Vertex AI Matching Engine (Vertex AI 벡터 검색)

전체 개요 및 주요 기능

Vertex AI Matching Engine(현재 명칭: Vertex AI 벡터 검색)은 대규모 **벡터 유사도 검색**을 위한 구글 클라우드의 완전 관리형 서비스입니다cloud.google.com. 이 엔진은 사용자가 사전 생성한 **임베딩 벡터**들을 색인(index)에 저장하고, 입력 쿼리 벡터와 가장 유사한 항목들을 매우 낮은 지연으로 찾아줍니다. **ScaNN**(Scalable Nearest Neighbor) 등 Google Research의 최첨단 알고리즘을 기반으로 구축되어 있어 **수십억 개** 벡터로 구성된 데이터베이스에서도 밀리초 수준의 검색 지연을 달성합니다cloud.google.com. 예를 들어 **8백만 개**에 달하는 임베딩이 있는 색인에서도 수십 밀리초 내에 의미적으로 유사한 결과를 찾아낼 수 있습니다yeoshim.tistory.com. 이러한 고성능 벡터 검색을 통해 **차세대 검색 엔진**이나 **추천 시스템**뿐만 아니라 생성형 AI 응용까지 구축할 수 있으며 cloud.google.comyeoshim.tistory.com, 복잡한 인프라 운영 없이 **완전 관리형 서비스**의 이점을 누릴 수 있습니다yeoshim.tistory.com.

주요 특징: Matching Engine은 유클리드 거리(L2), 맨해튼 거리(L1), **내적(dot product)**, 코사인 거리 등 다양한 유사도 지표를 지원하며 기본값은 내적입니다cloud.google.com. 특히 코사인 유사도의 경우 벡터를 단위 L2 노름 정규화하고 내적 거리로 계산하는 방식을 권장하는데, 이 방법이 코사인과 **동일한 순위** 결과를 내면서도 엔진의 최적화된 내적 연산을 활용할 수 있기 때문입니다cloud.google.com. 또한 **Approximate Nearest Neighbor (ANN)** 기법을 사용하여 대량의 벡터를 효율적으로 검색합니다. 기본 알고리즘으로는 **Tree-AH**(Shallow Tree + Asymmetric Hashing) 방법을 사용하여 정확도(재현율)와 지연 시간 사이 균형을 맞추는데cloud.google.com, 필요에 따라 **브루트포스(Brute-force)** 방식의 색인도 선택 가능합니다. 브루트포스는 검색 정확도 100%를 보장하지만 지연 시간이 크게 늘어나므로 **프로덕션 환경에는 권장되지 않으며**, 주로 오프라인 평가용으로 사용됩니다cloud.google.com. 서비스는 **자동 확장(Auto-scaling)**을 지원하여 검색 요청량(QPS)에 따라 백엔드 노드 수를 자동 조절하고, 일관된 저지연 검색 성능을 유지합니다cloud.google.com.

*그림 1: Vertex AI Matching Engine을 활용한 벡터 검색 파이프라인 개요. 우측에 임베딩 **색인 생성** 단계, 좌측에 **검색 쿼리 처리** 단계가 묘사되어 있습니다. 개발자는 임베딩 벡터들을 색인에 추가하고 엔드포인트로 **유사도 검색 쿼리**를 발행하게 되며, Matching Engine이 대규모 데이터셋에서 실시간으로 가장 유사한 항목들을 찾아냅니다 yeoshim.tistory.comyeoshim.tistory.com.*

