DERİN ÖĞRENME DERSİ

RAG + LLM (Retrieval-Augmented Generation) PROJE REHBERİ

Aşağıdaki guideline, öğrencilerin RAG sistemlerini doğru şekilde inşa edip, embedding–retriever–LLM bileşenlerini karşılaştırarak bilimsel bir analiz yapmalarını sağlar.

Rehber; veri toplama, chunking, retriever tasarımı, embedding seçimi, LLM çalıştırma (API + lokal), performans ölçümleri ve final karşılaştırmalarını kapsar.

1. Proje Tanımı

Her grup bir bilgi tabanına yönelik bir **Soru-Cevap (QA) RAG sistemi** inşa edecektir.

ZORUNLU:

- Doküman toplama ve temizleme
- Chunking
- Vektör veritabanı kurma (FAISS vb.)
- En az **iki embedding modeli**
- En az **iki retrieval ayarı** (k değeri, benzerlik metriği, re-ranking)
- En az **dört LLM** (biri API olabilir, **diğeri ikisi mutlaka lokal**)
- **RAG + LLM performans analizi**

2. Veri ve Knowledge Base Hazırlığı

Minimum gereksinimler:

- ≥ 50 doküman (domain müsaade ettiği ölçüde)
- ≥ 100 sayfa karşılığı metin
- PDF / txt / md / web scraping serbest
- Temizleme (headers, page numbers, artifacts çıkarılacak)

3. RAG Pipeline Bileşenleri

3.1. Chunking

Gruplar en az iki farklı strateji denemeli:

- Fixed-length chunking (ör. 512–1024 token)
- Overlap (ör. 128 token)
- Paragraph-based chunking
- Section-based chunking
- Page-based chunking, ...

3.2. Embedding Modelleri (Zorunlu: ≥ 2 model)

Örnek modeller:

- BGE family (bge-small-en, bge-base-en, bge-m3)
- MiniLM-L6-v2
- E5 models
- Türkçe için: BERTurk, DBMDZ Turkish BERT

Karşılaştırılacaklar:

- embedding boyutu
- encoding time
- retrieval performance

3.3. Vektör Veritabanı

Aşağıdakilerden biri zorunlu:

- FAISS
- Chroma
- Qdrant (lokal)

Ayarlanacak parametreler:

- similarity metric (cosine / inner product)
- k (top-k retrieval count)
- index type (flat / HNSW / IVF)

3.4. Retriever

Basit retriever + mümkünse:

- Re-ranking (cross-encoder)
- Similarity threshold filtering

4. LLM Katmanı

4.1. Faz 1 — API LLM

OpenAI / Gemini / Claude / Groq olabilir.

Bu faz **kısa özet** olarak raporda yer alacak.

4.2. Faz 2 — Lokal LLM (Zorunlu)

Her öğrenci **en az dört LLM'i lokal çalıştırmak zorunda**:

Desteklenenler:

- **LLaMA 3 8B Instruct**
- **Mistral 7B Instruct**
- **Gemma 2B / 7B**

- **Qwen 7B**
- **Phi-3, vb..**

Çalıştırma platformu:

- Ollama
- llama.cpp
- text-generation-webui
- LM Studio

Karşılaştırma yapılacak:

- Yanıt kalitesi
- Gecikme (latency)
- Token/s
- Hallucination rate
- Donanım kullanımı (CPU/GPU RAM)

5. Prompting ve Genel RAG Akışı

Prompt Template:

You are an assistant that answers questions using the provided context.

CONTEXT:

{retrieved_chunks}

QUESTION:

{user_question}

INSTRUCTIONS:

- Use only the information in the context.

- If the answer is not in the context, say "I don't know based on the given documents."

- Answer concisely in Turkish.

6. Performans Değerlendirme (Zorunlu)

Öğrenciler **30–50 soruluk bir Evaluation Set** hazırlamalıdır (gold answers ile).

Değerlendirme 3 seviyede yapılır:

A. Retriever-Only Performance (RAG Retriever)

Amaç:
Retriever'ın doğru dokümanları bulup bulamadığını görmek.

Girdi:

- Soru
- Retriever → top-k chunks

Zorunlu Metrikler:

- Recall@k
- Hit Rate@k
- MRR@k (opsiyonel)

Çıktı yorumu:

- Recall@k yüksek → retriever iyi
- Recall@k düşük → RAG başarısız olur, LLM suçlu değildir 😉

B. LLM-Only Performance (No RAG)

LLM'e context vermeden doğrudan soru sorulur.

Zorunlu Metrikler:

- Exact Match (EM)
- F1-score
- Human evaluation (3 kriter):
    - Relevance
    - Faithfulness
    - Fluency
- Hallucination Rate

Amaç:
LLM'in halüsinasyon eğilimini ölçmek.

C. Full RAG System Performance (RAG + LLM)

Retriever + LLM birlikte çalışır.

Zorunlu Metrikler:

- EM

- F1
- Human evaluation (3 kriter)- Burada bu arkadaş siz oluyorsunuz o sebeple domain e aşina olmanız ya da aşina olanları darlamanız beklenir 😊
- Hallucination Rate (LLM-only ile karşılaştırılacak)

Beklenti:
RAG + LLM → en iyi sonuç
LLM-only → en çok error & hallucination

7. Zorunlu Performans Karşılaştırma Tablosu

Aşağıdaki tablo final raporda DOLDURULMAK ZORUNDADIR:

Final Comparison Table

SYSTEM	EM ↑	F1 ↑	Human Relevance ↑	Faithfulness ↑	Hallucination ↓	Latency ↓
Retriever-Only	–	–	–	–	–	✓ fastest
LLM-Only	0.xx	0.xx	1–5	1–5	HIGH	varies
RAG + LLM	**0.xx**	**0.xx**	**higher**	**highest**	**LOW**	medium

Her öğrenci bu tabloyu gerçek sonuçlarıyla dolduracak.

