Agno HR Workflow Projesi

1. Giriş

Bu rapor, Agno framework üzerinde geliştirilen **NLQ → SQL & Semantic Search tabanlı HR Assistant** sisteminin çekirdek modüllerini kapsamlı bir şekilde analiz etmektedir.

Amaç:

- **Memory** entegrasyonu ile kullanıcı tercihlerinin hatırlanması,
- **Reasoning** ile doğruluk ve şeffaflığın artırılması,
- **AccuracyEval / PerformanceEval** ile kalite ve performansın ölçülmesi,
- **Router & SQL Agent** için değerlendirme sonuçlarının raporlanması,
- **Reranker ve Query Builder** gibi yardımcı bileşenlerle arama kalitesinin yükseltilmesi.

2. Memory Ekosistemi

2.1 Genel Tanım

Memory, ajanların (agent) ve iş akışlarının (workflow) bağlamı korumasını ve geçmiş deneyimlerden yararlanmasını sağlar.

Amaç:

- Çok turlu diyalogları sürdürebilmek,
- Tercihleri, geçmiş sorguları ve çıktıları hatırlayabilmek,
- Reasoning için semantik bağlam sunmak.

Bellek türleri:

- **Kısa süreli (Short-term)** → Session bazlı.
- **Uzun süreli (Long-term)** → DB tabanlı.
- **Anlamsal (Semantic)** → semantic search.
- **Epizodik (Episodic)** → geçmiş olaylar.
- **İşlemsel (Procedural)** → karmaşık süreçleri otomatik çözme.

2.2 Session Bazlı Bellek

- **AgentSession:** Tek agent'ın tüm runlarını saklar, kısa süreli.
- **WorkflowSession:** Çok-agent'lı iş akışlarında bağlam paylaşımı sağlar.
- Avantaj: Hafif, performanslı, DB'ye bağımlı değil.
- Dezavantaj: Kalıcı değil.

proje için:

Recruiter'ın yaptığı aramaları `WorkflowSession.session_state` içinde saklamak en ideal çözüm.

2.3 Memory Araçları

- **MemoryTools:** CRUD işlemleri (add, get, update, delete, analyze).
- **Mem0AI Tool:** Kullanıcı odaklı uzun vadeli hafıza. Semantic search destekler.

Kullanım önerisi:

- Kısa vadeli tercihler → session bazlı.
- Uzun vadeli tercih ("her zaman Almanca bilen adaylar") → Mem0AI.

2.4 Memory Manager

Görevi: Kullanıcı mesajlarından önemli bilgileri yakalamak ve DB'ye kaydetmek.

Fonksiyonlar:

- create_user_memories, update_memory_task, search_user_memories, clear_memories.
- Retrieval yöntemleri: last_n, first_n, agentic (semantic).

proje için:

"Ben artık sadece onsite aday istiyorum" → DB'ye yazılır, sonraki aramalarda otomatik eklenir.

2.5 Context Entegrasyonu

- `add_memories_to_context=True` → Memory otomatik cevaba eklenir.
- `False` → Manuel çağrılır.

2.6 HR Search Uygulaması

- Ana yapı: WorkflowSession → oturum bağlamı.
- Ek yapı: MemoryManager → kritik tercihleri kalıcı yapar.
- Opsiyonel: Mem0AI → recruiter profilini uzun vadeli yönetir.

Sonuç: Session-first yaklaşım + MemoryManager ile kişiselleştirilmiş, performanslı HR search deneyimi.

3. Reasoning (Mantık Yürütme)

3.1 Problem

LLM'ler kara kutu; neden o cevabı verdiğini açıklayamıyor.

3.2 Çözüm

Reasoning mekanizmaları → adım adım düşünme → şeffaflık.

3.3 Bileşenler

- **Reasoning Flag:** `reasoning=True`, step-by-step düşünme.
- **Reasoning Tools:** Tool koordinasyonu, CoT planlama.
- **Reasoning Agent:** Confidence score ve reasoning log üretir.

3.4 Deneysel Sonuçlar

- SQL Agent: 0.3–0.4 → `reasoning=True` ile 0.8.
- Semantic Agent: 1.0 → `reasoning=True` ile 0.8.
- SQL tarafında ciddi artış, semantic'te hafif düşüş.

3.5 Optimizasyon

- **is_complex_query:** Karmaşık sorgularda reasoning açılır.
- **Loop & Condition:** Skor düşükse workflow tekrar çalışır.
- **Session Cache:** Tekrarlayan sorgularda hız.

3.6 Yorum

Şu an için **Reasoning Flag yeterli**.

Dış veri (LinkedIn, Kariyer.net) eklenirse → Reasoning Tools/Agents gerekli olur.

4. AccuracyEval ve Router Sonuçları

4.1 Genel Mekanizma

- Agent output ↔ expected output kıyaslanır.
- Accuracy + completeness skorlanır.
- 1–10 arası skor + gerekçe.

4.2 Router Agent Deneyi

- 14 query test edildi.
- Ortalama skor: **8.6 / 10**.
- Güçlü: Boolean bayraklar doğru, reasoning detaylı.
- Zayıf: Job title & experience year kriterlerinde uyumsuzluk.

