Yapay zekâ bir programdaki hatayı nasıl bulabilir?



www.yzokul.com.tr

Yapay zekâ bir programdaki hatayı nasıl bulabilir?



Yapay zekâ, kodunu ve hata mesajını birçok programdan öğrendiği kalıplarla karşılaştırarak ve ardından olası sorunları önererek bir programdaki hatayı bulabilir. Ona kodu ve mesajı verirsin, o da hatanın muhtemelen nerede olduğunu tahmin eder. Kısa yanıt, yapay zekânın yaygın hataları saptamak için öğrendiği kalıpları kullandığıdır. Programını çalıştırmaz; yalnızca okur ve karşılaştırır.



Program hatası, programın çalışmasını durduran ya da yanlış sonuç vermesine yol açan şeydir. Bir hata oluştuğunda program genellikle durur ve ekranda kırmızı bir mesaj gösterir. Örneğin, iki sayıyı toplayan bir program, ona sayı yerine bir sözcük verirsen çökebilir. Hata, doğru sonucu engelleyen sorundur.



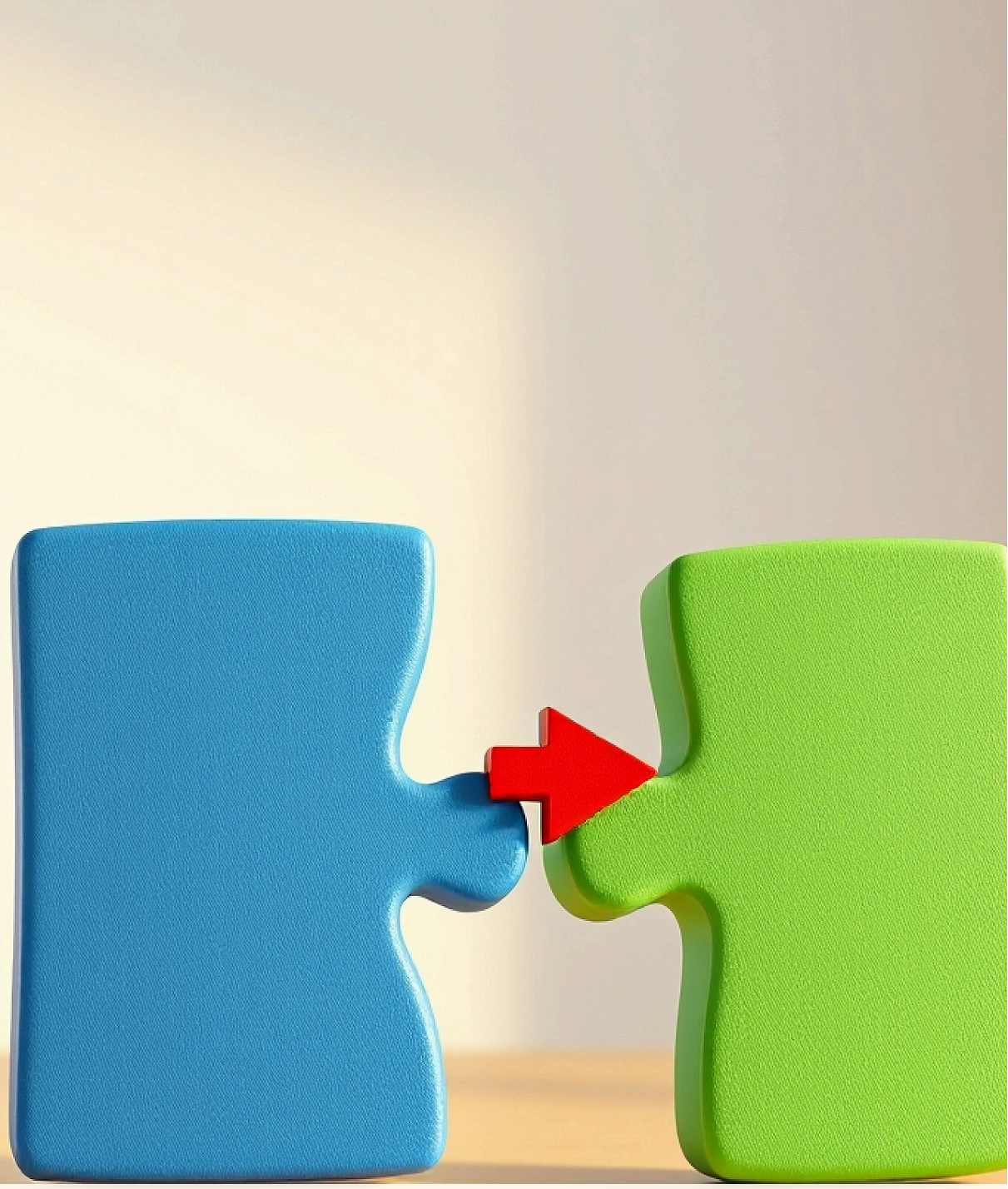
Hata mesajı, bir program başarısız olduğunda yazdırdığı kısa bir metindir. Genellikle neyin yanlış gittiğini ve sorunun kodun neresinde olduğunu söyler. Örneğin, bir adın tanımlanmadığını ya da bir liste dizininin aralık dışında olduğunu söyleyebilir. Bu mesaj, nedeni bulmak için hem senin hem de yapay zekânın okuyabileceği bir ipucudur.



