Bir AI Agent Nedir?

Bir AI agent'i tanımlayan şey: gözlemle-karar ver-eyleme geç loop'u, tek bir tool call'dan farkı, ve otonomi spektrumu.

Orta 18 dk

Son güncelleme: 5 Ekim 2026
Çevrimiçi oku: https://www.learnforgex.com/tr/topics/what-is-an-ai-agent

---

Bir AI Agent Nedir?

Bu kategoriye daha önce birkaç yerden atıfta bulunuldu, ama hiçbir yerde tanımlanmadı. "Tools and Function Calling", tek bir tool call ile "bir hedefe doğru birden fazla adım planlayabilen daha geniş bir sistem" arasına bir çizgi çekmişti, ama ikincisini tanımlamadan bırakmıştı (bkz. "Tool Use ve Agent Karşılaştırması" bölümü). "Machine Learning"deki "Üç Öğrenme Türü: Supervised, Unsupervised, Reinforcement" bölümü, reinforcement learning'in *agent*'ının bu kavrama kavramsal olarak en yakın, halihazırda var olan fikir olduğunu belirtmişti. "Generative AI Ne Değildir?" bölümü, tek bir prompt'ta bir paragraf ya da bir görsel üretmenin tek başına "agentic" olmadığını vurgulamıştı. Bu ders, üçünün de işaret ettiği soruyu nihayet yanıtlıyor: bir sistemi bir model'den, bir chatbot'tan ya da tek bir tool call'dan ayıran, spesifik olarak nedir?

Bir AI Agent Nedir?

Bir **AI agent**, bir ya da daha fazla model call'u ve tool use üzerine kurulu, bir hedefi, sırada ne yapacağına tekrar tekrar karar vererek, bir eylem alarak, sonucu gözlemleyerek, ve tekrar karar vererek -- hedef karşılanana ya da durmaya karar verene kadar -- takip eden bir sistemdir. Bunu "Tools and Function Calling"in zaten kapsadığından ayıran üç şey var: kararların *sırası* önceden bir insan tarafından sabitlenmemiştir, kaç adım gerekeceği önceden *bilinmez*, ve her karar, tek seferlik bir prompt-yanıt alışverişi değil, bir önceki adımda ne olduğuyla bilgilendirilir. Tek bir soruyu yanıtlayan tek bir tool call, bu tanıma göre bir agent değildir; bir şeyi kendi kararıyla arayan, sonucun soruyu yanıtlayıp yanıtlamadığını kontrol eden, ve yanıtlamıyorsa başka bir şey arayan bir sistem ise bir agent'tır.

Neden Var?

Bazı problemler, tek bir prompt ve tek bir tool call ile çözülemez, çünkü doğru adım sırası ancak önceki adımlar zaten çalıştıktan sonra bilinir. Başarısız bir testi debug etmek, bir şey

denemeyi, işe yarayıp yaramadığını gözlemlemeyi, ve o sonuca göre sırada neyi deneyeceğine karar vermeyi gerektirir. Birden fazla kaynak üzerinden bir konuyu araştırmak, birini okumayı, soruyu yanıtlayıp yanıtlamadığına karar vermeyi, ve yanıtlamadıysa daha fazla aramayı gerektirir. Çok adımlı bir görevi tamamlamak -- bir uçuşu yalnızca belirli bir fiyatın altındaysa rezerve et, değilse başka bir tarihi kontrol et -- görev başladığında henüz var olmayan bir sonuca bağlı bir karar gerektirir. Bunların hiçbiri, "Tools and Function Calling"in anlattığı tool-calling loop'a uymaz -- orada bir request gider, bir sonuç gelir, ve sıradaki her adımı hâlâ bir insan yönlendirir. Agent'lar tam da bu sınıftaki problemleri ele almak için var: gerekli adımların, ve kaç tane olduğunun, ancak süreç ilerledikçe belirlenebildiği hedefler.

