Agent Davranışını Kontrol Etmek

Bir agent loop'unu güvenli kılan korkuluklar: adım sınırları, human-in-the-loop onayı, en az yetki ilkesi, ve observability.

Orta 18 dk

Son güncelleme: 22 Ağustos 2026
Çevrimiçi oku: https://www.learnforgex.com/tr/topics/controlling-agent-behavior

Agent Davranışını Kontrol Etmek

"Bir AI Agent Nedir?", her agent'ı "Otonomi Spektrumu"nda bir yere yerleştirmişti -- ve daha fazla otonominin, bir insanın her birini kontrol etmediği daha fazla karar anlamına geldiğini belirtmişti. "Agent Planlama ve Akıl Yürütme Kalıpları", iyi tasarlanmış bir karar adımının bile, prensipte, bir termination koşulu olmadan sonsuza kadar dönebileceğini göstermişti. Bu ders, gerçek bir otonomiyle bir agent çalıştırmayı yaşanabilir kılan pratik mekanizmaları kapsıyor: adım sınırları, insan onay noktaları, bir agent'ın dokunabileceği şeyleri kısıtlamak, ve gerçekte ne yaptığını görebilmek.

Agent'lar Neden Korkuluğa İhtiyaç Duyar?

Tek bir tool call hakkında akıl yürütmek kolaydır: bir request, bir response, onu göndermeye bir insan karar vermiştir. Bir agent loop'u farklıdır -- sırayla birçok eylem alabilir, bunları bir insan değil model seçer, ve bu eylemlerin her biri gerçek bir yan etkisi olan (bir e-posta göndermek, bir kaydı değiştirmek, para harcamak) gerçek bir tool call olabilir. Agent'ları kullanışlı kılan şey -- her adım için loop'ta bir insan olmadan karar verip eyleme geçmek -- aynı zamanda kısıtlanmamış bir agent'ı riskli kılan şeydir: yanlış bir karar adımı, ya da yanlış bir argümanı hedefleyen bir tool call, çalışmadan önce bir insan tarafından yakalanmaz. Korkuluklar, o otonominin kullanışlı kısmını korurken bir hatanın verebileceği zararı sınırlamak için var.

Adım ve İterasyon Sınırları

"Agent Planlama ve Akıl Yürütme Kalıpları"daki "Termination: Ne Zaman Duracağını Bilmek" bu fikri zaten tanıtmıştı: bir agent, model hedefin karşılandığına karar verdiğinde durmalı, ama bu hiç gerçekleşmediğinde sağlam bir yedeğe de ihtiyacı var. Bir **adım sınırı** (ya da iterasyon sınırı), tam olarak bu yedektir -- loop'un çalışmasına izin verilen maksimum karar ver-eyleme geç döngüsü sayısı, modelin kendi yargısına bırakılmak yerine kodda zorunlu

kılınır. Sınıra ulaşıldığında, ne yapıyor olursa olsun loop koşulsuz olarak durur -- bu, bilinçli olarak kaba bir mekanizmadır, ve mesele de budur: agent'ın "neredeyse bitmiş" olup olmadığı konusunda akıllı davranmaya çalışmaz, sadece loop'un süresiz olarak kaçamayacağını garanti eder. "TypeScript ile Bir AI Agent Oluşturma", kendi agent loop'unun etrafında tam olarak bu türden bir sınırı uyguluyor.

Human-in-the-Loop: Riskli Eylemlerden Önce Onay

Bir agent'ın alabileceği her eylem aynı riski taşımaz. Bir şeyi aramak geri almayı kolaydır (daha doğrusu, geri alınacak bir şeyi yoktur); bir mesaj göndermek, bir kaydı silmek, ya da para harcamak öyle değildir. **Human-in-the-loop**, belirli eylemleri -- tipik olarak maliyetli, geri döndürülemez, ya da otomatik olarak doğrulanması zor olanları -- loop'un onları diğer her tool call gibi otomatik çalıştırmasına izin vermek yerine, çalışmadan önce açık bir onay adımından geçirmek anlamına gelir. Bu, her eylemin onay gerektirdiği anlamına gelmez -- bu, en başta bir agent'a sahip olmanın anlamını ortadan kaldırır -- bunun yerine, yanlış bir kararın yeterince pahalıya mal olduğu, kısa bir insan onay duraklamasının kaybedilen hıza değdiği eylem alt kümesini bilinçli olarak işaretlemek anlamına gelir, belirli bir agent'ın "Otonomi Spektrumu"nda nerede durduğuna geri bakarak.

