Prompting and Prompt Engineering

Prompt yapısı (system/user/assistant rolleri), zero-shot ve few-shot prompting, etkili prompt yazma pratikleri, ve chain-of-thought gibi yaygın prompting tekniklerine giriş.

Orta 20 dk

Son güncelleme: 6 Ekim 2026
Çevrimiçi oku: https://www.learnforgex.com/tr/topics/prompting-and-prompt-engineering

---

Prompting and Prompt Engineering

Son iki ders mekanizmayı kurdu: bir LLM'in inference zamanındaki davranışı tamamen context'i tarafından şekillendirilir ("Large Language Model'ler Nasıl Çalışır?"taki "Context'e İlk Bakış"a bakın), ve o context'in gerçek, sonlu sınırları vardır ("Token'lar ve Context Window'lar"a bakın). Bu ders, güvenilir, yararlı bir yanıt almak için o context'i bilinçli olarak inşa etme pratik becerisini kapsıyor -- yaygın olarak **prompting** denir, ve sistematik biçimde yapıldığında **prompt engineering** denir. Bir prompt, bir modelin context'ine yerleştirilen metinden başka bir şey olmadığı için, buradaki her şey aslında "In-Context Learning: Bir Model Ona Verdiğinizi Nasıl Kullanır?"ın doğrudan, pratik bir uygulamasıdır -- öğrenilecek yeni bir mekanizma yok, yalnızca zaten bildiğiniz mekanizmayı kullanmanın daha bilinçli bir yolu.

Prompt Nedir?

Bir **prompt,** bir yanıt üretmesi için bir LLM'e verdiğiniz metindir -- bir talimat, bir soru, dönüştürülecek bir metin parçası, ya da bunların bir kombinasyonu. "Context'e İlk Bakış"tan, context'in bir LLM'in çıktısını inference zamanında şekillendirmenin *tek* kanalı olduğunu hatırlayın; bir prompt, basitçe, o context'in sizin -- modeli kullanan kişinin ya da sistemin -- doğrudan yazdığı ve kontrol ettiği kısmıdır (bir sistemin ayrıca ekleyebileceği arka plan bilgisinin aksine, bunu bu kursun ilerleyen "Tool'lar ve MCP" kategorisi kapsıyor). İyi yazılmış bir prompt sihirli bir ifade değildir -- görevin net, eksiksiz bir spesifikasyonudur, modelin in-context learning'ine ("Large Language Model'ler Nasıl Çalışır?"a bakın) üzerinde çalışacak yeterli malzemeyi verecek şekilde yazılır.

Bir Prompt'un Yapısı: Rol'ler

Çoğu modern LLM etkileşimi tek, farklılaşmamış bir metin bloğu değildir -- modelin birbirinden ayırt edebilmesi için etiketlenmiş, farklı **rol'lere** yapılandırılmıştır:

- **System prompt (sistem prompt'u):** modelin tüm sohbet boyunca genel davranışını, persona'sını ya da kısıtlamalarını belirleyen talimatlar -- örneğin, "Yalnızca yemek pişirme hakkındaki soruları yanıtlayan yardımcı bir asistansın." Genellikle uygulama tarafından bir kez ayarlanır, son kullanıcı tarafından değil.
- **User prompt (kullanıcı prompt'u):** belirli bir turda modelle etkileşimde olan kişinin gerçek isteği ya da sorusu.
- **Assistant (model) yanıtı:** modelin kendi önceki yanıtları, çok turlu bir sohbette, bir sonraki turun context'inin bir parçası haline gelir ("In-Context Learning: Bir Model Ona Verdiğinizi Nasıl Kullanır?"ın doğrudan bir sonucu -- model, kendi önceki çıktısını sohbet geçmişinin bir parçası olarak yeniden okur).

Bu rolleri ayırmak, bir sistemin davranışı bir kez (system) kurmasına izin verir, ki bu davranış birçok farklı kullanıcı isteği (user) boyunca kalıcı olur -- bu, aynı talimatları her tek kullanıcı mesajının içinde tekrarlamaya çalışmaktan çok daha güvenilirdir.

Zero-Shot, Few-Shot ve Prompt'larda Örnekler

Bir **zero-shot** prompt, modelden yalnızca talimatlarla, örnek olmadan bir görevi gerçekleştirmesini ister -- "Bu cümleyi Fransızcaya çevir." Bir **few-shot** prompt, gerçek istekten önce görevin az sayıda çözülmüş örneğini içerir, modelin in-context learning'inin istenen örüntüyü, formatı ya da stili bir tarifinden değil doğrudan bu örneklerden almasını sağlar. Few-shot prompting, istenen çıktının kelimelerle tam olarak tarif etmesi zor ama göstermesi kolay belirli bir formatı ya da stili olduğunda en çok yardımcı olma eğilimindedir -- örneğin, geri istediğiniz tam JSON yapısının iki ya da üç örneğini göstermek, o yapıyı bir talimat paragrafında tarif etmekten genellikle daha güvenilirdir.

