터 DB에 저장.
이후 쿼리와 관련성을 기준으로 검색된 정보의 순위를 매김

LLM을 통한 답변 생성
Generation

RAG and Elasticsearch

RAG와 ElasticSearch

RAG 를 위한 Database

ElasticSearch
벡터 DB 활용



색인
벡터를 주어진 데이터 구조에 매핑

쿼리
인덱스 벡터를 쿼리 벡터와
비교하여 최근접 벡터 항목을 결정

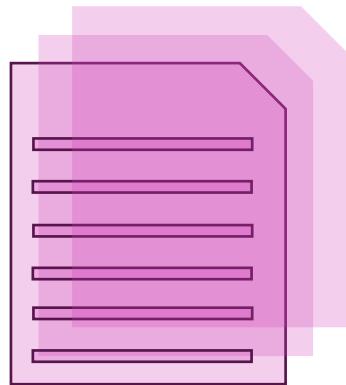
2. 텍스트 기반 검색

- A. 색인(Index)
- B. 쿼리(Query)

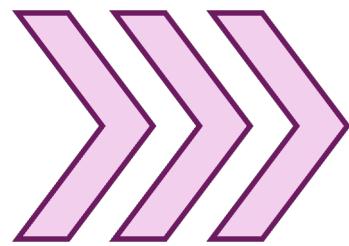
Text Based Search

Text 기반 검색

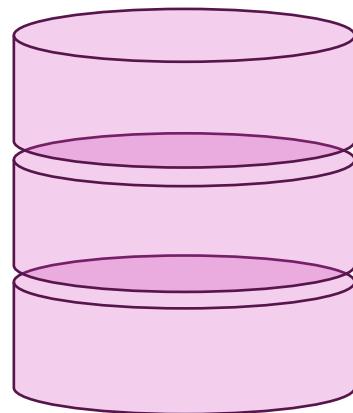
RAG 를 위한 Database



Raw Data



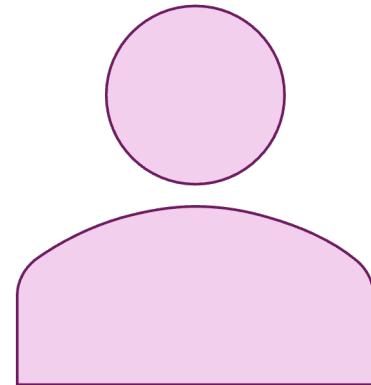
Indexing



Index



Search



User

Text Based Search

indexing 색인

데이터가 검색될 수 있는 구조로 변경하기 위해
원본 문서를 검색어 토큰으로 변환하여 저장하는 일련의 과정

Inverted
Indexing

text analysis

Analyzer

Char Filters

Tokenizer

Token Filters

Text 기반 검색

indexing 색인

데이터가 검색될 수 있는 구조로 변경하기 위해
원본 문서를 검색어 토큰으로 변환하여 저장하는 일련의 과정

Inverted Indexing

text analysis

Analyzer

Char Filters

Tokenizer

Token Filters

Text Based Search

Text 기반 검색 – 기존의 RDBMS

역 인덱싱

저장 방식

- 테이블 구조로 저장

ID	Content
1	The quick brown fox
2	The quick brown fox jumps over the lazy dog
3	The quick brown fox jumps over the quick dog
4	Brown fox brown dog
5	Lazy jumping dog

Text Based Search

Text 기반 검색 – 기존의 RDBMS

역 인덱싱

저장 방식

- 테이블 구조로 저장

검색 방식

- LIKE 검색을 통해
ROW안의 모든 내용 탐색

ID	Content
1	The quick brown fox
2	The quick brown fox jumps over the lazy dog
3	The quick brown fox jumps over the quick dog
4	Brown fox brown dog
5	Lazy jumping dog

ID	Content
1	The quick brown fox
2	The quick brown fox jumps over the lazy dog
3	The quick brown fox jumps over the quick dog
4	Brown fox brown dog

Text Based Search

Text 기반 검색 – Elasticsearch

역 인덱싱

저장 방식

- 각 키워드(term)의 내용이 어떤 도큐먼트에 있는지 저장

Term	ID
quick	1,2,3
brown	1,2,3,4
jump	2,3,5
dog	2,3,4,5
lazy	2,5
fox	1,2,3,4
over	2,3
fast	1,2,3,4

Text Based Search

Text 기반 검색 – Elasticsearch

역 인덱싱

저장 방식

- 각 키워드(term)의 내용이 어떤 도큐먼트에 있는지 저장

Term	ID
quick	1,2,3
brown	1,2,3,4
jump	2,3,5
dog	2,3,4,5
lazy	2,5
fox	1,2,3,4
over	2,3
fast	1,2,3,4