Tool Erişimini Kapsamlandırmak: En Az Yetki İlkesi

Bir agent yalnızca mevcut tool'larının izin verdiği eylemleri alabilir -- bu yüzden neyin yanlış gidebileceğini sınırlamanın en doğrudan yollarından biri, tool listesinin kendisini sınırlamaktır. Bir agent'a, gerçek işi yalnızca kayıtları okumayı gerektirdiğinde herhangi bir kaydı silebilen bir tool vermek, agent'ın karar vermesinin ne kadar iyi olduğuyla hiçbir ilgisi olmayan bir risk yaratır -- bu risk yalnızca kötüye kullanılabilecek yetenek mevcut olduğu için vardır. **En az yetki ilkesi** -- bir görevin ihtiyaç duyduğu erişimi tam olarak vermek, daha fazlasını değil -- bir agent'ın tool'larına, bir kullanıcı hesabının izinlerine uygulandığı şekilde uygulanır: daha dar tool erişimi, agent'ın akıl yürütmesini hiçbir şekilde iyileştirmez, ama kötü bir kararın (ister kusurlu bir karar adımından, ister beklenmedik bir girdiden gelsin) verebileceği zararın alanını küçültür.

Observability: Bir Agent'ın Kararlarını Loglamak ve İzlemek

Bir agent'ın kesin eylem sırası önceden senaryolanmadığı için -- ki bu, "Bir AI Agent Nedir?" deki "Agent Loop: Gözlemle, Karar Ver, Eyleme Geç"in tüm meselesidir -- gerçekte ne yaptığını sonradan bilmek, tamamen onu kaydetmiş olmaya bağlıdır. Burada **observability**, loop'un her adımını loglamak anlamına gelir: karar adımının ne seçtiği, hangi argümanları kullandığı, gerçek tool sonucunun ne olduğu, ve durmadan önce kaç adım çalıştığı. Bu olmadan, yanlış bir nihai yanıtı debug etmek neredeyse imkansızdır -- modelin yanlış akıl yürüttüğünü mü, doğru tool'u yanlış argümanlarla mı çağırdığını, yoksa doğru bir sonuç alıp onu yanlış mı kullandığını söylemenin hiçbir yolu yoktur. "TypeScript ile Bir AI Agent Oluşturma", tam da bu nedenle, kendi agent loop'u için adım adım bu türden bir izi yazdırıyor.

Best Practices

- Yanlış davranması pek olası görünmeyenler dahil, her agent loop'unda bir adım sınırı belirleyin -- bkz. "Adım ve İterasyon Sınırları" -- nadiren tetiklenen bir korkuluk yine de boşa gitmiş değildir.
- Hangi eylemlerin insan onayına ihtiyacı olduğuna, eylemin kodda ne kadar karmaşık göründüğüne değil, maliyete ve geri döndürülebilirliğe göre karar verin -- bkz. "Human-in-the-Loop: Riskli Eylemlerden Önce Onay".
- Bir agent'ın tool'larını, görevinin gerçekten gerektirdiği şeye göre verin, ve görev değiştikçe o listeyi yeniden kontrol edin -- bkz. "Tool Erişimini Kapsamlandırmak: En Az Yetki İlkesi".
- Yalnızca nihai yanıtı değil, loop'un her adımını loglayın -- bkz. "Observability: Bir Agent'ın Kararlarını Loglamak ve İzlemek" -- arkasında bir iz olmayan bir nihai yanıt, yanlış olduğunda debug edilemez.