> Bir zero-shot prompt sürekli yanlış formatta ya da stilde çıktı üretiyorsa, gitgide daha uzun talimatlar yazmadan önce birkaç iyi seçilmiş örneğe başvurun -- göstermek, tam olarak in-context learning'in gerçekte nasıl çalıştığı yüzünden, genellikle tarif etmekten daha güvenilirdir.

Etkili Prompt'lar Yazmak

Birkaç somut pratik, bir modelin gerçekten istediğinizi ne kadar güvenilir biçimde yaptığını tutarlı olarak iyileştirir:

- **Görev ve istenen çıktı hakkında belirli olun.** "Bunu özetle," "Bunu, finansal rakamlara odaklanarak tam olarak üç madde işaretinde özetle"den çok daha tutarsız bir sonuç üretme olasılığı taşır.
- **Modele gerçekten ihtiyacı olan bilgiyi verin.** "Large Language Model'ler Nasıl Çalışır?"tan, pretraining'de ya da context'te olmayan hiçbir şeyin model için basitçe mevcut olmadığını hatırlayın -- bir görev belirli olgulara bağlıysa, onları dahil edin, modelin onları zaten bildiğini varsaymayın.

- **Belirli bir yapı önemli olduğunda çıktı formatını açıkça belirtin** (bir liste, düz yazıda bir tablo tarifi -- bu platform markdown tablolarından kaçındığı için --, belirli bir JSON şekli) -- alt akış kodunuz buna bağlıysa formatı şansa bırakmayın.
- **Karmaşık bir görevi, tek seferde karmaşık, çok parçalı bir sonuç istemek yerine, prompt'un kendi içinde daha küçük, açık adımlara bölün** -- bu, tek büyük, belirsiz bir talimattan daha güvenilir sonuçlar üretme eğilimindedir.

Yinelemeli Bir Süreç Olarak Prompt Engineering

Prompt engineering, ilk taslağı nihai sayma yerine prompt'ları sistematik olarak yazma, test etme ve iyileştirme pratiğidir. Pratikte bu şöyle görünür: bir prompt yazın, onu birkaç gerçekçi girdiye karşı çalıştırın, çıktının nerede yanlış gittiğine yakından bakın, ve prompt'u o belirli başarısızlığı ele alacak şekilde revize edin -- sonra tekrarlayın. Bu, tek seferlik yazmaktan çok, hata ayıklamaya ya da yinelemeli yazılım geliştirmeye daha yakındır, ve bu şekilde ele almaya değer: prompt versiyonları arasında neyin ve neden değiştiğini, inşa ettiğiniz herhangi başka bir sistem parçasındaki değişiklikleri takip ettiğiniz gibi takip edin.

Yaygın Prompting Teknikleri

Birkaç isimlendirilmiş örüntü, isimleriyle bilinmeye değecek kadar sık karşımıza çıkar:

- **Chain-of-thought prompting:** modelden son cevabı vermeden önce adım adım akıl yürütmesini istemek (örneğin, "bunu adım adım düşün, sonra son cevabını ver"), bu, birkaç mantıksal adım gerektiren görevlerde doğruluğu iyileştirme eğilimindedir -- esasen modele, son cevabını dayandırması için ek context olarak kendi akıl yürütmesinden daha fazlasını vermektir ("In-Context Learning: Bir Model Ona Verdiğinizi Nasıl Kullanır?"a bakın).
- **Role ya da persona prompting:** modelden belirli bir tür uzmanmış gibi yanıt vermesini istemek ("deneyimli bir güvenlik mühendisi olarak, bu kodu gözden geçir") -- bu, yanıtın *stilini ve odağını*, modelin eğitim verisinde o rolle ilişkilendirilen örüntülere doğru kaydırır, ama modele zaten sahip olmadığı hiçbir yeni bilgi vermez.
- **Kısıtlamaları açıkça sağlamak:** uzunluk sınırlarını, kaçınılacak şeyleri ya da gerekli öğeleri, modelin bunları çıkarsamasını ummak yerine doğrudan prompt'ta belirtmek -- context'te belirtilen kısıtlamalar, örtük bırakılan kısıtlamalardan çok daha güvenilirdir.