검색 방식

- Term의 역인덱스가 가리키는 문서ID를 찾음

Term	ID
fox	1,2,3,4

Text 기반 검색 Based Search

indexing
색인

데이터가 검색될 수 있는 구조로 변경하기 위해
원본 문서를 검색어 토큰으로 변환하여 저장하는 일련의 과정

Inverted
Indexing

text analysis

Analyzer

Char Filters

Tokenizer

Token Filters

Analyzer

Char Filters

Tokenizer

Token Filters

텍스트 분석 중 가장 먼저 처리되는 과정으로,
색인된 텍스트가 토크나이저에 의해 텀으로 분리되기 전에 전체 문장에 대해 적용됨

HTML Strip

HTML 태그를 제거하여
일반 텍스트로 바꿈

- <> 태그 제거
- 해석

Mapping

특정 단어를
지정한 단어로 치환

-> 특수 문자 처리를 위해
반드시 필요한 필터

Pattern Replace

정규식을 이용해
복잡한 패턴을 치환

Analyzer

Char Filters

Tokenizer

Token Filters

텍스트를 받아 토큰(개별 단어)로 텍스트를 분해

Standard

공백으로 term을 분리

* Term의 끝에 있는
특수문자는 제거

Letter

알파벳을 제외한
모든 공백, 숫자, 기호를
기준으로 term을 분리

Whitespace

공백으로 term을 분리

* Term의 끝에 있는
특수문자는 제거하지 않음

Analyzer

Char Filters

Tokenizer

Token Filters

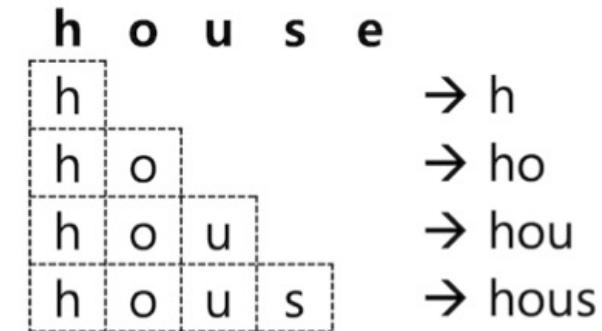
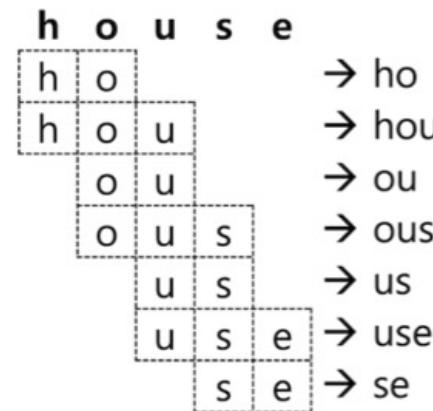
분리된 각각의 텀을 지정한 규칙에 따라 처리

Ngram

Edge Ngram

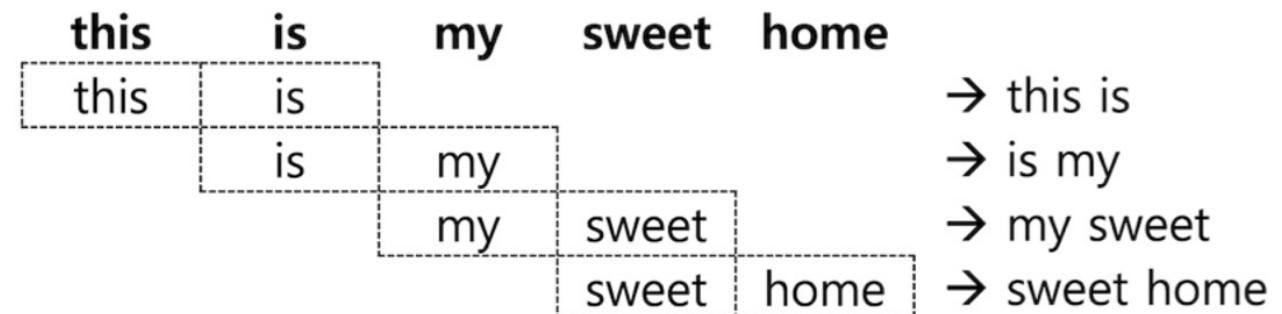
글자 단위로 구성된 묶음에서

N개의 연속된 요소를 추출



Shingle

Term 단위로 구성된 묶음에서
N개의 연속된 요소를 추출



Text Based Search

Query 쿼리

데이터베이스나 데이터 Repository 시스템에서
데이터나 정보를 요청하는 것

TF-IDF 단어 빈도와 역빈도를 기반으로 각 단어의 중요도를 평가합니다.

BM25 문서 길이와 같은 추가 요소를 사용해 TF-IDF 를 보완한 알고리즘 입니다.

TF-IDF 단어의 빈도와 역빈도 정도에 따라 단어의 중요도를 평가

$$\text{tfidf}(t, d, D) = \boxed{\text{빈도}} \times \boxed{\text{역빈도}}$$
$$\text{tfidf}(t, d, D) = \boxed{\text{tf}(t, d)} \times \boxed{\text{idf}(t, D)}$$

TF-IDF 단어의 빈도와 역빈도 정도에 따라 단어의 중요도를 평가

$$\text{tfidf}(t, d, D) = \boxed{\text{tf}(t, d)} \times \boxed{\text{idf}(t, D)}$$

$$\text{TF}(t, d) = \frac{\text{문서 } d \text{ 에서 단어 } t \text{ 가 등장한 횟수}}{\text{문서 } d \text{ 에 등장한 모든 단어의 수}}$$

