Clase 152 — RAG básico y embeddings (+ hybrid search, re-ranking, MCP)

Parte: 2 — Deep Learning · Fuente: Lewis et al. (2020) RAG paper + LlamaIndex / LangChain docs + Anthropic MCP spec. Duración estimada: 100 min.

Clase 152 — RAG básico y embeddings (+ hybrid search, re-ranking, MCP)

> *Parte: 2 — Deep Learning · Fuente: Lewis et al. (2020) RAG paper + LlamaIndex / LangChain docs + Anthropic MCP spec. Duración estimada: 100 min.*

Objetivo

Construir un sistema RAG (Retrieval-Augmented Generation) — pipeline que enriquece a un LLM con conocimiento externo: documentos propios, base de datos, web. Pipeline: embedding los docs → almacenar en vector DB → al query, hacer retrieval de los k más relevantes → inyectar como contexto al LLM → respuesta basada en docs. Conocer mejoras modernas: hybrid search (denso + sparse), cross-encoder re-ranking, y el Model Context Protocol (MCP) que estandariza la conexión LLM-herramientas.

Resultados de aprendizaje

Al finalizar, el estudiante podrá:

- Generar embeddings de texto con sentence-transformers o modelos HF.
- Almacenar y buscar embeddings con FAISS, Chroma, Pinecone, Qdrant, o pgvector.
- Construir el flujo query → embed → top-k → context → LLM.
- Aplicar hybrid search: combinar BM25 (sparse) con embeddings (dense) → mejor recall.
- Aplicar cross-encoder re-ranking sobre los top-100 retornados para promover los top-10 reales.
- Reconocer el MCP como protocolo abierto para conectar LLMs a herramientas (filesystems, DBs, APIs).

Temas

- Por qué RAG: LLMs no saben de documentos privados; entrenar fine-tune no escala para conocimiento dinámico.
- Embeddings: vectores de dimensión ~768-1536.
- Vector DBs: FAISS (in-memory), Chroma (local), Qdrant/Weaviate (server), pgvector (Postgres extension).
- Chunking: dividir docs en piezas de ~200-1000 tokens.
- Top-k retrieval + context window.
- Complemento moderno: hybrid search, cross-encoder rerank, MCP.

Versión profundizada — 2026

El tema moderno que antes vivía como complemento dentro de esta clase ahora tiene su(s) clase(s) propia(s) con patrón completo, ejercicios y homework:

- Clase 129a — MCP (Model Context Protocol)
- Clase 129b — Agentes: tool use, ReAct, multi-agent
- Clase 129c — LLM Evaluation: MMLU, MT-Bench, LLM-as-judge

Definiciones y características

- Embedding model: red que mapea texto a vector denso. all-MiniLM-L6-v2 (small, free), text-embedding-3-large (OpenAI, mejor calidad).
- Vector DB: base optimizada para similarity search aproximada (HNSW, IVF).
- Chunking: dividir documentos largos. Trade-off: chunks chicos → contexto perdido; grandes → ruido.
- Top-k: cuántos docs retorna el retriever. Típico 5-20.
- Reranker: modelo que re-puntúa pares (query, doc).
- MCP: protocolo standardizado de Anthropic para tool use.
- RAG-Fusion: variante que genera N variantes del query y agrega resultados.

Dataset / recursos

- Cualquier corpus de docs propios (PDFs, markdown, HTML).
- Wikipedia dump para experimentos.
- Librerías: sentence-transformers, chromadb / faiss-cpu, rank_bm25, langchain / llama-index, mcp.

Ejercicios

1. Embed + index: con sentence-transformers/all-MiniLM-L6-v2, embeber 100 párrafos y guardarlos en Chroma. Query y top-5.
2. BM25 baseline: con rank_bm25, query y comparar resultados vs dense.
3. Hybrid (RRF): combinar ambos. Verificar que hybrid > dense > BM25 sola en queries técnicas.
4. Cross-encoder rerank: tomar top-50 del paso 3, rerankar con cross-encoder/ms-marco-MiniLM-L-6-v2. Comparar nDCG.
5. RAG con LLM: top-5 → contexto + query → Claude o GPT → respuesta con citas.

Homework verificable

Mini RAG sobre 100-1000 documentos propios:

1. Chunking + embedding + indexado en Chroma.
2. Pipeline retrieval (dense), generate (LLM via API o vLLM local).
3. 10 queries de prueba.
4. Evaluar manualmente: ¿cuántas respuestas correctas con citas válidas?
5. Bonus: agregar BM25 + cross-encoder rerank y medir mejora.

Criterio de aceptación: al menos 7/10 respuestas correctas con citas válidas; hybrid + rerank mejora el ratio.

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
Retrieval devuelve docs irrelevantes	Chunks muy grandes o embeddings malos. Fix
LLM alucina aunque hay contexto	Prompt no instruye a "responder solo con e
Recall bajo en queries con nombres propios	Dense pierde. Fix: hybrid con BM25.
Vector DB lento con millones de docs	Sin índice ANN. Fix: HNSW (default en Chro
Citas falsas (LLM inventa números)	El modelo no respeta formato. Fix: usar fu

Preguntas frecuentes

¿RAG o fine-tuning?

RAG para conocimiento dinámico, citable, separable del modelo. Fine-tune para skills (formato de respuesta, tono). A menudo ambos: fine-tune para how, RAG para what.

¿Cuál vector DB?

- Prototipo local: Chroma o FAISS.
- Producción small: Qdrant (Rust, rápido).
- Postgres existente: pgvector.
- Enterprise: Pinecone (managed, sin maintenance).

¿Embeddings open o API?

Open: bge-large-en-v1.5, Nomic-embed-text-v1.5. API: text-embedding-3-large (OpenAI), voyage-3 (Voyage AI). API es mejor pero costo recurrente.

¿Cómo evalúo el RAG?

Métricas: Recall@k (¿el doc correcto está entre top-k?), MRR, nDCG. Para end-to-end: RAGAS library, LLM-as-judge.

¿MCP en producción?

Sí, ya. Anthropic y comunidad ofrecen MCP servers para muchas tools comunes. Adopt earlier para ganar portabilidad cross-LLM.

Referencias

- Lewis et al. (2020), Retrieval-Augmented Generation, NeurIPS — paper original.
- Reimers & Gurevych (2019), Sentence-BERT, EMNLP.
- Robertson & Zaragoza (2009), BM25 and Beyond.
- Anthropic (2024), Model Context Protocol — <https://modelcontextprotocol.io/>.
- LlamaIndex docs, LangChain docs.

Siguiente clase

Clase 153 — MCP (Model Context Protocol): herramientas y datos para LLMs

Apéndice: notebook (primer bloque)

Primera celda ejecutable del notebook de la clase.

```
# Imports y configuración inicial
```

Archivos complementarios

- notebook.ipynb