Best Practices

- Prompt yazmayı tek seferlik bir görev değil, yinelemeli bir süreç olarak ele alın ("Yinelemeli Bir Süreç Olarak Prompt Engineering"e bakın) -- yalnızca işe yarayan o tek örneğe karşı değil, gerçekçi girdilere karşı test edin.
- Modelin eksik olguları çıkarsayabileceğini varsaymak yerine ihtiyacı olan her şeyi doğrudan prompt'ta verin ("Etkili Prompt'lar Yazmak"a bakın) -- inference zamanında mevcut tek kanalın context olduğunu hatırlayın.

- Bir format ya da stil tam olarak tarif etmesi zor ama göstermesi kolay olduğunda few-shot örneklere başvurun ("Zero-Shot, Few-Shot ve Prompt'larda Örnekler"e bakın).

Yaygın Hatalar

- **Daha uzun, daha ayrıntılı bir prompt'un her zaman daha iyi bir prompt olduğunu varsaymak.** Kesinlik ve ilgi, uzunluktan çok daha önemlidir -- odaksız bir prompt, gerçek talimatı ilgisiz ayrıntının altına gömebilir, ve her ek token hâlâ para ve zamana mal olur ("Token'lar ve Context Window'lar"ı hatırlayın).
- **Bir ya da iki örnekle gösterilebilecek bir formatı tarif etmek için bir paragraf talimat yazmak.** "Zero-Shot, Few-Shot ve Prompt'larda Örnekler"in açıkladığı gibi, göstermek genellikle tarif etmekten daha güvenilirdir.
- **Role ya da persona prompting'i gerçekten yeni bilgi vermekle karıştırmak.** "Yaygın Prompting Teknikleri"nin belirttiği gibi, bir modelden "bir uzman olarak" yanıt vermesini istemek tonu ve stili değiştirir, ona mevcut olan altta yatan olguları değil -- hâlâ yalnızca pretraining'de ya da context'te olanı bilir.

Özet, Cheat Sheet ve Terimler Sözlüğü

Özet

- Bir **prompt,** bir kullanıcının ya da sistemin bir LLM'in yanıtını şekillendirmek için yazdığı metindir -- in-context learning'in doğrudan, pratik bir uygulaması.
- Prompt'lar yaygın olarak **rol'lere** yapılandırılır: system (genel davranış), user (istek), ve assistant (çok turlu sohbetlerde context'in bir parçası haline gelen önceki yanıtlar).
- **Zero-shot** prompt'lar yalnızca talimat verir; **few-shot** prompt'lar çözülmüş örnekler içerir, ki bunlar genellikle formatı ve stili tariflerden daha güvenilir biçimde iletir.
- Etkili prompt'lar belirlidir, ihtiyaç duyulan tüm bilgiyi sağlar, çıktı formatını açıkça belirtir, ve karmaşık görevleri açık adımlara böler.
- **Prompt engineering,** prompt yazmayı tek seferlik bir taslak yerine yinelemeli, test-et-ve-iyileştir bir süreç olarak ele alır.
- Chain-of-thought prompting, role/persona prompting, ve açık kısıtlamalar, model çıktısını şekillendirmek için yaygın isimlendirilmiş tekniklerdir.

Cheat Sheet

- Prompt = bir modelin yanıtını şekillendiren, doğrudan context'e yerleştirilen metin.
- System prompt = genel davranış. User prompt = istek. Assistant = önceki yanıtlar (sonraki turlarda context).
- Zero-shot = yalnızca talimat. Few-shot = talimat + çözülmüş örnekler.
- Chain-of-thought = modelden önce adım adım akıl yürütmesini isteyin.
- Prompt engineering = yinele: yaz, test et, başarısızlıkları gözlemle, revize et.

Terimler Sözlüğü

- **Prompt:** bir yanıt üretmesi için bir LLM'e verilen metin.
- **Prompt engineering:** prompt'ları sistematik olarak yazma, test etme ve iyileştirme pratiği.
- **System prompt / User prompt / Assistant yanıtı:** bir LLM sohbetini yapılandırmak için kullanılan üç yaygın rol.
- **Zero-shot prompting:** bir modele hiçbir çözülmüş örnek olmadan talimatlar vermek.
- **Few-shot prompting:** istenen örüntüyü, formatı ya da stili göstermek için bir prompt'a az sayıda çözülmüş örnek dahil etmek.
- **Chain-of-thought prompting:** bir modelden son bir cevap üretmeden önce adım adım akıl yürütmesini istemek.
- **Role / persona prompting:** bir modelden belirli bir tür uzman ya da karaktermiş gibi yanıt vermesini istemek.