Tools and Function Calling

Bir LLM'in gerçek kod çalıştırmasını isteyebilmesini sağlayan tool use/function calling mekanizması: tool-calling loop, bir tool'u tanımlamak, ve tool use'un bir "agent"tan farkı.

Orta 18 dk

Son güncelleme: 5 Ekim 2026
Çevrimiçi oku: https://www.learnforgex.com/tr/topics/tools-and-function-calling

---

Tools and Function Calling

"LLM Yetenekleri ve Sınırlamaları" tek bir kök nedenle kapanmıştı: bir LLM makul metin tahmin eder, yerleşik bir doğrulayıcısı, hesap makinesi, canlı veri bağlantısı ya da dünyada gerçekten bir şey *yapma* yolu yoktur. Tek başına bir model yalnızca daha fazla metin üretebilir. **Tools** (**function calling** olarak da adlandırılır), bu açığı kapatan mekanizmadır -- bir modelin, modelin dışında çalışan gerçek kodu tetiklemesine ve sonucunu kullanmasına izin verir. Bu ders, standart protokolü ele alan "MCP'ye Giriş"ten önce, tool use'un aslında ne olduğunu ele alıyor.

Tool Use (Function Calling) Nedir?

Tool use, bir LLM'in yalnızca nihai bir cevap üretmek yerine, belirli bir fonksiyonu çağırmak için *yapılandırılmış bir istek* üretebildiği -- bu fonksiyonu adlandırıp ona argümanlar sağladığı -- ve modelin dışındaki bir programın bu fonksiyonu gerçekten çalıştırıp sonucu döndürdüğü bir kalıptır. Modelin kendisi hiçbir zaman kod çalıştırmaz, hiçbir zaman bir ağa ya da veritabanına doğrudan dokunmaz -- daha doğru bir zihinsel model: model, dış sistemlerdeki işlemleri kendisi gerçekleştirmez, bunları yapılandırılmış tool-call çıktılarıyla kendisini barındıran uygulamadan *ister*. ("Yalnızca metin üretir" ifadesi bu fikrin sadeleştirilmiş bir anlatımıdır -- multimodal modeller ya da sağlayıcıya özgü çıktı biçimleri söz konusu olduğunda bunu mutlak bir teknik iddia olarak okumayın; burada önemli olan model çıktısının biçimi değil, modelin dış dünyaya asla doğrudan erişememesidir.) Tool use'u sıradan bir yanıttan farklı kılan şey, üretilen metnin (modelin etrafında kurulu uygulama tarafından) kullanıcıya doğrudan gösterilmek yerine bir tool call olarak tanınacak şekilde bilinçli olarak biçimlendirilmiş olmasıdır. Bu özelliğin "tool use," "function calling" ya da "tool calling" olarak adlandırılması sağlayıcıya göre değişir, ama altında yatan fikir burada anlatılanla aynıdır.

Neden Var?

"LLM Yetenekleri ve Sınırlamaları"dan hatırlayın: bir modelin bilgisi **knowledge cutoff**'unda durur, ve model gerçekleri doğrulayamaz, ölçekte güvenilir hesaplama yapamaz, ya da **context**'i dışında hiçbir şeye erişemez (bkz. "Large Language Model'ler Nasıl Çalışır?" dersindeki "Context'e İlk Bakış"). Bunların hiçbiri modeli büyüterek ya da farklı eğiterek düzeltilemez -- bu, bir LLM'in ne olduğunun yapısal bir sonucudur: bir sonraki token tahmincisi. Tool use, bu yapısal sınırı yamamaya çalışmak yerine ETRAFINDAN DOLAŞMAK için var: güncel, kesin ya da gerçek bir eyleme dayalı olması gereken her şey için -- bugünün havası, bir veritabanı sorgusu, kesin bir aritmetik sonuç, bir e-posta gönderme -- model bu bilgiyi eğitilmiş bilgisinden üretmeye HİÇ çalışmaz. Bir tool'un çalıştırılmasını ister, ve modelin tahmini değil, *tool'un* çıktısı cevap olur.