8. Analiz ve Sonuç Kısmında Zorunlu Tartışma

Öğrenciler şu soruları cevaplamalıdır:

1. RAG retriever başarılı mı?

- Eğer Recall@k yüksek ama RAG+LLM düşükse → sorun LLM'dedir.

2. LLM-only neden düşük performanslıdır?

- Halüsinasyon oranı yüksek mi?
- Bilgi eksikliği mi?
- Dil modeli domain'e uzak mı?

3. RAG sistemi LLM-only'e göre ne kadar iyileştirme sağlamıştır?

- Yüzdesel iyileştirme hesaplanabilir.

4. Lokal LLM ile API LLM arasındaki fark neydi?

- Kalite
- Hız
- Maliyet

5. En ideal konfigürasyon ne oldu?

- Hangi embedding + hangi retriever + hangi LLM en iyisi?

9. İngilizce Şemalar (Pipeline Diagrams)

Figure 1 — Overall RAG Pipeline

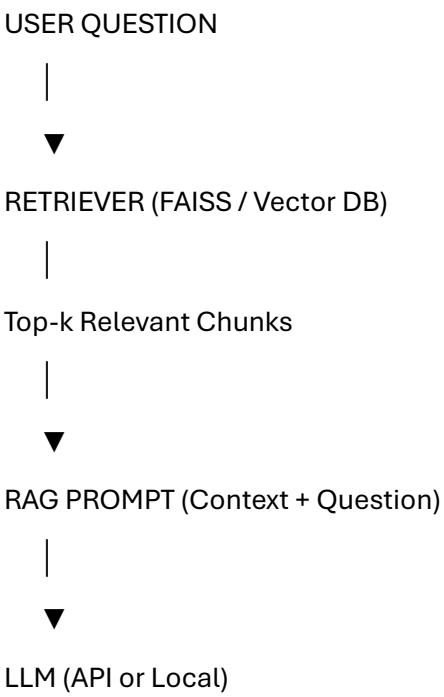


Figure 2 — Retriever Evaluation Flow

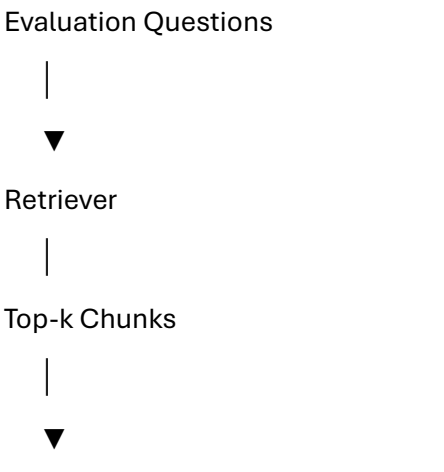
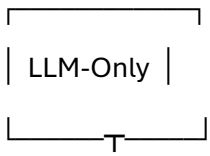
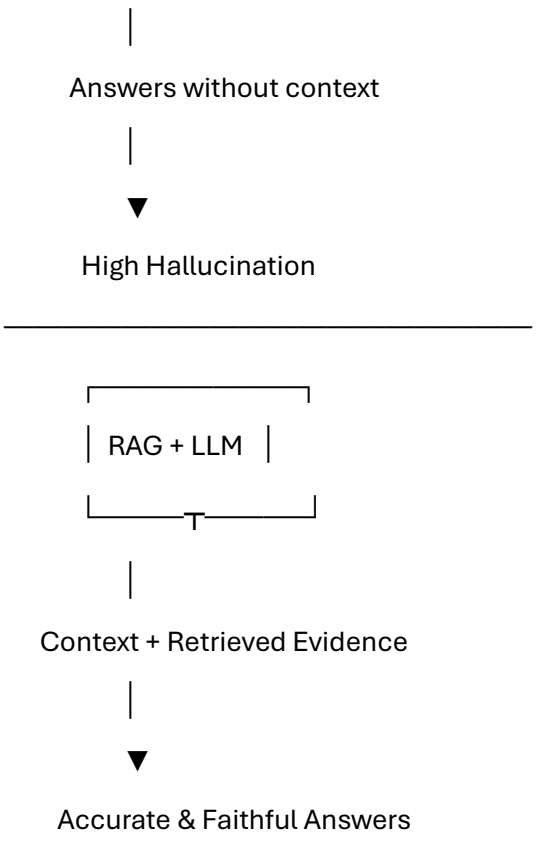


Figure 3 — LLM-only vs RAG+LLM Comparison





10. Raporlama Yapısı (Final)

1. Problem tanımı
2. Veri hazırlığı
3. Chunking stratejileri
4. Embedding seçimleri
5. Vektör DB ayarları
6. Retrieval sonuçları (Retriever-only)
7. LLM-only sonuçları
8. RAG + LLM sonuçları
9. Üç sistemin karşılaştırması
10. Tartışma (zorunlu sorular)
11. Hallucination örnekleri
12. Sonuç & gelecek çalışmalar