특정 문서 d 에서 특정 단어 t 의 등장 횟수의 비율

$$\text{idf}(t, D) = \ln \left(\frac{D}{1 + df(t)} \right)$$

TF-IDF 단어의 빈도와 역빈도 정도에 따라 단어의 중요도를 평가

$$\text{tfidf}(t, d, D) = \boxed{\text{tf}(t, d)} \times \boxed{\text{idf}(t, D)}$$

$$\text{TF}(t, d) = \frac{\text{문서 } d \text{ 에서 단어 } t \text{ 가 등장한 횟수}}{\text{문서 } d \text{ 에 등장한 모든 단어의 수}}$$

$$\text{idf}(t, D) = \ln \left(\frac{D}{1 + df(t)} \right)$$

D : 총 문서의 개수

df(t) : 특정 단어 t가 등장한 문서의 개수

BM25 빈도/역빈도/문서의 길이에 따라 가중치를 부여

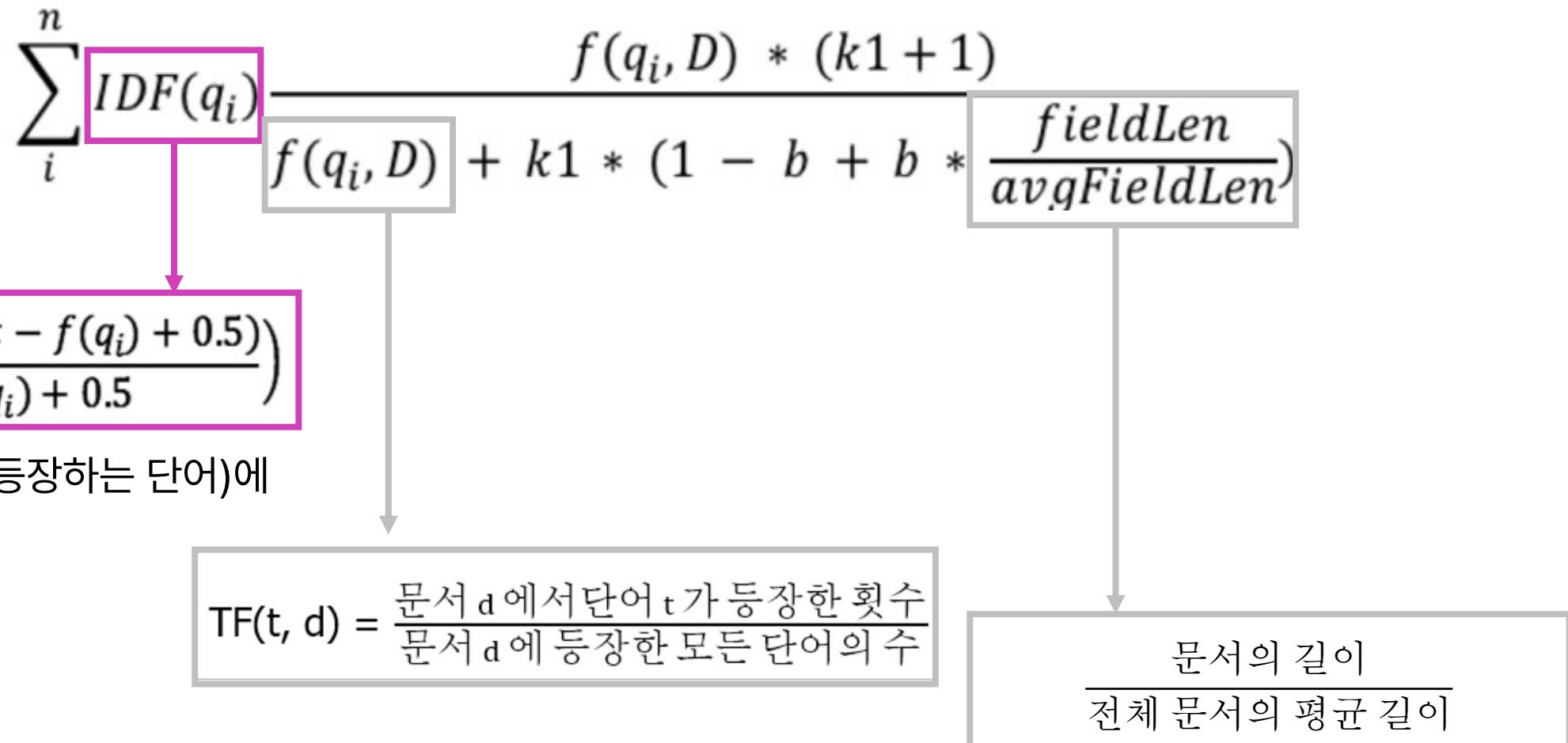
$$\sum_i^n \frac{IDF(q_i) * f(q_i, D) * (k1 + 1)}{f(q_i, D) + k1 * (1 - b + b * \frac{fieldLen}{avgFieldLen})}$$