4.3 Öneriler

- Job title için hybrid kural.
- Experience → default semantic.
- Partial credit sistemi eklenmeli.

5. PerformanceEval

5.1 Neden Var?

- Ne kadar hızlı?
- Kaç MB RAM tüketiyor?
- Memory leak var mı?

5.2 Mimari

- Runtime ölçümü (warm-up + tekrar).
- Memory leak analizi (tracemalloc).
- DB logging → Grafana entegrasyonu.

5.3 Kullanım Alanları

- Router, SQL, Semantic agent hız karşılaştırması.
- Workflow latency ölçümü.
- Model benchmark (GPT-4 vs GPT-5-mini).

AccuracyEval + PerformanceEval → Kalite + Verimlilik ikilisi.

6. Benchmark ve CI/CD

- Benchmark = Kıyaslama standardı.
- Sabit dataset + ground truth → model skorları ölçülür.
- Continuous Evaluation → her push sonrası AccuracyEval testleri koşar.
- CI/CD entegrasyonu → kaliteyi bozan versiyon canlıya çıkmaz.

7. Reranker ve Query Katmanı

7.1 Neden?

Embedding → hızlı ama kaba sıralama.

Reranker (CrossEncoder) → daha doğru final sıralama.

7.2 Çözümler

- **Cohere Reranker:** Production-ready, multilingual.
- **SentenceTransformers Reranker:** Açık kaynak, offline, fine-tune edilebilir.
- **Infinity Reranker:** Self-hosted, gizlilik + maliyet kontrolü.

7.3 Kullanım

- SQL + Semantic sonuçları → top-K.
- Reranker ile yeniden sıralama.
- Final relevance score'a göre sonuçlar.

8. SQL Parser Agent Sorunları

8.1 Eksiklikler

- Domain/sector bilgilerini (bankacılık, fintech) source alanına mapleyemiyor.
- Status (ihbar süresi, availability) kriterlerini kaçırıyor.

8.2 Yanlışlar

- contains yerine equals kullanımı.
- Relative tarihleri absolute'a çevirmesi.
- Fazladan unit=null, currency=null gibi alanlar.

8.3 Çözüm

- Prompt güçlendirme: "sector → source, status → status alanı".
- Post-processing: Gereksiz alanları temizle.
- Değerlendirmeyi Accuracy + Completeness ayrı skorla yap.

9. Genel Sonuç ve Yol Haritası

- **Memory** → Session-first + MemoryManager + opsiyonel Mem0AI.
- **Reasoning** → SQL'de doğruluk artışı, semantic'te hafif düşüş ama faydalı.
- **AccuracyEval** → Router 8.6/10 → güvenilir.
- **PerformanceEval** → Ölçeklenebilirlik için kritik.
- **Reranker** → Hibrit aramada kaliteyi yükseltiyor.
- **SQL Parser** → domain/status mapping sorunları çözülmeli.
- **Benchmark + CI/CD** → Sistem sürekli test edilmeli.

Önerilen Yol Haritası:

1. MVP: Router + SQL + Semantic pipeline (Memory olmadan).
2. Faz 2: MemoryManager + AccuracyEval entegrasyonu.
3. Faz 3: Reranker ekleme (Cohere veya ST).
4. Faz 4: Benchmark dataset + CI/CD pipeline.
5. Faz 5: Domain-specific fine-tuning (SQL parser + reranker).

Agno toolları: Docker tools, OpenCV tools, wtools, apple whisper tools, knowledge tools, reasoning tools, SQL tools

DockerTools, Agno framework içindeki bir araç setidir.

Bir **Agent (yapay zekâ ajanı)** bu araç sayesinde doğrudan **Docker konteynerleri, imajları, volume'leri ve network'leri yönetebilir**.

Yani, sen terminalden `docker ps`, `docker run`, `docker logs` gibi komutlar yazmak yerine, ajana "List all running containers" diye yazıyorsun → Ajan DockerTools üzerinden bu işlemleri yapıyor.

Agno Framework Hakkında Basit Bilgiler

- **Agno**, yapay zekâ ajanları (AI Agents) üzerine kurulmuş bir framework'tür.
- Bu ajanlar, senin verdiğin komutları alıp belirli **araç setleri (tools)** üzerinden gerçek işlemler yapabilir.
- Örneğin:
  - Web'de arama yapmak için **WebTools**,
  - Veritabanında sorgu çalıştırmak için **SQLTools**,
  - Docker konteynerlerini yönetmek için **DockerTools** kullanılır.

DockerTools Örneği

- Normalde terminalden `docker ps`, `docker run`, `docker logs` gibi komutlar yazmak gerekir.
- Agno'da ise bir ajana **doğal dil** ile şunu söyleyebilirsin:
  - *"List all running containers"* (Tüm çalışan konteynerleri listele)

- Ajan, **DockerTools** aracını kullanarak bu komutu kendisi çalıştırır ve sonucu sana getirir.

Agno’nun Avantajı

- Kullanıcı, karmaşık teknik komutlarla uğraşmak zorunda kalmaz.
- Ajanlar, verilen görevi doğru araca yönlendirir.
- Böylece hem yazılımcılar hem de teknik olmayan kullanıcılar, doğal dil kullanarak işlem yapabilir.