Tool-Calling Döngüsü

Bir tool call tek bir adım değildir -- model ile onu barındıran uygulama arasında kısa bir döngüdür:

1. Uygulama, modele bir prompt'un yanı sıra kullanılabilir tool'ların bir listesini gönderir (isimler, açıklamalar, ve her birinin kabul ettiği argümanlar).
2. Model, yalnızca konuşmanın metnine ve tool açıklamalarına dayanarak, cevap vermenin bir tool gerektirip gerektirmediğine karar verir. Gerektirmiyorsa, normal şekilde yanıt verir.
3. Bir tool gerekiyorsa, model normal bir cevap yerine yapılandırılmış bir tool call üretir: hangi tool, ve hangi argümanlarla.
4. **Uygulama** (model değil) gerçek fonksiyonu çalıştırır -- bir API çağırır, bir veritabanı sorgular, kod çalıştırır -- ve sonucunu yakalar.
5. Bu sonuç, yeni bir girdi parçası olarak modelin context'ine geri eklenir, ve model tekrar çağrılır -- artık tool'un çıktısını gerçek cevabını yazmak için kullanabilir, ya da bir sonuç bir başka tool gerektiriyorsa başka bir tool call isteyebilir.

Bu döngü nihai bir cevap üretilmeden önce birden fazla kez çalışabilir, ve her tur modelin context'inden daha fazlasını tüketir (bkz. "Token'lar ve Context Window'lar"). Kritik olan: model bu döngüde hiçbir zaman hiçbir şeyi kendisi çalıştırmaz -- 4. adım her zaman modelin doğrudan erişimi olmayan, sıradan uygulama kodunda gerçekleşir.

Bir Tool Tanımlamak: Ad, Açıklama ve Şema

Bir tool modele üç parçayla tarif edilir, ve modelin bunu doğru kullanabilmesi tamamen bunların ne kadar iyi yazıldığına bağlıdır:

- **Name** -- `get_current_weather` ya da `search_orders` gibi kısa, belirsizliğe yer bırakmayan bir tanımlayıcı.

- **Description** -- tool'un ne yaptığının ve ne zaman kullanılması gerektiğinin sade bir dille açıklaması. Bu, modelin doğru zamanda doğru tool'u seçmesindeki en önemli sinyallerden biridir -- belirsiz bir açıklama ("veri getirir"), spesifik bir açıklamaya ("bir siparişin güncel kargo durumunu, sipariş ID'si verildiğinde arar") kıyasla modelin çok daha sık yanlış tahmin etmesine yol açar. (Tool seçimi yalnızca description'a bağlı değildir -- tool adı, parameter schema, konuşmanın context'i, ve modelin/uygulamanın kendi tasarımı da etkiler; ama description, geliştiricinin doğrudan kontrol ettiği ve en çok fark yaratan sinyallerden biridir.)
- **Parameter schema** -- tool'un hangi argümanları kabul ettiğinin, tiplerinin, ve hangilerinin zorunlu olduğunun yapılandırılmış bir tanımı (yaygın olarak JSON Schema). Model, hangi değerleri dolduracağına karar vermek için bu schema'yı kullanır, ve iyi tiplenmiş bir schema (ör. sipariş ID'sinin serbest metin değil bir string olması), hatalı biçimlendirilmiş call'ları belirgin şekilde azaltır.

Bunların hiçbiri, serbest metin üretiminde olduğu gibi model tarafında bir tahmin yürütme değildir -- model, verilen bir schema'yla eşleşen tool call'lar üretmek için özel olarak eğitilmiştir, ve pratikte schema kalitesinin bu kadar önemli olmasının nedeni de budur.

Tool Use ve Agent Karşılaştırması

Tek bir tool call -- havayı al, cevabı döndür -- bu kursun ilerleyen bölümlerinde **agent** olarak adlandırdığı şey HENÜZ DEĞİLDİR. Tool calling, agent sistemlerinin temel mekanizmalarından biridir; agent ise bir hedef doğrultusunda birden fazla adımı planlayabilen, hangi tool'ları ne sırayla kullanacağını kendi başına seçebilen, ve yukarıda anlatılan döngüyü, her adıma bir insan karar vermeden, belirli ölçüde otonom şekilde sürdürebilen daha geniş bir *sistemdir*. Her agent tool use'a dayanır, ama bunun tersi doğru değildir -- sıradan bir konuşma içinde bir tool'u bir kez kullanmak, tek başına bir agent DEĞİLDİR -- bu kursun ilerleyen "AI Agents" kategorisi, bir sistemin bu adı hak etmesi için ayrıca neyin doğru olması gerektiğini ele alıyor.

