

Proje Raporu: ir_explain Kütüphanesi ve Açıklanabilir IR

1. Genel Bakış

ir_explain, modern Bilgi Erişimi (Information Retrieval - IR) sistemlerinin, özellikle de "kara kutu" olarak adlandırılan derin öğrenme temelli sıralama modellerinin (Neural Rankers) kararlarını analiz etmek ve açıklamak için geliştirilmiş açık kaynaklı bir Python kütüphanesidir.

2. Temel Metodoloji: Üç Seviyeli Açıklama

Sistem, arama sonuçlarını üç farklı perspektiften inceler:

- Noktasal (Pointwise):** Münferit bir belgenin neden o sırada olduğunu, içindeki hangi terimlerin (pozitif veya negatif) sonuca ne kadar etki ettiğini analiz eder.
- İkili (Pairwise):** İki belgeyi kıyaslar. "A belgesi neden B'den daha üstte?" sorusuna mantıksal kurallar (aksiyonlar) çerçevesinde yanıt arar.
- Liste Bazlı (Listwise):** Tüm sonuç listesini bir bütün olarak ele alır. Karmaşık modeli, daha basit ve anlaşılır bir modelle (taklitçi model) simüle ederek genel bir "sorgu niyeti" özeti çıkarır.

3. Abraham Wald ve "Survivorship Bias" Analojisi

Notlarında bu kütüphaneyi anlamlandırmak için kullanacağın en güçlü kavram budur. ir_explain, aslında yapay zekanın **"Hayatta Kalma Yanlılığı"**ni (Survivorship Bias) kırmaya çalışan bir denetcidir:

Kavram	Abraham Wald'ın Uçağı	ir_explain Analizi
Gözlem Birimi	Üsse geri dönebilen (başarılı) uçaklar.	Arama sonucunda üst sıralara çıkan (başarılı) belgeler.
Kurşun Delikleri	Uçağın hasar almasına rağmen uçabildiği yerler.	Belgedeki "olsa da olur olmasa da olur" kelimeler (Zayıf etkili terimler).
Görünmez Hasar	Motor veya kokpit gibi vurulduğunda uçağı düşüren yerler.	Belgeyi sıralamadan tamamen düşürecek veya yanlış anlamlandırılacak kritik terimler.

Zırhlama Kararı	Deliklerin olmadığı yerleri güçlendirmek.	Modelin aslında neye odaklanması gerektiğini (asıl niyeti) tespit etmek.
-----------------	---	--

Not: Makaledeki "sağlamlık" (robustness) deneyi şunu kanıtlar: Mevcut sistemler bazen sadece uçağın üzerindeki rastgele deliklere (alakasız kelime'lere) odaklanıp uçağın neden havada kaldığını (belgenin neden seçildiğini) yanlış yorumlayabiliyor.

4. Kullanım Senaryoları ve Faydaları

- Model Hata Ayıklama (Debugging):** Yazılımcılar, modelin "motoruna" isabet eden hatalı mantıkları bu araçla tespit edebilir.
- RAG (Geri Getirme ile Artırılmış Nesil) Denetimi:** Yapay zekanın bir cevabı uydurup uydurmadığını (halüsinasyon), cevabı gerçekten okuduğu belgelerdeki hangi terimlere dayandırdığını (atıf) bularak kontrol eder.
- Akademik Kıyaslama:** Farklı açıklama yöntemlerini tek bir platformda toplayarak hangisinin daha "insani ve sezgisel" sonuç verdiğiini ölçer.

5. Kritik Çıkarım (Self-Reflection)

Sadece başarılı sonuçlara (üstteki belgelere) bakarak bir modeli anlamaya çalışmak, "eve dönen uçakları zırhlamaya" benzer. `ir_explain`, hem başarılı belgelerdeki kritik noktaları bularak hem de ufacık değişikliklerle (perturbation) modelin nerede "düştüğünü" göstererek, sistemin gerçek çalışma mantığını ortaya çıkarmayı hedefler.

Abraham Wald'ın hikayesiyle `ir_explain` kütüphanesinin (ve genel olarak açıklanabilir yapay zekanın) yaptığı iş arasında muazzam bir felsefi benzerlik var.

1. "Kurşun Delikleri" vs. "Önemli Kelimeler"

Donanma subayları sadece eve dönen uçaklara (yani arama sonucunda en üstte çıkan belgelere) bakıyordu.

- İstatistikçiler:** "Uçak kanattan vurulmuş ama gelmiş, demek ki kanat zırhlanmasa da olur."
- ir_explain (Noktasal/Pointwise):** Belgedeki kelimeleri tek tek çıkarıp deniyor. "Bak, bu kelimeyi silince uçak düşmedi (belge sıralamada kaybolmadı), demek ki sistem bu kelimeye bakmıyor." diyorlar. Yani sistemin "**"kurşun deliklerini" (hasara dirençli yerleri)** haritalandırıyorlar.

2. "Düşen Uçaklar" ve Görünmez Tehlike

Senin dediğin gibi, asıl mesele uçağın neresinden hasar aldığında **eve dönemediği**.

Makaledeki "**sağlamlık**" (**robustness**) testi tam olarak bu: "Ben kelimeyi ufacık değiştirdim (başka bir yerden vurdum) ve uçak (açıklama) bir anda düştü."

Sistem şuna bakıyor:

"Acaba sistem bu belgeyi içindeki 'X' kelimesi olduğu için mi getirdi (hayatta kalan veriye odaklanma), yoksa içinde 'Y' kelimesi **olmadığı** için mi getirdi?"

3. "Survior Bias"ı Kırma Çalışıyorlar

Açıklama yöntemleri aslında o "görünmeyen" yerleri bulmaya çalışıyor:

- **Aksiyomatik Yöntem:** Bu yöntem Wald gibi davranıştır. "Sadece bu belgenin hayatta kalmasına bakma" der. "Matematiksel olarak bir belgenin eve donebilmesi için (üst sırada çıkışması için) şu mantık kurallarına (zırha) sahip olması gereklidir" der.
- **Negatif Katkı:** Makalede gördüğün o "negatif ağırlıklı kelimeler" aslında uçağın üzerine gelen mermilerdir. Sistem şunu der: "*Bu kelime uçağın (belgenin) motoruna gelmiş, eğer bu kelime biraz daha ağır olsaydı bu belge sıralamada asla görünmeyecekti.*"

Neden "Hata Yapıyorlar"?

Makalede yazarların dert yandığı konu da senin örneğinle ortaşuyor: Mevcut açıklama modelleri hala uçağın üzerindeki her deliğe bakıp "Aman tanım, uçak delik deşik olmuş" diyor (yani her kelimeyi önemli sanıyor). Ama o deliklerin hangisinin hayatı (motoru vuran), hangisinin önemsiz (kanat ucu) olduğunu ayırt etmekte hala zorlanıyorlar.

ir_explain, işte bu "neresi zırhlanmalı?" sorusuna (yani modelin hangi özelliğinin gerçekten sonucu değiştirdiğine) cevap vermek için bilim insanlarına bir laboratuvar sunuyor.

Bu bakış açınl, makaledeki "**Hata Ayıklama**" (**Debugging**) kısmını daha iyi anlayabilirsin: Yazılımcı, uçağın (modelin) motoruna gelen ama fark edilmeyen o "kritik vuruşu" bu araçlarla bulmaya çalışıyor.

Ne: Cevap vermeden önce doğal olarak düşünen önceden eğitilmiş modeller (örneğin OpenAI gpt-5, Claude 4.5 Sonnet, Gemini 2.0 Flash Thinking, DeepSeek-R1).