역빈도

빈도

문서의 길이

BM25 빈도/역빈도/문서의 길이에 따라 가중치를 부여

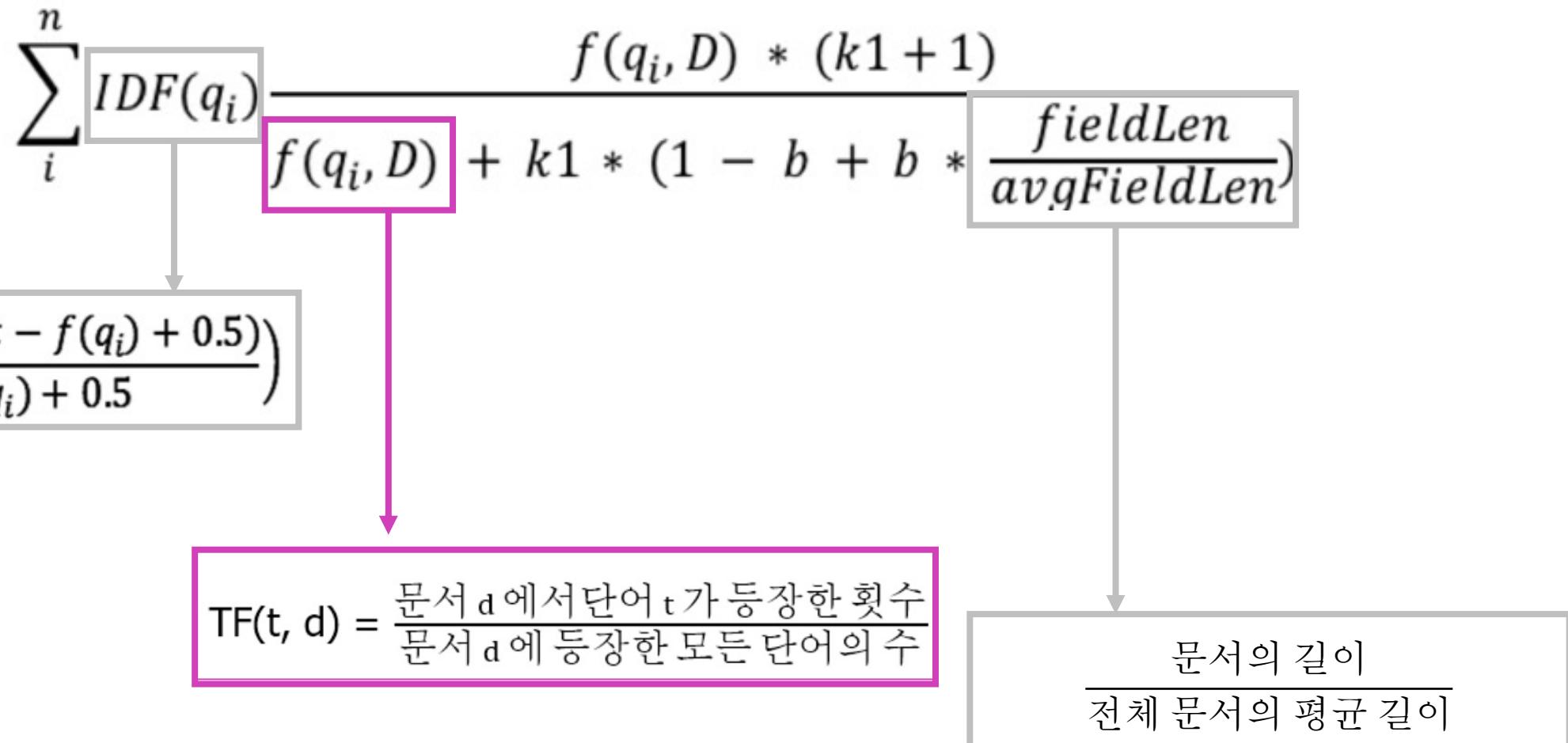


일반적인 단어(자주 등장하는 단어)에
페널티 적용

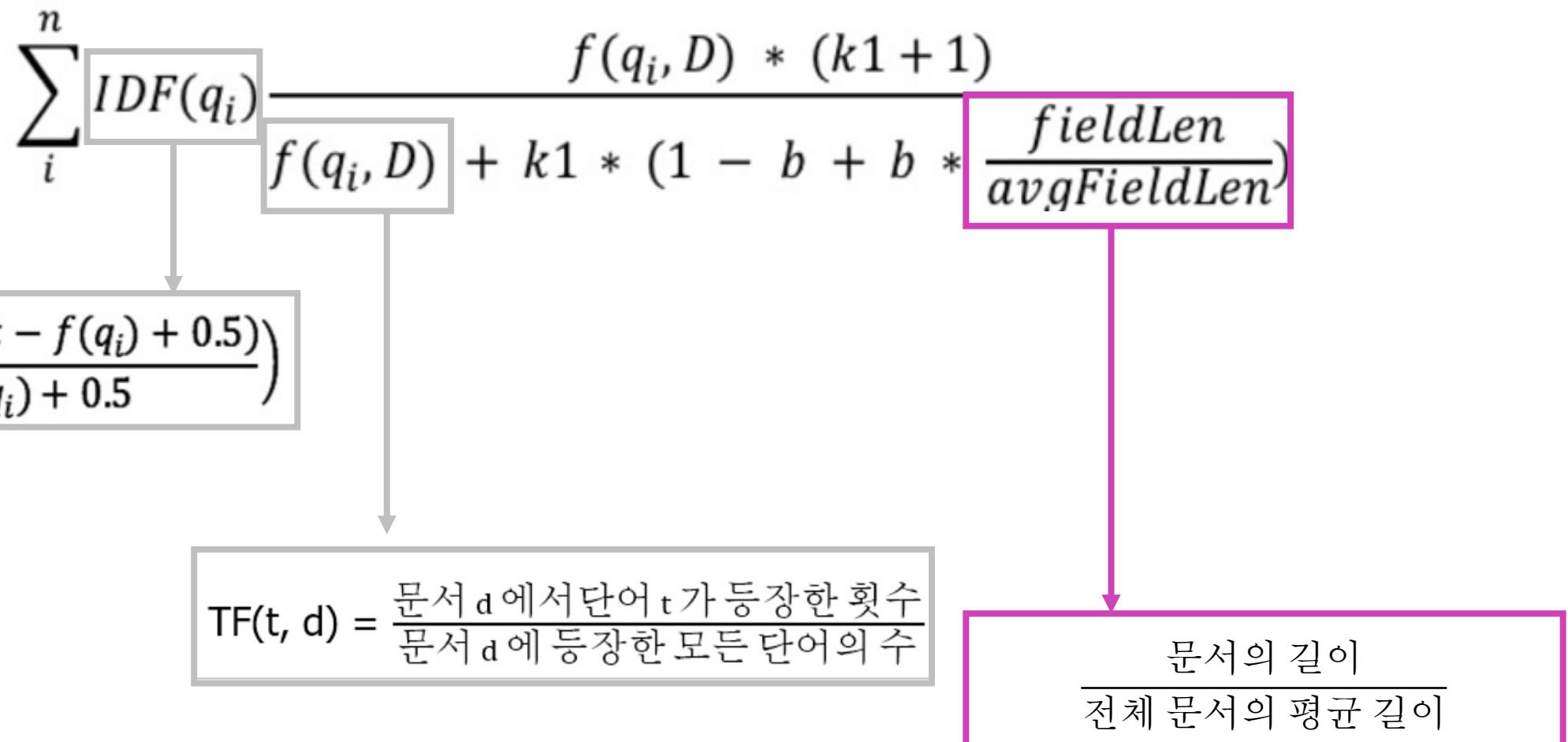
$$TF(t, d) = \frac{\text{문서 } d \text{에서 단어 } t \text{가 등장한 횟수}}{\text{문서 } d \text{에 등장한 모든 단어의 수}}$$