벡터 인덱스 생성 및 관리 방법

Vertex AI 벡터 검색에서 **색인(index)**은 임베딩 벡터가 저장된 하나 이상의 파일로 구

성됩니다cloud.google.com. 먼저 사용자 데이터(예: 문서, 이미지 등)에 대해 사전에 임베딩 벡터를 생성해야 하며, 이를 JSON Lines 포맷의 파일로 준비합니다. 각 행에 고유 ID와 벡터 값이 포함되며, 선택적으로 메타데이터 **속성 태그**(restricts)를 추가할 수 있습니다. 예를 들어 아래와 같이 벡터에 "class":"pet" 또는 "category":"feline" 등의 속성을 부여해 두면 추후 검색 시 특정 범주의 결과만 필터링할 수 있습니다cloud.google.com:

json

CopyEdit

{"id": "42", "embedding": [0.5, 1.0], "restricts": [{"namespace": "class", "allow": ["cat","pet"]},{"namespace": "category","allow": ["feline"]}]}

{"id": "43", "embedding": [0.6, 1.0], "restricts": [{"namespace": "class", "allow": ["dog","pet"]},{"namespace": "category","allow": ["canine"]}]}

준비된 임베딩 파일은 Cloud Storage에 업로드하고, **색인 생성** 단계에서 해당 GCS 경로를 지정합니다. 색인은 콘솔 UI, gcloud CLI 또는 **Vertex AI SDK(Python)**로 생성할 수 있습니다. 예를 들어 Python SDK를 사용하는 경우 다음과 같이 코드 한 줄로 ANN 색인을 만들 수 있습니다cloud.google.com:

python

CopyEdit

```
from google.cloud import aiplatform

aiplatform.init(project=프로젝트ID, location=지역)

index = aiplatform.MatchingEngineIndex.create_tree_ah_index(
    display_name="my-index",
    contents_delta_uri="gs://<버킷명>/embeddings/",
    dimensions=768,
    approximate_neighbors_count=100
)
```

위 코드에서는 display_name으로 색인 이름을 지정하고, contents_delta_uri에 임베딩 JSON 파일이 있는 GCS 폴더 경로를 설정합니다. 또한 벡터의 **차원 수**(dimensions)와 **근사 이웃 개수**(approximate_neighbors_count) 등을 인자로 제공합니다cloud.google.com. 주요 구성 옵션들은 다음과 같습니다:

- **contents_delta_uri** – 색인에 로드할 임베딩 파일이 저장된 **Cloud Storage 경로** cloud.google.com. 하나의 폴더 경로 아래에 다수의 JSON 파일을 둘 수도 있습니다.
- **dimensions** – 각 임베딩 벡터의 **차원 크기**(예: 768)cloud.google.com. 모든 벡터는 동일한 길이여야 합니다.
- **distance_measure_type** – 유사도 판정에 사용할 **거리 함수** 유형 (디폴트 DOT_PRODUCT_DISTANCE; 지원 유형: L2, L1, Dot, Cosine 등)cloud.google.com. 코사인의 경우 내적+정규화 조합을 권장합니다.
- **approximate_neighbors_count** – ANN 검색 시 고려할 후보 이웃 개수 cloud.google.com. 값을 크게 하면 정확도는 올라가나 검색 지연이 증가하며, 일반적으로 100 정도로 설정합니다.
- **algorithmConfig** – 색인에 사용할 **검색 알고리즘** 설정. treeAhConfig 또는 bruteForceConfig 중 하나를 지정합니다cloud.google.com. 별도 설정이 없으면 기본적으로 Tree-AH 근사 알고리즘이 적용됩니다.
    - *Tree-AH 세부 설정:* leafNodeEmbeddingCount는 하나의 리프 노드에 저장될 벡터 수(기본 1000)이며, fractionLeafNodesToSearch는 쿼리당 탐색할 리프 노드의 비율(0~1 사이, 기본값 0.05)입니다cloud.google.com. 이 값들을 조정해 검색 정확도와 속도 간 트레이드오프를 튜닝할 수 있습니다.
    - *Brute-force:* {} 빈 객체로 설정하면 선형 검색을 사용합니다. 정확도 100%이지만 **고비용**이므로 대규모 실시간 서비스에는 부적합합니다 cloud.google.com.
- **index_update_method** – 색인의 업데이트 방식을 지정합니다. **BATCH_UPDATE**는 대량의 벡터를 **일괄** 추가/갱신하는 모드를 의미하며, **STREAM_UPDATE**는 실시간 **스트리밍**으로 개별 벡터 추가/삭제가 가능한 모드입니다cloud.google.com. 일반적으로 대용량 데이터셋에는 Batch 모드를 사용하고, 자주 갱신이 필요한 서비스(예: 실시간 피드)는 Stream 모드를 사용합니다.