Yapay zekâ, sen onu kullanmadan önce çok sayıda programı ve bunların hatalarını okumuştur. Bundan, hangi hataların sıklıkla hangi mesajlara yol açtığı gibi yaygın kalıpları öğrenmiştir. Örneğin, belirli bir mesajın sıklıkla bir değişken adı yanlış yazıldığında ortaya çıktığını öğrenmiştir. Bu öğrenilmiş kalıplar, kodunda neyin yanlış olduğunu tahmin etmesine yardımcı olur.



Yapay zekâ aynı anda iki şeye bakar: kodun ve hata mesajı. Bunları öğrendiği kalıplarla eşleştirir ve sorunun muhtemelen nerede olduğunu tahmin eder. Örneğin, mesaj bir adın tanımlanmadığını söylüyorsa, kodunda o adı arar. İki ipucunu birleştirerek hatanın olası yerini daraltır.



Yaygın bir kalıp yanlış değişken adıdır. Değişken, bir değeri tutan addır ve onu iki yerde farklı yazarsan program onu bulamaz. Yapay zekâ adların eşleşmediğini fark eder ve doğru yazımı önerir. Örneğin, bir satırda total, başka bir satırda totl yazarsan, uyuşmazlığa işaret eder.



Bir başka kalıp eksik kontroldür. Kod, kullanmanın güvenli olduğunu önce kontrol etmeden bir şeyi kullanır. Örneğin, sayının sıfır olmadığını kontrol etmeden bir sayıya bölmeye çalışabilir. Yapay zekâ, o adımdan önce bir kontrol eklemeyi önerir; örneğin bölmeden önce sayının sıfır olup olmadığını test etmeyi.



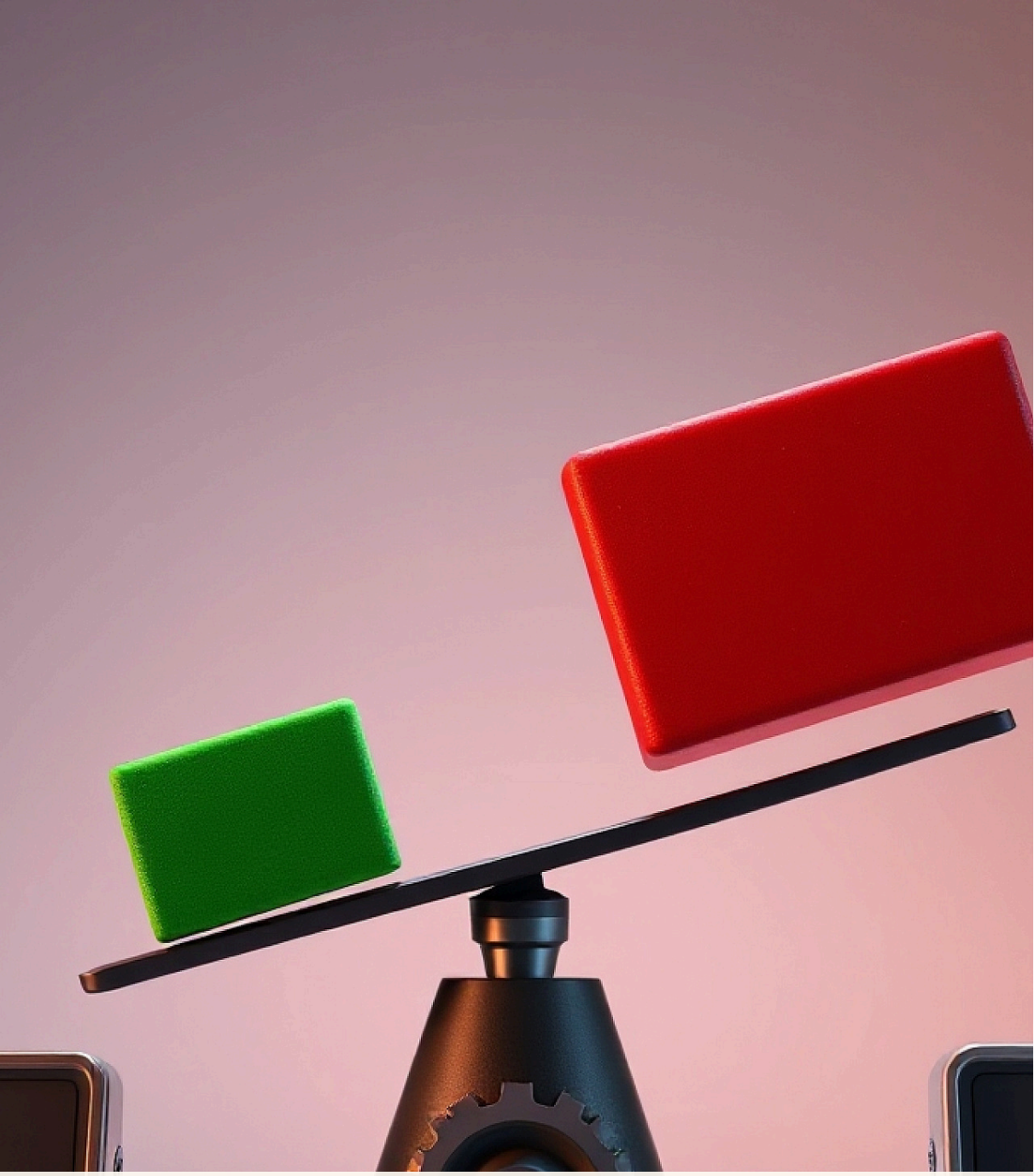
Üçüncü bir kalıp yanlış işlem sırasıdır. Adımlar yanlış sırada gerçekleşir, bu yüzden bir değer hazır olmadan kullanılır. Örneğin, bir sonucu hesaplamadan önce yazdırabilirsin. Yapay zekâ işleyecek sıraya işaret eder; örneğin önce hesaplayıp sonra yazdırmak gibi. Bu, değerin neden yanlış olduğunu görmene yardımcı olur.