$$\frac{\text{문서의 길이}}{\text{전체 문서의 평균 길이}}$$

BM25 빈도/역빈도/문서의 길이에 따라 가중치를 부여



BM25 빈도/역빈도/문서의 길이에 따라 가중치를 부여



문서가 평균보다 길면 점수가 감소,
문서가 평균보다 짧으면 점수가 증가

Vector Based Search

Vector 기반 검색

목차

A. Text 기반 검색과의 차이점

1) Embedding

B. 거리를 측정하는 계산식

1) Cosine Similarity

2) Dot Product Similarity

3) Euclidean Distance

C. ANN

1) LSH

2) KD Trees

3) Annoy

I N D E X

Vector Based Search

Vector 기반 검색

TEXT 기반 검색의 개념

- 키워드 매칭에 의존합니다.
- 쿼리와 일치하는 키워드나 구문을 검색 결과로 반환합니다.

TEXT 기반 검색 :: 한계점

- 텍스트의 문맥이나 의미를 반영할 수 없습니다.
- 문서의 내용과 검색 의도 간의 차이가 있을 수 있습니다.

Text 기반 검색과의 차이점

VECTOR 기반 검색의 개념

- 벡터 기반 검색은 텍스트나 이미지를 고차원 벡터로 변환하여 의미적 유사성을 평가하는 방식입니다.
- 문맥과 의미를 더 잘 반영한 검색이 가능합니다.
- 자연어 처리(NLP), 이미지 검색, 추천 시스템 등 다양한 응용 분야에서 활용됩니다.

Vector Based Search

Vector 기반 검색

Text 기반 검색과의 차이점

VECTOR 기반 검색의 개념

- 벡터 기반 검색은 텍스트나 이미지를 고차원 벡터로 변환하여 의미적 유사성을 평가하는 방식입니다.
- 문맥과 의미를 더 잘 반영한 검색이 가능합니다.

	Text 기반 검색	Vector 기반 검색
방식	키워드 매칭	벡터 거리 측정
유의어	유의어, 동의어 사전	임베딩 모델 활용

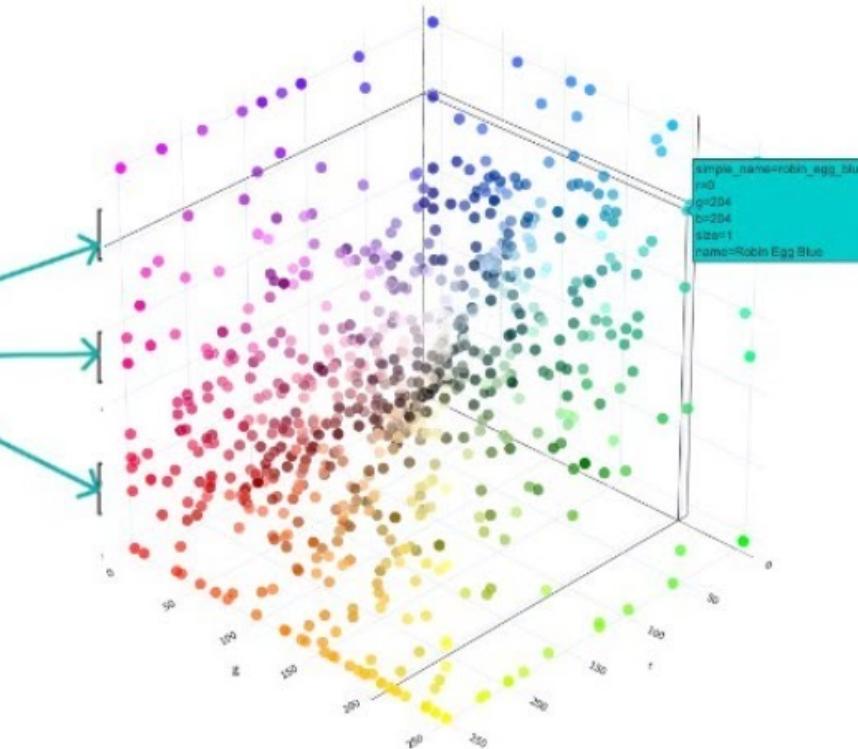
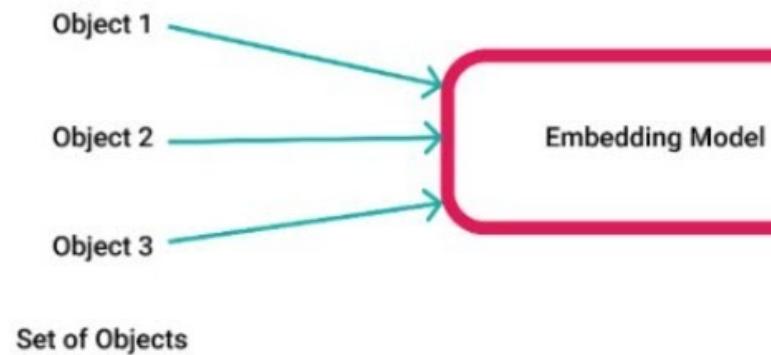
Vector Based Search