Best Practices

- Tool açıklamalarını bir değişkeni isimlendirir gibi değil, tool'u yeni bir takım arkadaşına anlatır gibi yazın -- belirsiz açıklamalar, modelin yanlış tool'u çağırmasının en yaygın nedenidir (bkz. "Bir Tool Tanımlamak: Ad, Açıklama ve Şema").
- Bir tool'un parameter schema'sını, gerçek fonksiyonun izin verdiği kadar dar ve tipli tutun -- üç geçerli değerden oluşan bir enum, modele kısıtsız bir serbest metin alanından çok daha az geçersiz call üretme alanı bırakır.
- Tek bir call için değil, döngü için tasarlayın (bkz. "Tool-Calling Döngüsü") -- bir tool'un sonucu başka bir tool call'ı tetikleyebilir, bu yüzden bir tool'un çıktısı, ham bir veri dökümü değil, modelin makul şekilde akıl yürütebileceği bir şey olmalıdır.

Yaygın Hatalar

- **Modelin tool'u kendisinin çalıştırdığını varsaymak.** "Tool-Calling Döngüsü"nün anlattığı gibi, model yalnızca bir istek üretir -- kodu gerçekten çalıştıran ve bunu güvenli şekilde yapmaktan sorumlu olan her zaman etrafındaki uygulamadır.
- **Tek satırlık bir tool açıklaması yazıp güvenilir bir seçim beklemek.** "Bir Tool Tanımlamak: Ad, Açıklama ve Şema" açıklama kalitesinin, modelin tool'lar arasında seçim yaparken kullandığı ana sinyal olduğunu gösterdi -- yetersiz tanımlanmış bir açıklama, yanlış tool'un çağrılmasının en yaygın gerçek dünya nedenidir.
- **Bir tool'un herhangi bir şekilde kullanımına "agent" demek.** "Tool Use ve Agent Karşılaştırması" bölümünün açıkladığı gibi, tool use mekanizmadır; agent ise onunla kurulmuş, bu kursun ilerleyen bölümlerinde ele alınan belirli bir tür sistemdir.

Özet, Cheat Sheet ve Terimler Sözlüğü

Özet

- Tool use (function calling), bir modelin kendisi dışında gerçek kod çalıştırılmasını istemesine izin vererek, "LLM Yetenekleri ve Sınırlamaları"nda anlatılan knowledge cutoff ve yerleşik doğrulama eksikliğinin bıraktığı yapısal açığı kapatır.
- Model hiçbir zaman hiçbir şeyi çalıştırmaz -- yapılandırılmış bir istek üretir; barındıran uygulama gerçek fonksiyonu çalıştırır ve sonucu döndürür.
- Tool-calling loop birden fazla tur çalışabilir: tool'larla prompt -> model karar verir -> uygulama çalıştırır -> sonuç context'e geri eklenir -> model devam eder ya da cevap verir.
- Bir tool, adı, açıklaması ve parameter schema'sıyla tanımlanır -- açıklama kalitesi doğru tool seçimindeki en önemli sinyallerden biridir.
- Tool calling, agent sistemlerinin temel mekanizmalarından biridir; agent bir hedef doğrultusunda birden fazla adımı planlayıp tool seçebilen daha geniş bir sistemdir -- tek bir tool call tek başına bir agent değildir.

Cheat Sheet

- Tool use / function calling = model bir fonksiyon call'ı ister; onu model değil, uygulama çalıştırır.
- Loop = prompt+tool'lar -> model karar verir -> uygulama çalıştırır -> sonuç context'e döner -> model devam eder.
- Tool tanımı = ad + açıklama + parameter schema; açıklama kalitesi doğru seçimi güçlü şekilde etkiler.
- Tool use != agent. Agent, hedef doğrultusunda planlayan, tool seçip kullanan, döngüyü belirli ölçüde otonom sürdüren daha geniş bir sistemdir.

Terimler Sözlüğü

- **Tool use / function calling:** bir LLM'in dışarıdaki bir fonksiyonun çalıştırılmasını istemesine ve sonucu yanıtında kullanmasına izin veren mekanizma.
- **Tool-calling loop:** modelin bir tool call istemesi, uygulamanın onu çalıştırması, ve sonucun context'e geri beslenmesinin tekrar eden döngüsü.
- **Parameter schema:** bir tool'un kabul ettiği argümanların yapılandırılmış tanımı (yaygın olarak JSON Schema), model tarafından geçerli call'lar oluşturmak için kullanılır.
- **Agent:** tool use üzerine kurulu, bir hedef doğrultusunda birden fazla adımı planlayabilen, tool'ları seçip kullanabilen, ve çalışma döngüsünü belirli ölçüde otonom sürdürebilen, bu kursun ilerleyen bölümlerinde ele alınan daha geniş bir sistem.