**chain\_type Nedir?**

LangChain içinde RetrievalQA kullanırken, retriever'dan gelen verileri LLM'e nasıl aktaracağımızı belirleriz.

Yani: "Belge parçalarını model nasıl birleştirsin, nasıl düşünsün?"

**⚙️ Kullanılabilir chain\_type Seçenekleri:**

**stuff**

* **Tanım:** Tüm chunk’ları alır, birleştirir, tek bir prompt'a gömer ve LLM'e verir.
* **Avantaj:** Basit, hızlı.
* **Dezavantaj:** Chunk sayısı fazla olunca token limiti aşılır, model saçmalayabilir (halüsinasyon artar).
* **Ne zaman kullanılır?** Az sayıda küçük chunk varsa.

**map\_reduce**

* **Tanım:**
  + Her chunk’ı **ayrı ayrı** işleyip cevaplar üretir (map).
  + Sonra bu cevapları toplayıp bir özet çıkarır (reduce).
* **Avantaj:**
  + Daha kontrollü çalışır.
  + Büyük belge miktarlarında **halüsinasyon daha az olur**.
* **Dezavantaj:**
  + Biraz daha yavaş.
  + Prompt token kullanımı daha fazla olabilir.
* **Ne zaman kullanılır?**
  + Chunk sayısı fazla, içerik uzun, doğruluk önemliyse.

**refine**

* **Tanım:**
  + İlk chunk üzerinden taslak bir cevap üretir.
  + Diğer chunk'larla bu cevabı **adım adım iyileştirir**.
* **Avantaj:**
  + Yüksek doğrulukta, sağlam yanıtlar üretebilir.
* **Dezavantaj:**
  + Çok yavaş olabilir.
  + Her adımda yeniden LLM çağrılır.
* **Ne zaman kullanılır?**
  + Belge kronolojisi önemliyse veya detaylı anlatım gerekiyorsa.

**Özetle:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| chain\_type | Hız | Doğruluk | Token Yönetimi | Halüsinasyon Riski |
| stuff | ⚡️ Hızlı | 🟡 Orta | ❌ Kötü | 🔺 Yüksek |
| map\_reduce | 🟡 Orta | ✅ Yüksek | ✅ Daha iyi | 🔽 Düşük |
| refine | 🐢 Yavaş | ✅✅ Çok yüksek | 🟢 Akıllı ama pahalı | 🔽 Çok düşük |