색인 생성 작업은 데이터 크기에 따라 수 분에서 수 시간까지 소요될 수 있습니다 cloud.google.com. 색인이 성공적으로 만들어지면 **색인 엔드포인트(Index Endpoint)**를 생성하고 이를 통해 색인을 **배포(deploy)**해야 비로소 쿼리 요청을 처리할 수 있습니다cloud.google.com. 하나의 엔드포인트에 여러 색인을 탑재할 수도 있으며, 엔드포인트는 자동으로 백엔드 VM 노드를 프로비저닝하여 쿼리 트래픽을 처리합니다. 초기 배포 시에는 백엔드 인프라 준비를 위해 최대 20~30분 정도 소요될 수 있습니다

cloud.google.com. 배포 완료 후에는 Vertex AI 콘솔의 **색인 엔드포인트 대시보드**에서 상태를 모니터링할 수 있습니다.

색인 관리 측면에서, **일괄 업데이트(Batch)** 모드 색인은 새로운 데이터가 생기면 임베딩 파일을 갱신한 후 **색인 재생성** 또는 **부분 업데이트** 작업을 수행해야 합니다. 반면 **스트리밍(Stream)** 모드 색인은 배포 후 API를 통해 개별 벡터를 추가/삭제하는 연산을 제공하므로, 온라인 서비스에 더 적합합니다. 다만 스트리밍 모드는 현재 지원 용량에 제한이 있을 수 있고(미리보기 기능), 실시간 업데이트 편의와 성능 사이의 균형을 고려해야 합니다. **색인 삭제**를 원할 경우 콘솔이나 gcloud로 해당 색인을 **언디플로이(undeploy)** 한 후 삭제해야 합니다 (배포 중인 색인은 바로 삭제할 수 없음)cloud.google.com.

검색 API 사용 방법 및 예제

색인이 엔드포인트에 배포되었다면, **유사도 검색 API**를 통해 쿼리를 수행할 수 있습니다. Python SDK를 사용할 경우 앞서 생성한 index_endpoint 객체의 find_neighbors() 메서드를 호출하여 유사 벡터를 검색합니다. 아래는 예시 코드입니다cloud.google.com:

python

CopyEdit

```
# 임의의 쿼리 벡터 (예: ID가 "6523"인 제품 임베딩을 조회)
query_vector = product_embs["6523"]

# 유사한 이웃 10개 검색
response = index_endpoint.find_neighbors(
    deployed_index_id = "<배포한 색인ID>",
    queries = [query_vector],
    num_neighbors = 10
)

# 결과 출력
for neighbor in response[0]:
    print(f"{neighbor.id} (거리: {neighbor.distance:.4f})")
```

위 요청에서는 queries 리스트에 하나의 쿼리 벡터를 포함하고 num_neighbors=10으로 가장 가까운 10개 벡터를 조회했습니다. 응답은 각 쿼리에 대해 **가장 가까운 이웃**들의 ID와 **거리(distance)** 점수를 리스트로 반환합니다. 거리 값은 선택한 distance_measure_type에 따라 계산되며, 일반적으로 **값이 작을수록 유사도가 높은** 것을 의미합니다. 예를 들어 내적(dot product)을 사용한 경우 엔진은 내부적으로 **"-내적 값"**을 거리로 사용하므로, 내적값이 클수록 (두 벡터가 더 유사할수록) 거리 점수는 더 낮게 나타납니다cloud.google.com. 위 코드에서 neighbor.distance가 작은 순으로 정렬되어 출력되는 것이 이러한 이유입니다.