Yaygın Hatalar

- **Loop'u durdurmak için yalnızca modelin kendi yargısına güvenmek.** "Adım ve İterasyon Sınırları"nın açıkladığı gibi, bir karar adımı hâlâ "LLM Yetenekleri ve Sınırlamaları"daki "Akıl Yürütme Sınırları"nın kapsadığı aynı türden bir modeldir -- sağlam bir adım sınırı, tam olarak kendi yargısının bir sonuca varmadığı durumlar için bir yedektir.
- **Her tek eylem için insan onayı istemek.** "Human-in-the-Loop: Riskli Eylemlerden Önce Onay"ın belirttiği gibi, bu, bir agent'ı en başta kullanışlı kılan otonomiyi ortadan kaldırır -- onay, hepsini tekdüze değil, bir hatanın maliyetli olduğu belirli eylemleri hedeflemelidir.
- **Bir agent'a "belki gerekir diye" geniş tool erişimi vermek.** "Tool Erişimini Kapsamlandırmak: En Az Yetki İlkesi", bunun, agent'ın karar vermesini hiçbir şekilde daha güvenilir kılmadan kötü bir kararın verebileceği zararı nasıl genişlettiğini kapsadı.
- **Nihai yanıt doğru göründüğü için eksik bir logu zararsız saymak.** "Observability: Bir Agent'ın Kararlarını Loglamak ve İzlemek", izi olmayan yanlış bir yanıtın sonradan teşhis edilmesinin neredeyse imkansız olduğunu anlattı -- iz, tam olarak ihtiyaç duyulduğunda önemlidir, ki bu önceden tahmin edilemez.

Özet, Cheat Sheet ve Terimler Sözlüğü

Özet

- Korkuluklar, bir agent'ın kararları ve eylemleri bir insan her birini kontrol etmeden gerçekleştiği için var -- agent'ları kullanışlı kılan aynı otonomi, kısıtlanmamış bir agent'ı riskli kılan şeydir.

- Bir adım sınırı, modelin kendi karar adımının hedefin karşılandığı sonucuna varıp varmadığından bağımsız olarak, loop'u ne olursa olsun durduran, kodla zorunlu kılınan sağlam bir yedektir.
- Human-in-the-loop, belirli maliyetli ya da geri döndürülemez eylemleri bir onay adımından geçirir, tekdüze olarak onay istemek (ya da reddetmek) yerine.
- En az yetki ilkesi, bir agent'ın tool erişimini görevinin gerçekten gerektirdiğiyle sınırlar, kötü bir karardan doğabilecek zarar alanını küçültür.
- Observability -- yalnızca nihai yanıtı değil, her adımı loglamak -- yanlış bir sonucu sonradan debug edilebilir kılan şeydir.

Cheat Sheet

- Adım sınırı = karar ver-eyleme geç döngülerinde kodla zorunlu kılınan sağlam bir maksimum, ulaşıldığında loop'u her zaman durdurur.
- Human-in-the-loop = her eylem değil, belirli maliyetli/geri döndürülemez eylemlerden önce açık bir onay adımı.
- En az yetki = agent'ın tool'ları, "belki gerekir diye" daha geniş değil, görevinin gerektirdiği ile tam olarak sınırlı.
- Observability = tüm çalışmayı kapsayan, yalnızca nihai yanıtı değil, adım başına bir log/iz (karar, argümanlar, gerçek sonuç).

Terimler Sözlüğü

- **Korkuluk (guardrail):** otonom bir agent loop'unun ne yapabileceğini, kendi kararlarının doğruluğundan bağımsız olarak sınırlayan bir mekanizma.
- **Adım sınırı (iterasyon sınırı):** bir agent loop'unun durmaya zorlanmadan önce çalıştırabileceği karar ver-eyleme geç döngüsü sayısına sağlam bir maksimum.
- **Human-in-the-loop:** belirli agent eylemlerini, çalışmadan önce açık bir insan onay adımından geçirmek.
- **En az yetki ilkesi:** bir görevin ihtiyaç duyduğu erişimi (burada, tool listesini) tam olarak vermek, daha fazlasını değil.
- **Observability:** bir agent'ın sonradan gerçekte hangi adımları attığını ve nedenini görebilme yeteneği -- tipik olarak loop'un her adımını loglayarak ya da izleyerek.