Vector 기반 검색

Embedding

Embedding

단어와 문장을 고차원 벡터 공간에 매핑하여 의미를 반영한 벡터를 생성합니다.



Vector Based Search

Vector 기반 검색

거리를 측정하는 계산식

코사인 유사성

Cosine Similarity

두 벡터 간의 각도를 측정

텍스트 임베딩에서
두 문장의 의미적 유사성을 비교

점 곱 유사성

Dot Product Similarity

크기와 방향에 따라 유사성을 측정

추천 시스템에서
사용자가 선호하는 항목과
유사한 항목 찾기

유클리드 거리

Euclidean Distance

두 벡터 간의 직선 거리를 측정

이미지 검색에서
피처 벡터 간의
시각적 유사성을 평가

각 Vector 간의 유사도를 정확하게 계산할수록,
반대로 계산에 걸리는 시간이 긴 단점도 있습니다.

Vector Based Search

Vector 기반 검색

Approximate Nearest Neighbor

정확한 거리 측정의 문제점

고차원 벡터 공간에서는 모든 데이터 포인트 간의 거리를 계산하는 것이 비효율적입니다.
연산 시간이 오래 걸리고, 대규모 데이터셋에서는 현실적으로 불가능할 수 있습니다.

“ ”
근사 최근접 이웃(ANN) 검색은 정확성을 일부 포기하는 대신,
빠르고 효율적으로 유사한 데이터를 찾는 방법입니다.

Vector Based Search

Vector 기반 검색

Approximate Nearest Neighbor

1 지역성 기반 해싱 (LSH)

유사한 벡터들이 동일한 해시 버킷에 맵핑되도록 설계된 해싱 기법입니다.

2 KD 트리

차원별로 데이터를 분할하여 트리 구조로 저장하는 방법입니다.

저차원 데이터에서 효율적이며, 트리를 탐색하여 근사 최근접 이웃을 찾습니다.

3 Annoy

여러 개의 랜덤 KD 트리를 생성하여, 근사 최근접 이웃을 효율적으로 찾는 방법입니다.

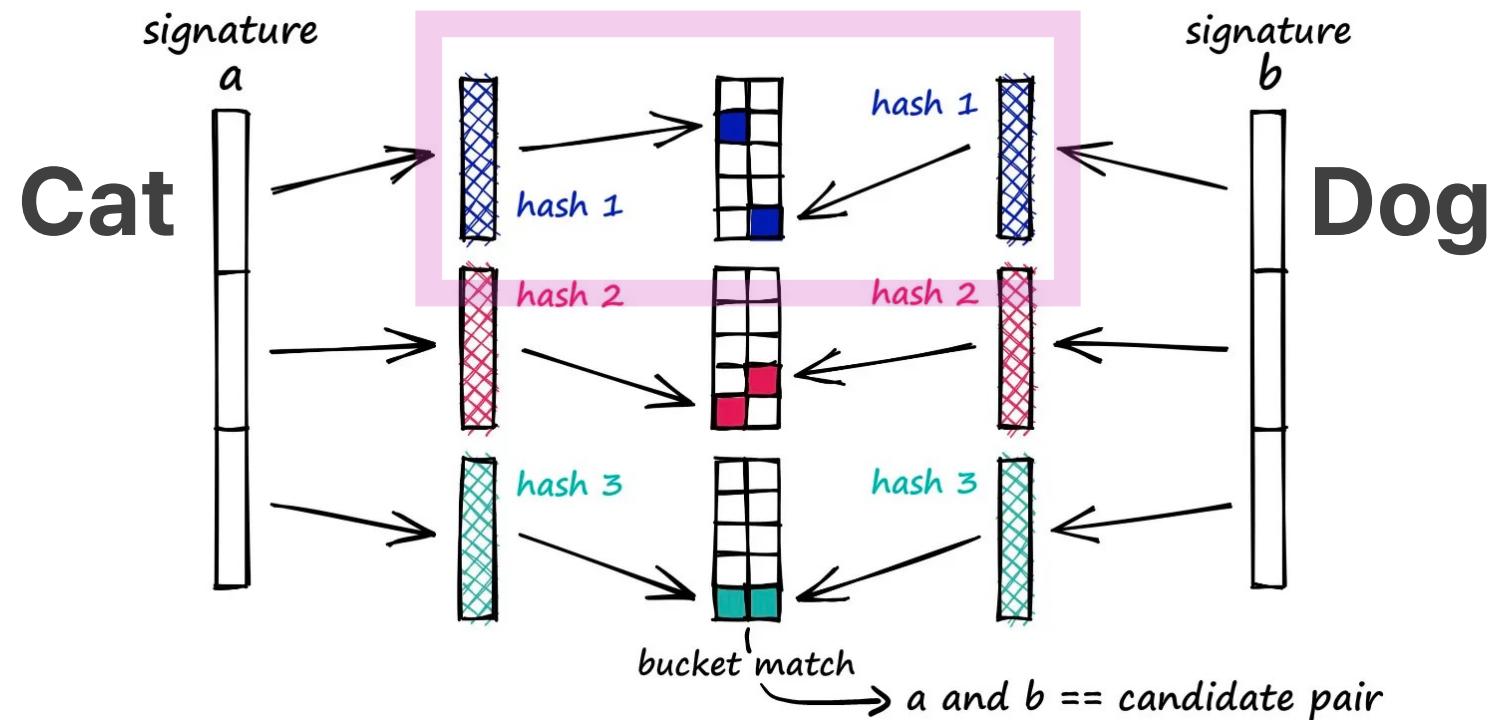
트리 탐색 결과를 결합하여 유사한 데이터를 빠르게 검색할 수 있습니다.