또한 Matching Engine은 한 번의 호출에 **여러 쿼리**를 배치로 검색할 수도 있으며 (queries = [vec1, vec2, ...] 형태), 고성능의 백엔드가 대량의 동시 검색을 실시간 처리할 수 있습니다. 엔드포인트 URL을 통해 **REST API**로도 호출 가능하며, REST/GRPC 프로토콜을 모두 지원합니다. 예를 들어 REST 호출 시 엔드포인트 URL의 <IndexEndpoint>:findNeighbors 메서드에 쿼리 페이로드를 POST하는 방식입니다 cloud.google.com.

검색 결과 해석: neighbor.id는 유사한 벡터의 원본 객체 ID를 나타내므로, 이를 통해 원본 데이터(예: 상품명, 문서 내용 등)를 어플리케이션에서 조회할 수 있습니다. neighbor.distance는 해당 벡터와 쿼리 벡터 간 거리/유사도 점수입니다. 앞서 설정한 거리 함수에 따라 이 값이 산출되며, 필요에 따라 응용 단계에서 점수를 **유사도 점수**로 변환해 활용할 수도 있습니다. 예를 들어 코사인 유사도의 경우 1 - cosine_distance 형태로 유사도 스코어를 계산할 수 있습니다.

결과 필터링: Matching Engine은 쿼리 시에 **벡터 메타데이터 필터링** 기능을 제공하여, 특정 조건을 만족하는 벡터들만 검색하도록 제한할 수 있습니다cloud.google.com. 예를 들어 상품 추천 시 **"카테고리가 전자제품인 아이템 중에서 유사한 상위 N개"**만 찾고 싶다면, 쿼리 요청에 해당 namespace의 **restrict 필터**를 포함시켜 결과를 제한할 수 있습니다. 복수의 조건은 AND/OR 조합으로 구성 가능하며, 숫자 범위 조건도 지원합니다 cloud.google.comcloud.google.com. 이러한 속성 필터를 활용하면 **언어**, **상품 종류**, **가격대** 등의 조건에 따른 세분화된 유사도 검색이 가능합니다.

Vertex AI Matching Engine의 백엔드 검색 서버는 **메모리 기반 벡터 DB**로 동작하며, 대량의 벡터 비교 연산을 실시간으로 처리하도록 최적화되어 있습니다. 수평 확장을 통해 높은 QPS 환경을 지원하고, 앞서 언급한 자동 확장 기능으로 부하 변화에 유연하게 대응합니다cloud.google.com. 실제 테스트에서 색인 내 벡터 개수가 **수십억 개에 달하더라도** 검색 질의(find_neighbors) 한 번이 걸리는 시간은 불과 몇 밀리초 수준으로 보고되고 있습니다cloud.google.com. 이는 대규모 데이터셋에서도 사용자가 체감하기에 **실시간**에 가까운 유사 항목 검색을 제공함을 의미합니다.

사용 사례

*그림 2: **추천 시스템** 시나리오에서 Vertex AI 벡터 검색의 활용 개념도. 벡터로 변환된 제품, 콘텐츠, 사용자 프로필 등을 벡터 DB에 저장해 두고 유사한 항목을 실시간으로 찾아 추천에 활용할 수 있다.*

벡터 검색 기술은 **AI 시대의 데이터 검색 허브**로 부상하고 있으며, 문서, 이미지, 제품, 사용자, 이벤트 등 다양한 비즈니스 요소들을 **유사성**이라는 기준으로 연결해줍니다 cloud.google.com. 단순한 키워드 매칭을 넘어서 **의미 기반**으로 데이터를 연결함으로써, 기업 애플리케이션에 새로운 **지능형 기능**을 제공할 수 있습니다yeoshim.tistory.com. 아래는 Vertex AI Matching Engine의 대표적인 활용 분야입니다:

- **추천 시스템:** 사용자의 행동이나 관심사를 반영한 **개인화 추천**에 활용됩니다. 예를 들어 일본의 중고 거래 플랫폼 Mercari는 Matching Engine을 도입하여 사용자 **관심사와 판매자 재고를 벡터로 표현**하고, 실시간으로 유사도가 높은 상품을 찾아 추천함으로써 쇼핑 경험을 향상시켰습니다fastercapital.com. 이처럼 제품 추천, 콘텐츠 추천, 친구 추천 등 다양한 추천시스템에서 대규모 벡터 유사도 검색을 통해 높은 품질의 결과를 실시간 제공할 수 있습니다.

- **유사 문서 검색:** 문서나 텍스트 데이터를 임베딩해 두고 **의미적으로 유사한 문서**를 찾는 검색에 활용됩니다. 예를 들어 Stack Overflow의 800만 개에 달하는 Q&A 글을 미리 임베딩한 후, 사용자의 질문과 가까운 의미의 질문들을 Matching Engine으로 조회하면 **밀리초 단위**로 관련 Q&A를 찾아낼 수 있습니다 yeoshim.tistory.com. 전통적 키워드 검색으로는 찾기 어려운 의미적 유사 문서, 뉴스 기사, 보고서 등을 빠르게 탐색하여 **검색 정확도**와 **이용자 만족도**를 크게 높일 수 있습니다.

- **이미지 유사도 비교:** 이미지 데이터를 벡터로 표현한 후 **시각적 유사성**에 따라 검색하거나 군집화하는 데 활용할 수 있습니다. 예를 들어 전자상거래에서 특정 상품 이미지와 비슷한 스타일의 상품을 찾거나, 사진 아카이브에서 겹치는 장면이나 중복 이미지를 식별하는 작업에 응용 가능합니다. Vertex AI의 벡터 검색은 **텍스트 데이터와 이미지 데이터를 모두 처리**하여 멀티모달 검색을 구현할 수도 있습니다cloud.google.com. 즉, 하나의 쿼리에 이미지 임베딩과 텍스트 임베딩을 함께 활용해, 이미지와 설명이 모두 유사한 항목을 찾는 **복합 검색**도 지원합니다.

- **지식 검색 및 QA**: 사내 문서나 위키, FAQ 등을 임베딩하여 **AI 챗봇**이나 QA 시스템의 지식 베이스로 활용할 수 있습니다. 예를 들어 기업은 대량의 문서를 벡터 색인에 저장해 두고, 사용자의 질문이 들어오면 Matching Engine으로 관련 문서를 조회하여 **대규모 언어모델(LLM)**의 답변 생성을 보조할 수 있습니다. 실제

로 임베딩 생성 API와 Matching Engine을 결합하면 LLM이 정확한 근거에 기반한 응답을 하도록 하는 **Grounding** 기법을 손쉽게 구현할 수 있으며, 이를 통해 챗봇의 **환각(hallucination)** 문제를 줄이고 신뢰성을 높일 수 있습니다 yeoshim.tistory.com.

이 밖에도 **음악/동영상 추천**, **이상징후 탐지**, **유사 사용자 군집화**, **특허 문헌 유사성 검색** 등 **무엇을 벡터화하느냐**에 따라 다양한 산업 분야에 응용할 수 있습니다. Vertex AI Matching Engine은 이러한 벡터 유사도 검색 기능을 **확장성**, **신속성**, **관리 편의성**을 갖춘 형태로 제공함으로써, 개발자가 **최신 AI 응용**을 빠르게 구축하는 것을 돕고 있습니다 yeoshim.tistory.comyeoshim.tistory.com.

자료 출처: 본 답변은 Google Cloud 공식 문서와 블로그 자료 cloud.google.comcloud.google.com를 기반으로 정리되었습니다.