Programın boş bir listenin ilk öğesini almaya çalıştığı küçük bir örneği ele alalım. Liste, öğelerden oluşan bir koleksiyondur ve boş bir listede hiçbir şey yoktur. Yapay zekâ, alınacak bir şey olmadığını ve bunun hataya yol açtığını açıklar. İlk öğeyi almadan önce listenin boş olmadığını kontrol etmeyi önerir.



Yapay zekâ yanılabilir çünkü yalnızca ona gösterdiğini görür, programın çalışırken yaptığı her şeyi değil. Programın yürütülmesini izlemez ya da gerçek verileri görmez. Eksik bilgi onu yanlış bir teşhise götürebilir. Örneğin, gerçek sorun başka yerde olduğu için aslında sorunsuz olan bir kod satırını suçlayabilir.



Gerçek programlar, yapay zekânın bilmediği girdilere, verilere ve ayarlara bağlıdır. Örneğin, bir program küçük sayılarla çalışıp çok büyük sayılarla başarısız olabilir. Bu gizli çalışma koşulları olmadan yapay zekânın neden hakkındaki tahmini isabetsiz olabilir. Gerçek sorunu ele almayan bir düzeltme önerebilir.



Yapay zekânın önerisini her zaman gerçek program üzerinde dene. Kodu önerildiği gibi değiştir, programı çalıştır ve hatanın gerçekten ortadan kalkıp kalkmadığına bak. Örneğin, yapay zekâ boş liste için bir kontrol eklemeyi söylüyorsa, ekle ve boş bir listeyle test et. Düzeltmenin işe yarayıp yaramadığını bilmenin tek yolu test etmektir.