Approximate Nearest Neighbor

1

지역성 기반 해싱 (LSH)

유사한 벡터들이 동일한 해시 버킷에 매핑되도록 설계된 해싱 기법입니다.

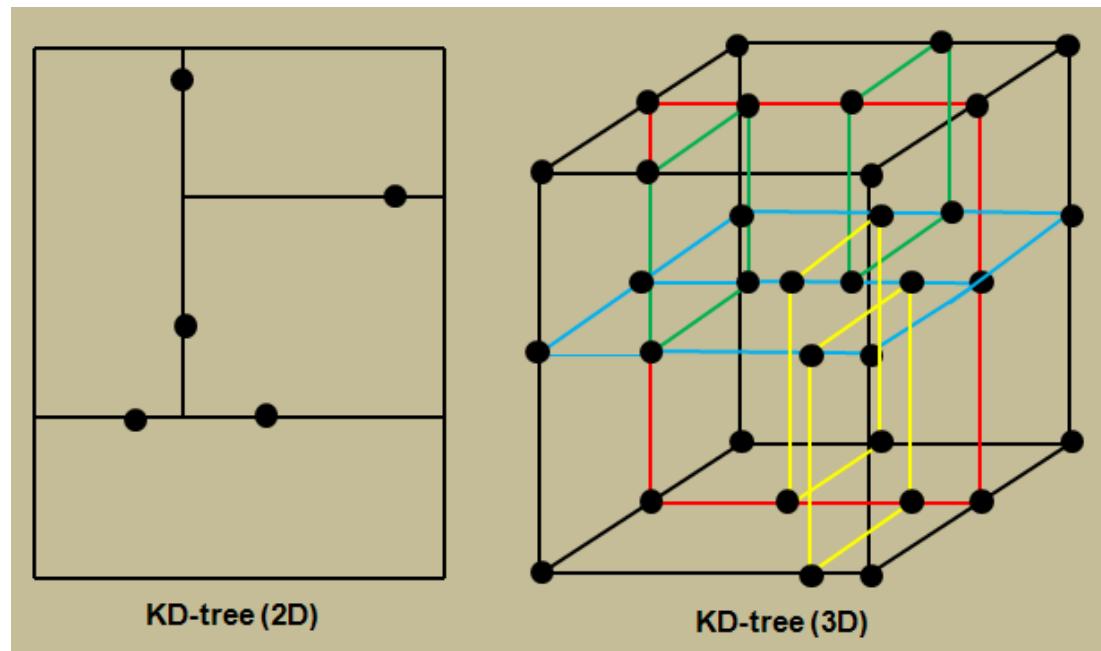


Approximate Nearest Neighbor

2 KD 트리

차원별로 데이터를 분할하여 트리 구조로 저장하는 방법입니다.

저차원 데이터에서 효율적이며, 트리를 탐색하여 근사 최근접 이웃을 찾습니다.

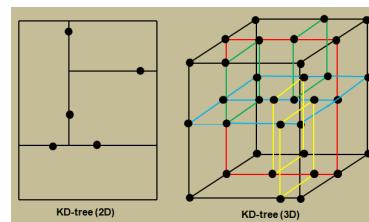
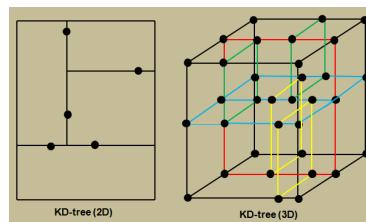
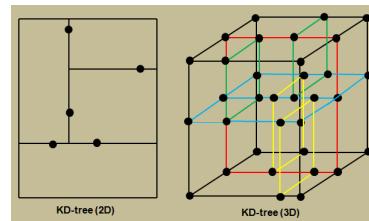
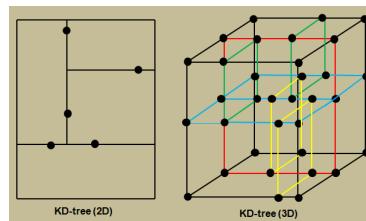


KD 트리는 차원이 증가할수록
성능이 급격히 저하되기 때문에,
고차원 데이터에는 적합하지 않을 수 있습니다.

