

# Reasoning vs Non-Reasoning Çıktılar

## Karşılaştırmalı Rapor

### 1. Semantic Search Agent

#### İlk Çıktı (reasoning\_content var)

- Extracted Semantic Query: “Software engineers with strong leadership skills”
- Reasoning: Adım adım Chain-of-Thought (CoT) açıklamaları mevcut.
- Çıktı daha kapsamlı, “Software engineers” rolü korunmuş.

#### Son Çıktı (reasoning\_content None)

- Extracted Semantic Query: “strong leadership skills”
- Reasoning: None.
- Yalnızca soft skill kısmı alınmış, rol (software engineer) göz ardı edilmiş.

#### Yorum:

İlk çıktı semantic extraction açısından daha doğru ve eksiksiz. Reasoning'in olması, hem sürecin anlaşılabilirliğini artırıyor hem de confidence score üretimine imkân veriyor. Son çıktıda reasoning olmayınca açıklama ve güven derecesi kayboluyor.

Sonuç: **İlk çıktı daha güvenilir.**

### 2. SQL Parser Agent

#### İlk Çıktı

- JSON output: job\_title + age.
- Reasoning: Mapping adımları açıkça yazılmış.

#### Son Çıktı

- JSON output: job\_title + age.
- Reasoning: None.

#### Yorum:

Her iki durumda da SQL filtreleri aynı. Ancak reasoning içeren ilk çıktı, hangi kriterlerin nasıl eşlendiğini veya neden eşleşmediğini gösteriyor. Reasoning olmayınca sadece doğru JSON veriliyor ama sürecin şeffaflığı kayboluyor, confidence score hesaplanamıyor.

Sonuç: **Doğruluk aynı, ama reasoning açıklama ve güven sağlıyor.**

## 3. Router Agent

### İlk Çıktı

- SQL / Semantic analizi: sql=True, similarity=True.
- Reasoning: Adım adım açıklama mevcut.
- Structured vs semantic ayrımı net.

### Son Çıktı

- SQL / Semantic analizi: sql=True, similarity=True.
- Reasoning: None.

### Yorum:

Her iki durumda da sınıflandırma doğru. Ancak reasoning olmayan versiyon, “neden” sorusuna yanıt veremiyor. İlk çıktı ise hangi kriterin neden SQL, neden semantic olduğunu adım adım açıklıyor.

Sonuç: **İlk çıktı daha şeffaf.**

## 4. Confidence Score Sorunu

- **Gemini model native reasoning modeli değil.**
  - Uyarı:

WARNING Reasoning model: Gemini is not a native reasoning model, defaulting to manual Chain-of-Thought

- Bu yüzden otomatik confidence score üretilemiyor.

- Agno, confidence'u reasoning\_content üzerinden çıkarıyor. Eğer reasoning\_content None ise confidence score da yok.

#### Çözüm Seçenekleri:

- Native reasoning model (örn. gpt-3o-reasoning) kullanmak.
- Veya proxy confidence hesaplamak (ör. SQL/semantic mapping oranı).

## 5. Üç Dosya Kıyaslaması

- **sql\_search\_hco.py (reasoning\_model, reasoning=None)**

Basit query: "strong leadership skills" → reasoning yok, minimal çıktı.

- **sql\_search\_reasoning\_models.py (reasoning=True, manual CoT)**

Query: "Software engineers with strong leadership skills" → reasoning adım adım var, mapping net.

- **sql\_search\_reasoning.py (reasoning=True, detaylı CoT)**

Karmaşık query: "Senior Python developers with cloud experience and strong teamwork skills" → hem semantic hem structured kriterler tek tek ayrılmış, unmappable kriterler belirtilmiş, detaylı reasoning mevcut.

#### Genel Yorum:

- Basit query'de reasoning=False ve reasoning=True benzer doğrulukta, ama reasoning=True şeffaflık katıyor.
- Karmaşık query'de reasoning=True çok daha iyi performans gösteriyor; unmappable kriterler ve structured/semantic ayrımı net görünüyor.

## 6. Reasoning\_models vs Reasoning Flag

#### Reasoning\_models:

- Chain-of-Thought için özel eğitilmiş küçük modeller.
- Ama bazen basit görevlerde doğruluk düşük olabilir; fazla adım üretmeye odaklanır.

### **Reasoning=True Flag:**

- Güçlü genel modeller (GPT-4, GPT-4o vb.) üzerinde çalışır.
- Sadece output tarzını değiştirir (step-by-step düşünme).
- Doğruluk genellikle daha yüksek olur.

### **Karşılaştırma:**

Özellik	<code>reasoning_models</code>	<code>reasoning=True</code>
Amaç	CoT için tasarlanmış	Normal modelde step-by-step tetikleme
Model	Küçük/özel	Büyük güçlü LLM
Doğruluk	Göreve göre düşük	Genellikle yüksek
Avantaj	Karmaşık reasoning için optimize	Yüksek doğruluk + açıklama
Dezavantaj	Basit görevlerde hata	Bazen fazla uzun output

## **7. Router + ReasoningTools**

- **Router + reasoning=True:**

Router metni ayırır, agent'lar kendi işini step-by-step yapar.

→ Basit yapı, kolay debug.

- **Router + ReasoningTools:**

Router yönlendirir, agent'lar `reasoning_tools` ile adım adım düşününebilir ve gerektiğinde tool kullanabilir (ör. web search, hesaplama).

→ Daha güçlü ve esnek, ama kurulum karmaşık.

### **Pratik:**

- Sadece metin analizi → `reasoning=True` yeterli.
- Tool kullanımı gerekirse → `reasoning_tools` eklenmeli.

## 8. Instruction Prompt vs ReasoningTools

- **Instruction (prompt):**

Modelin düşünme tarzını belirler. (“Adım adım çöz, varsayımları açıkla” gibi.)

Ama sadece modelin “içinde” çalışır, dışa yansımaz.

- **ReasoningTools:**

Modelin adımlarını somutlaştırır, intermediate step’leri kaydeder, gerektiğinde araçları çağırır.

### Karşılaştırma:

Katman	Amacı
Instructions (prompt)	Modelin nasıl düşüneceğini yönlendirir
ReasoningTools	Step-by-step reasoning’i kaydeder, tool kullanımını mümkün kılar

### Örnek:

- Sadece instruction: Model doğru cevabı üretir ama ara adımları göremezsin.
- Instruction + ReasoningTools: Model hem step-by-step düşünür, hem intermediate step’leri loglar, hem de tool call yapabilir.

## Genel Özeti

- **Reasoning\_content varsa:**

→ Step-by-step açıklama, unmappable kriterlerin not edilmesi, confidence score üretimi mümkün.

- **Reasoning\_content yoksa:**

→ Çıktı doğru olabilir ama şeffaflık kaybolur, güven skoru hesaplanamaz.

- **Basit query’lerde:** Reasoning olmasa da yeterli.

- **Karmaşık query'lerde:** Reasoning çok daha kritik, özellikle SQL + semantic ayrimında.
- **Reasoning\_models:** küçük, CoT odaklı, doğruluk düşebilir.
- **Reasoning=True flag:** güçlü model + step-by-step düşünce → genelde en iyi kombinasyon.
- **Router + ReasoningTools:** geleceğe dönük daha güçlü yapı; tool kullanımını da yönetebilir.