BBDD de Vectores

Ampliación de Bases de Datos

Stanislav Vakaruk

U-Tad

9 de diciembre de 2025

Agenda

1. Conceptos básicos
2. Métricas de similitud
3. Generación de embeddings
4. Modelo de datos
5. Estrategias de búsqueda
6. Evaluación

Agenda

¿Qué es un *embedding*?

Definición

Un embedding es una representación vectorial densa que captura el significado de un objeto (texto, imagen, audio, código) en un espacio métrico.

- **Objetivo**: objetos semánticamente similares: vectores cercanos.
- **Dimensionalidad típica**: 128, 384, 768, 1536 (según el modelo).
- **Analogía**: Piensa en un mapa donde cada ciudad es una palabra; ciudades similares están cerca.

Intuición visual

Cada objeto se proyecta a un punto en un espacio de altas dimensiones donde las distancias codifican similitud.

Propiedades clave

- **Densos**: todos los componentes son reales (sin dispersión).
- **Normalización**: frecuente L2 para fijar $\|x\| = 1$. Evita sesgos por magnitud y permite usar coseno $\approx$ producto interno.
- **Espacio métrico**: la similitud se mide con métricas específicas.

Ejemplo práctico

Frases similares como “El perro juega en el parque” y “Un can corre en el jardín” tendrán embeddings cercanos.

Buenas prácticas

Normalizar si se usa coseno/IP y mantener *consistencia* de modelo y *pipeline* en todo el índice.

Agenda

Similitud del coseno

Fórmula

$$\cos\theta = \frac{x \cdot y}{\|x\|\,\|y\|}$$

- Ideal con vectores normalizados ($\|x\| = \|y\| = 1$).
- Captura **dirección** (significado) e ignora **magnitud**.

Ejemplo

Dos vectores con ángulo pequeño tienen alta similitud.

Producto interno y distancia L2

Producto interno (dot)

- Útil si el modelo fue entrenado para maximizarlo.
- En vectores normalizados, se alinea con coseno.

Distancia Euclídea (L2)

- Menos común en texto.
- Más usada en imagen/audio cuando la magnitud aporta información.

Regla práctica

Imagen/Audio : coseno *o* L2 según el modelo.

Agenda

Cómo se generan embeddings

APIs (OpenAI, Cohere)

- Ventajas: alta calidad, facilidad de uso.
- Inconvenientes: coste, privacidad.

Open-source (`sentence-transformers`)

- Ventajas: control, ajuste al dominio (*fine-tuning*).
- Inconvenientes: infraestructura, mantenimiento.

Buenas prácticas

- Normalizar si se usa coseno/IP.
- Consistencia: mismo modelo y pipeline para todo el índice.
- Preprocesado: limpieza y *chunking* (200–400 palabras con solape).
- Versionado: guardar modelo y parámetros en metadatos.

Agenda

Modelo de datos en una BBDD de vectores

Antes de buscar por similitud

El vector es el núcleo, pero rara vez es suficiente: casi siempre añadimos **metadatos** (payload) para describir el objeto, limitar el espacio de búsqueda con filtros y combinar señales semánticas con señales léxicas.

Estructura típica de un registro (5 piezas)

1. **Identificador estable**: `doc_id` o `chunk_id` (si dividimos documentos largos).
2. **Vectores con nombre y cuerpo** y su *métrica* (coseno, L2 o producto interno).
3. **Metadatos** descriptivos: tipo, idioma, categorías, fechas, autor, etiquetas, permisos, geolocalización... usados luego para filtrar.
4. **Trazabilidad mínima**: modelo de embeddings, versión, fecha de generación y estrategia de *chunking*.
5. **Vínculo al origen**: URL o ruta para reconstruir el resultado o aplicar *re-ranking* fuera del índice.

Diseño de payload y filtros efectivos

Cómo filtrar bien

Los filtros deben **reducir de verdad** el conjunto candidato. Funcionan especialmente bien: *idioma*, *categoría*, *estado* o *ventana temporal*.
Aportan poco los campos de **cardinalidad extrema** (p.ej. el `id` único de cada registro).

- Si el motor lo soporta, **indexar** los campos de filtro frecuentes evita *post-filtrado* costoso sobre resultados provisionales.
- En colecciones grandes, **particionar** por un metadato muy usado (p.ej. *multicliente*, *idiom* o *año*) acota aún más el trabajo por consulta.

Reglas prácticas

- Define el *esquema* del payload antes de indexar.
- Prioriza filtros con **alta selectividad** y **uso frecuente**.
- Mantén **consistencia** de modelo y **versionado** en la trazabilidad.

Agenda

FLAT vs. ANN: idea general

Dos estrategias de búsqueda

Búsqueda exacta (FLAT): compara la consulta contra *todos* los vectores y garantiza el **top-k verdadero**.
Búsqueda aproximada (ANN): recorre sólo una parte *bien elegida* del espacio para reducir **latencia** y **coste**, aceptando un **pequeño error** controlable.

FLAT (brute-force)

- Coste $\sim$ **lineal** en nº de elementos y en la **dimensión**.
- Devuelve **top-k exacto**.
- Funciona muy bien con **filtros selectivos** por metadatos que reducen el candidato.
- Común como **segunda etapa**: reordenar candidatos con distancia exacta.

ANN (aproximada)

- Usa **índices especializados** (grafos, particiones en listas/centroides).
- Compromiso latencia $\leftrightarrow$ **recall**.
- Requiere **memoria** para el índice y **tiempo** de construcción.
- Natural cuando hay **millones** de vectores.

Cuándo elegir FLAT vs. ANN

Elige FLAT cuando...

- La colección es **pequeña/mediana** o está **pre-filtrada** con alta selectividad.
- Necesitas **exactitud total** (top-k verdadero) o **ground truth** para evaluar.
- Implementas un **re-ranking** exacto sobre un conjunto candidato reducido.

Elige ANN cuando...

- Escalas a **millones+** de vectores con **baja latencia** y **alto QPS**.
- Aceptas un **pequeño error** y optimizas **latencia/coste**.
- Puedes **tunear** el índice para cumplir Recall@k y P95 de latencia.

Agenda

Evaluación de embeddings: visión general

La calidad de los embeddings determina la utilidad del sistema.

- **Intrínseca**: analiza el espacio vectorial sin tarea concreta.
- **Extrínseca**: impacto en tareas reales (búsqueda, RAG, recomendación).

Evaluación intrínseca

- **Separabilidad**: vectores de clases distintas bien separados.
- **Agrupamiento**: métricas como *Silhouette Score*.

Uso

Útil para diagnósticos rápidos y comprensión del espacio antes de integrar en una tarea.

Evaluación extrínseca y métricas de ranking

- **Recall@k**: $\frac{\#\,\text{relevantes recuperados en top-}k}{\#\,\text{relevantes totales}}$. Prioritario en búsqueda semántica.
- **Precision@k**: $\frac{\#\,\text{relevantes en top-}k}{k}$. Valora la pureza del ranking.
- **MRR**: $\text{MRR} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{\text{posición del primer relevante}}$.
- **nDCG**: considera relevancia múltiple y posición, $\text{nDCG} = \frac{\text{DCG}}{\text{IDCG}}$.
- **Hit Rate**: porcentaje de consultas con al menos un resultado relevante.

Cómo medir en la práctica

- Usar **ground truth (FLAT)** como referencia.
- Comparar **ANN vs. FLAT** para Recall@k.
- Validar en un conjunto **representativo**, no solo ejemplos aislados.

Consejo

Mantén un banco de consultas y relevancias versionado; evalúa tras cualquier cambio de modelo o indexación.