Agent Loop: Gözlemle, Karar Ver, Eyleme Geç

Bir agent'ın çalışması tek bir request/response alışverişi değil, bir loop'tur:

1. **Gözlemle (Observe)** -- agent, hedefine ve bu çalışmada şu ana kadar olan her şeye bakar: önceki kararlar, tool call'lar, ve sonuçları.
2. **Karar ver (Decide)** -- bu gözleme dayanarak, agent (bir model call'u aracılığıyla) tam olarak bir sonraki adıma karar verir: belirli argümanlarla belirli bir tool'u çağırmak, ya da hedefin karşılandığı sonucuna varıp nihai bir yanıt üretmek.
3. **Eyleme geç (Act)** -- bir tool seçildiyse, gerçekten çalıştırılır (bu loop'u saran uygulama tarafından, tam olarak "Tool-Calling Döngüsü"nün anlattığı gibi -- modelin kendisi yine hiçbir şeyi hiçbir zaman çalıştırmaz), ve gerçek sonucu bir sonraki gözlemin parçası olur.

Bu, "Tool-Calling Döngüsü"nün doğrudan bir genellemesidir: bir kez çalışıp durmak yerine, aynı gözle-karar ver-eyleme geç döngüsü, her seferinde kendinden önce olan her şeyle bilgilendirilerek tekrarlanır -- ta ki "Karar ver" adımı hedefin karşılandığı sonucuna varana kadar (ya da, bu kategorinin sonraki bir dersi olan "Agent Davranışını Kontrol Etmek"in anlattığı gibi, bir güvenlik sınırına ulaşılana kadar).

Tek Bir Tool Call'dan Agent'a

"Tool Use ve Agent Karşılaştırması" bu çizgiyi zaten yüksek seviyede çizmişti; yukarıdaki loop artık masadayken bunu tam olarak yeniden ifade etmekte fayda var. Tek bir request /response alışverişi içinde bir tool'u bir kez kullanmak, bir şeyi agent yapan şey değildir -- bugünün havasını sorulduğunda arayan sıradan bir chatbot, bunu yaparak bir agent olmaz. Bir sistemi agent yapan şey, bir önceki bölümdeki loop'un *sistemin kendi kontrolü altında* çalışmasıdır: hangi tool'un çağrılacağına, ne zaman bir tane daha çağrılacağına, ve durmak için yeterince bilgi toplandığına -- bir insan değil, sistemin kendisi karar verir -- bu, ulaşılması kaç adım süreceği önceden bilinemeyen bir hedefe doğru, birden fazla adım boyunca sürdürülür.

Otonomi Spektrumu

"Agent" sabit bir bağımsızlık miktarını değil, bir spektrumu tanımlar. Bir uçta, bir sistemin her kararı fiilen bir insan tarafından önceden senaryolanmıştır (tek bir tool call bu uca yakındır). Diğer uçta, bir sistem birçok karar ver-eyleme geç döngüsünü tamamen kendi başına, işi bitene kadar kimseye danışmadan çalıştırır. Pratikte gerçekten kullanışlı olan agent'ların çoğu bu iki uç arasında bir yerde durur: bazı kararlar tamamen yukarıdaki loop'a bırakılırken, maliyetli, geri döndürülemez ya da riskli olan her şey, ilerlemeden önce bilinçli olarak bir insana yönlendirilir. Belirli bir agent'ın bu spektrumda tam olarak nerede durması gerektiği, ve bu seçimi kodda nasıl zorunlu kılacağınız, bu kategorinin sonraki dersi "Agent Davranışını Kontrol Etmek"in konusu.

Bu Kategori Buradan Sonra Nereye Gidiyor?

Bu ders loop'u tanımladı; kategorinin geri kalanı bunun üzerine inşa ediliyor. "Agent Planlama ve Akıl Yürütme Kalıpları", yukarıdaki "Karar ver" adımına daha derinlemesine iniyor -- o kararın gerçekte nasıl verildiğine dair somut kalıplar (ReAct, plan-and-execute, ve reflection gibi), "model karar veriyor"un ötesinde. "Agent Davranışını Kontrol Etmek", böyle bir loop'un gerçek bir otonomiyle çalışmadan önce ihtiyaç duyduğu korkulukları kapsıyor: adım sınırları, insan onay noktaları, ve observability. "TypeScript ile Bir AI Agent Oluşturma" ise, burada anlatılan her şeyi somutlaştırmak için gerçek, çalışan bir agent loop inşa ediyor -- "TypeScript ile MCP Sunucusu Oluşturma"da `GeoFactsServer.ts`'nin sunduğu tam olarak aynı MCP tool'larını yeniden kullanarak.