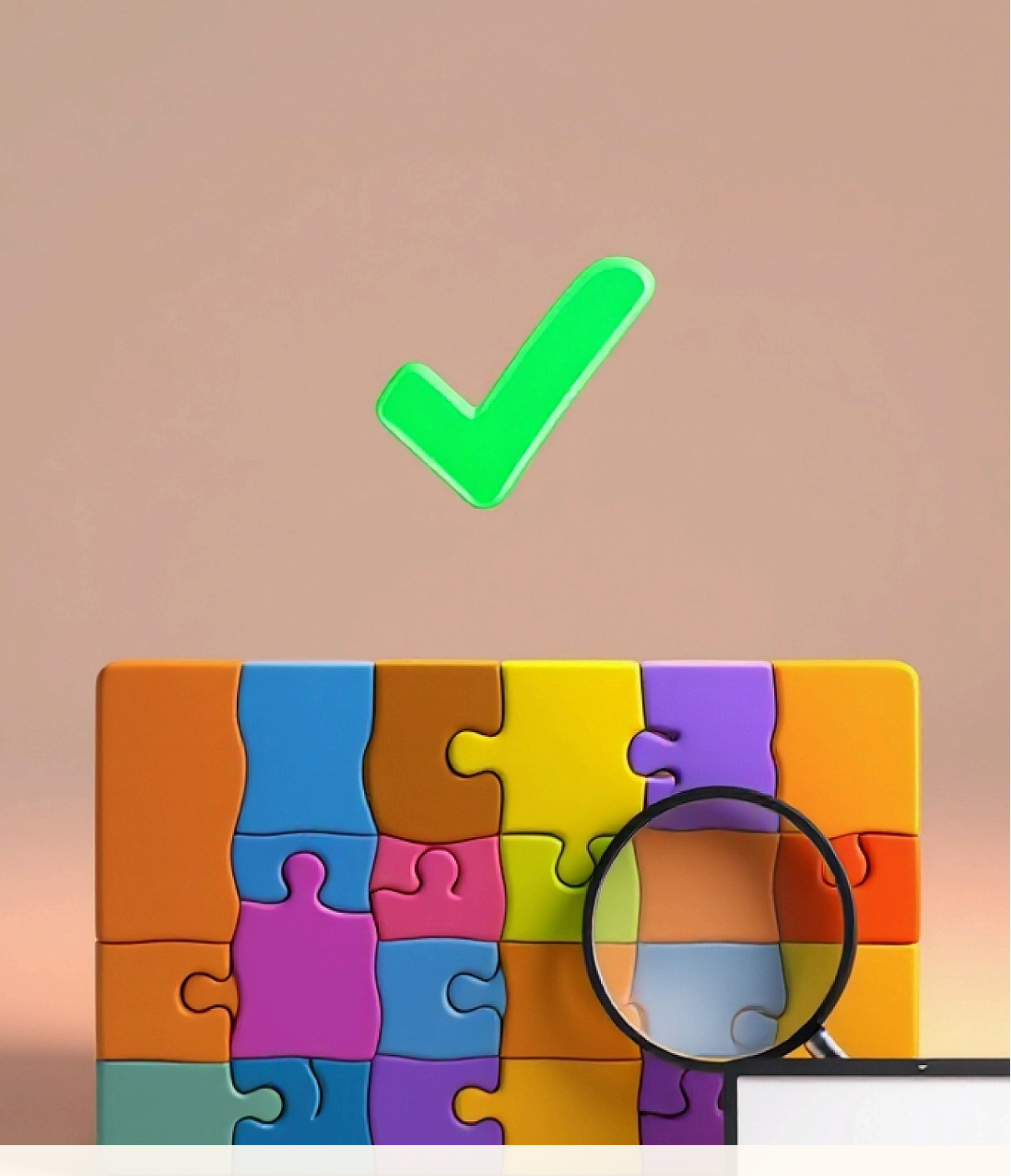
Yapay zekânın açık ve kendinden emin bir açıklaması yine de yanlış olabilir. Açıklama inandırıcı gelebilir, ama yalnızca kalıplara dayanan bir tahmindir. Düzeltmenin gerçekten işe yarayıp yaramadığını yalnızca gerçek programı çalıştırmak gösterir. Bir açıklamayı kanıt olarak kabul etme; değişikliği her zaman test et ve sonucu gözlemle.



Tipik bir acemi hatası, yapay zekânın yanıtını test etmeden kabul etmektir. İnandırıcı bir açıklama okur, kodu değiştirir ve düzeldiğini varsayarsın. Ama programı yeniden çalıştırdığında aynı hata ortaya çıkar. Görsel, düzeltme gerçek program üzerinde hiç test edilmediği için geri dönen hata mesajını gösterir.



Bir başka hata, yapay zekâya çok az bilgi vermektir. Kodun yalnızca bir kısmını gösterirsen ya da hata mesajını hiç göstermezsen, önerisi bir teşhis değil tahmin olur. Örneğin, tam hata mesajı olmadan hangi satırın başarısız olduğunu bilemez. Daha iyi bir tahmin için her zaman tam kodu ve tam hata mesajını sağla.



Yapay zekâ, kodunu ve hata mesajlarını birçok programdan öğrendiği kalıplarla eşleştirerek olası hataları bulur, ama yanılabilir. Yanlış değişken adı, eksik kontrol ya da yanlış işlem sırası gibi sorunlar önerebilir. Ona güvenmeden önce önerisini her zaman gerçek program üzerinde test et. Tek başına inandırıcı bir açıklama yeterli değildir.