Approximate Nearest Neighbor

3 Annoy

여러 개의 랜덤 KD 트리를 생성하여, 근사 최근점 이웃을 효율적으로 찾는 방법입니다.
트리 탐색 결과를 결합하여 유사한 데이터를 빠르게 검색할 수 있습니다.



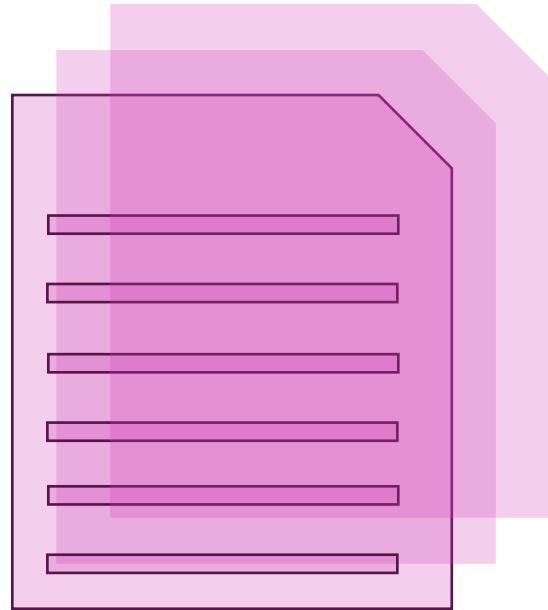
여러 개의 랜덤 KD 트리를 생성하여,
근사 최근점 이웃을 효율적으로 찾는 방법입니다.

트리 탐색 결과를 결합하여 빠르게 검색할 수 있습니다.

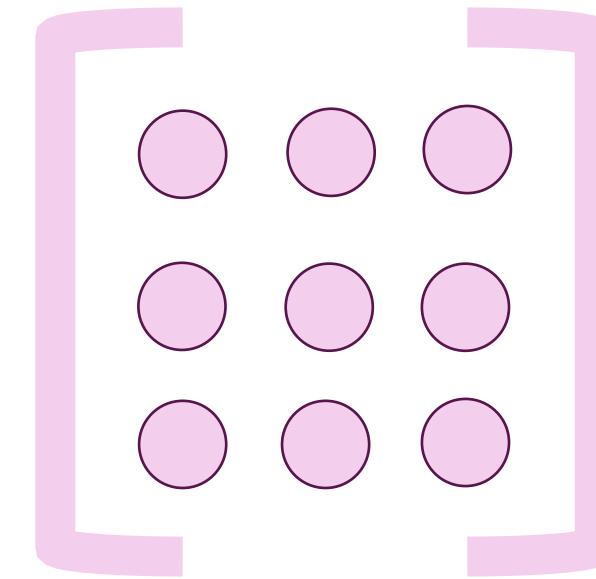
Hybrid Based Search

Hybrid 검색

TEXT 기반 검색과 Vector 기반 검색 결과를 조합



TEXT

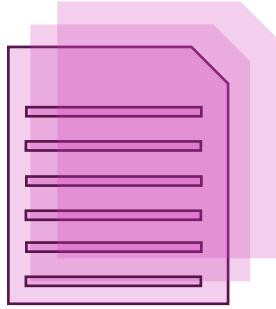


Vector

Hybrid Based Search

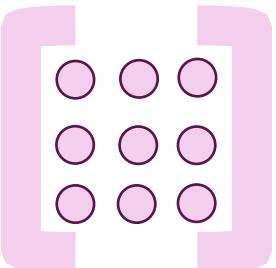
Hybrid 검색

TEXT 기반 검색과 Vector 기반 검색 결과를 조합



TEXT

- 빠른 결과 탐색
- 동의어, 유의어 사전을 통한 토큰화



Vector

- 문맥을 반영한 탐색
- 고차원 벡터 정보로 변환

Scoring

전통적인 텍스트 기반 검색 결과와
Vector 기반의 유사도 계산 결과를
점수로 수치화 합니다.

수치화 된 점수를 정규화하고
조합해 가장 잘 맞는 방식으로
조정 가능합니다.

Hybrid Based Search

Hybrid 검색

RRF (Reciprocal rank fusion)



각 쿼리는 순위가 매겨진 결과 집합을 생성하며

RRF는 순위를 쿼리 응답에서



반환된 단일 결과 집합으로 병합하고 균질화하는 데 사용

1. 각 검색 결과에 가중치를 부여한다
2. 중요도가 높아진 결과에 더 높은 점수를 안배하고,
3. 그 조합을 통해 가장 유사한 결과를 제공한다.

Hybrid Based Search

Hybrid 검색

CC (Convex Combination)

/// 어휘 검색 결과와 의미 검색 결과의 정규화된 점수를 각각의 가중치와 함께 결합하는 것을 목표 ///

어휘 점수와 의미 점수의 가중 평균

Questions Answers



REFERENCE 출처

- Deep Learning Bible - Natural Language Processing
- The Power of Embeddings in Machine Learning - Rian Dolphin
- Faiss: The Missing Manual - James Briggs
- Voxel-based volume modelling of individual trees using terrestrial laser scanners - John C. Trinder
- what-is query-language - elastic.co

REFERENCE 출처

- Deep Learning Bible - Natural Language Processing
- Understanding retrieval augmented generation (RAG)
Elastic Snackable Series
- What is retrieval augmented generation – elastic.co
- Neural networks Series - 3Blue1Brown
- Superb_AI Vector Store Explained - Superb_AI