Best Practices

- "Agent" kelimesini, "Agent Loop: Gözlemle, Karar Ver, Eyleme Geç"teki loop'u gerçekten kendi kontrolü altında çalıştıran bir sistem için saklayın -- tek bir tool call'ın neden yeterli olmadığı için bkz. "Tek Bir Tool Call'dan Agent'a".
- Agent-şeklinde bir özellik inşa etmeden önce, onu bilinçli olarak "Otonomi Spektrumu" üzerine yerleştirin -- gerçekte ne kadar bağımsızlığa ihtiyacı olduğuna karar vermek, tasarımının geri kalanını şekillendirir.
- Tek bir geçiş için değil, loop'un kendisi için tasarım yapın -- bir adımın sonucunun sıradaki adımı gerçekten değiştirmesi beklenir, yalnızca bir transkript'e eklenmesi değil.

Yaygın Hatalar

- **Bir tool kullanan her sisteme "agent" demek.** "Tool Use ve Agent Karşılaştırması" ve "Tek Bir Tool Call'dan Agent'a"nın ikisinin de açıkladığı gibi, sıradan bir alışveriş içindeki tek bir tool call bir agent değildir -- ayırt edici özellik, tool use'un kendisi değil, kendi kendini yönlendiren loop'tur.

- **Daha fazla otonominin otomatik olarak daha iyi olduğunu varsaymak.** "Otonomi Spektrumu" bir nedenden dolayı bir spektrumdur -- daha fazla gözetimsiz adım, maliyetli ya da geri döndürülmesi zor bir hatanın daha fazla şansı demektir, ki bu tam olarak "Agent Davranışını Kontrol Etmek"in var olma nedenidir.
- **Bir agent'ın adım planlama yeteneğini gerçek akıl yürütme ya da anlama ile karıştırmak.** Bir sonraki adıma karar veren model, hâlâ "LLM Yetenekleri ve Sınırlamaları"daki "Akıl Yürütme Sınırları"nın kapsadığı aynı türden bir modeldir -- bir dizi tool call'u planlamak, bu altta yatan sınırları ortadan kaldırmaz.

Özet, Cheat Sheet ve Terimler Sözlüğü

Özet

- Bir AI agent, sırada ne yapacağına tekrar tekrar karar vererek, eyleme geçerek, ve sonucu gözlemleyerek bir hedefi takip eden bir sistemdir -- tek bir tool call ya da tek bir model yanıtı değil.
- Agent'lar, doğru adım sırasının (ve kaç adım gerektiğinin) önceden bilinemediği ve önceki adımların sonucuna bağlı olduğu problemler için var.
- Agent loop -- gözlemle, karar ver, eyleme geç -- "Tools and Function Calling"deki "Tool-Calling Döngüsü"nün doğrudan bir genellemesidir, bir kez çalışmak yerine sistemin kendi kontrolü altında tekrarlanır.
- Bir agent'ı tek bir tool call'dan ayıran şey, o loop'un kendi başına çalışması, yalnızca bir tool call'ın varlığı değildir.
- Otonomi bir ikili değil, bir spektrumdur -- pratikteki çoğu agent, kendi kendine yönlendirilen adımları bilinçli insan kontrol noktalarıyla karıştırır, ki bu "Agent Davranışını Kontrol Etmek"in konusudur.

Cheat Sheet

- Agent = hedef + gözlemle-karar ver-eyleme geç loop'u, bir insan değil, sistemin kendi kontrolü altında, birden fazla adım boyunca çalışır.
- Loop = genelleştirilmiş tool-calling loop: gözlemle (hedef + geçmiş) -> karar ver (model sıradaki eylemi seçer) -> eyleme geç (gerçek tool çalıştırma) -> tekrarla.
- Tek tool call != agent. Loop'un adımlar boyunca kendi başına çalışması, onu agent yapan şeydir.
- Otonomi spektrumu: tamamen senaryolanmış <-> tamamen otonom; gerçek agent'ların çoğu arada bir yerde durur.

Terimler Sözlüğü

- **AI agent:** bir hedefi, eylemlere tekrar tekrar karar vererek ve onları alarak, sonuçlarını gözlemleyerek, ve tekrar karar vererek -- belirli bir otonomi derecesiyle -- takip eden bir sistem.

- **Agent loop:** bir agent'ın hedefi karşılanana ya da durdurulana kadar tekrarladığı gözlemle-karar ver-eyleme geç döngüsü.
- **Otonomi spektrumu:** bir agent'ın tasarımının, tamamen insan tarafından senaryolanmış davranıştan tamamen gözetimsiz, çok adımlı davranışa kadar herhangi bir yerinde durabileceği aralık.