Projeyi Calisma Mantigi ve Teknik Detaylar

Bu dokuman, sistemin nasil calistigini, kullanilan teknolojileri ve "neden" kullanildiklarini hic bilmeyen birine anlatir gibi sade ve detayli bir sekilde aciklar.
---

1. Temel Kavramlar: Ne Yapiyoruz?

Amac: Bir bankanin musteri sozlesmeleri (ornegin Kredi Karti Sozlesmesi) ile devletin belirledigi kurallar (Teblig/Mevzuat) arasindaki uyumu denetlemek.
Sorun: Sozlesmeler cok uzun (200+ sayfa) ve kurallar cok karmasik (Teblig). Insanlar yavas okur ve gozden kacirir.
Cozum: Bu dokumanlari bilgisayara okutup, yapay zekaya denetletmek.
---

2. Adim Adim Calisma Sureci (Pipeline)

Sistemin "Baslat" dugmesine bastiginizda arka planda su islemler sirasiyla gerceklesir:

A. Veri Hazirligi ve "Chunking" (Parcalama)

Bilgisayarlar (ve Yapay Zeka modelleri), 200 sayfalik bir kitabi tek seferde hafizasinda tutamaz. Tipki bir insanin kitabi sayfa sayfa veya paragraf paragraf okumasi gerektigi gibi, biz de dokumanlari kucuk parcalara boleriz.
* **Chunking Nedir?** Uzun bir metni, anlam butunlugunu bozmadan kucuk parcalara (ornegin her biri bir sozlesme maddesi olacak sekilde) bolme islemidir.
* **Neden Yaptik?**
1. **Odaklanma:** Yapay zeka sadece ilgili maddeye odaklansin, dikkati dagilmasin.
2. **Arama Basarisi:** "Kart aidati" diye arattigimizda, 200 sayfalik dosya yerine sadece aidatla ilgili paragrafi bulmak isteriz.

B. Indeksleme: ChromaDB ve BM25

Parcaladigimiz bu kucuk metinleri (Chunk), daha sonra hizlica bulabilmek icin ozel bir kutuphaneye (veritabanina) kaydederiz. Burada iki farkli teknoloji kullaniyoruz:
1. **ChromaDB (Vektor Veritabani):**
* **Nedir?** Kelimelerin "anlamini" sayilara (vektorlere) cevirir.
* **Ornek:** Siz "ek masraf" diye ararsiniz, sistem bunun "ilave ucret" veya "komisyon" ile benzer anlama geldigini bilir ve onlari bulur. Kelime ayni olmasa bile *anlam* ayniysa bulur.
2. **BM25 (Anahtar Kelime Indeksi):**
* **Nedir?** Klasik "Google Aramasi" gibi calisir. Kelimelerin birebir kendisini arar.
* **Ornek:** "Madde 12/A" diye ararsak, vektor veritabani bunu "Madde 12/B" ile karistirabilir (sayilar birbirine benzerdir). Ancak BM25, tam olarak "12/A" yazan yeri bulur.

> **Teknik Detay: Turkce icin BM25 Stratejimiz (Neden Zemberek Kullanmadik?)**
> Turkce, sondan eklemeli bir dildir (orn: *Banka > Bankalar > Bankalarin*). Klasik arama motorlarinda "Banka" diye aratinca "Bankalarin" kelimesini bulmak icin *Stemming/Lemmatization* (Kelime kokune inme - Zemberek vb.) yapilmasi onerilir.
>
> Ancak biz bu projede **Basit Tokenizasyon** (sadece kucuk harfe cevirme) kullandik.
> **Neden?**
> 1. **Hiz (Latency):** Kok bulma islemi her sorguda sistemi yavaslatir. Hackathon'da hiz kritiktir.
> 2. **Hibrit Gucu:** BM25'in "eklerden dolayi kacirdigi" kelimeleri, zaten anlamdan yakalayan **Vektor Arama (ChromaDB)** tamamlamaktadir. Yani agir bir Turkce kutuphanesi kullanmadan, Hibrit Mimari sayesinde ayni basariyi cok daha hizli elde ettik.

C. Hibrit Arama (Hybrid Search) ve RRF

Banka sozlesmesindeki bir maddeyi denetlemek icin, o maddeyle ilgili Mevzuat (Teblig) kuralini bulmamiz gerekir.

* **Soru:** Banka sozlesmesindeki "Yillik uyelik bedeli 500 TL" maddesi icin hangi yasaya bakmaliyim?
* **Islem:** Sistem bu maddeyi hem ChromaDB'de (anlam) hem BM25'te (kelime) aratir.
* **Birlestirme (RRF - Reciprocal Rank Fusion):** Iki farkli arama sonucunu harmanlar. Hem anlam olarak en yakin, hem de kelime olarak en uyusan sonuclari ust siraya tasir. Boylece dogru yasayi bulma ihtimalimiz maksimuma cikar.

D. Karar Motoru (Decision Engine): Once Kural, Sonra Zeka

Dogru yasayi bulduk (Hibrit Arama tamamlandi). Simdi "Banka maddesi bu yasaya uygun mu?" sorusunu cevaplamaliyiz.
Bu motor, **her zaman ve kesinlikle sirasiyla** calisan 3 adimdan olusur:

1. **Adim: Kural Motoru (Rule Engine) - [Hesaplama & Uyari]**

* **Yapi:** Sadece basit Regex (Kelime Arama) degildir. **Regex + Python Mantigi** birlikte calisir.
* **Nasil Calisir?**

1. **Veri Cikarma (Regex):** Metin icindeki sayilari ("5 TL"), oranlari ("%10") ve anahtar kelimeleri ("Onaysiz") Regex ile cekip cikarir.
2. **Mantiksal Kiyas (Python):** Cikarilan bu sayilari matematiksel olarak karsilastirir (`if banka_ucreti > yasa_limiti`).

* **Ornek:** Yasa "EFT max 2 TL" derken Banka "5 TL" demisse, Regex "2" ve "5" sayilarini bulur, Python kodu "5 > 2" islemini yapip hatayi yakalar.
* **Kritik Nokta:** Kural motoru burada islemi bitirmez. Tespit ettigi hatayi bir "UYARI ETIKETI" (Warning Label) haline getirir ve bir sonraki asamaya (LLM'e) paslar.

2. **Adim: Buyuk Dil Modeli (LLM - Gemma) - [Akil Yurutme & Karar]**

* Yapay zeka devreye girer ve kendisine sunulan **TUM** verileri okur:
* Girdi 1: Banka Maddesi

* Girdi 2: Ilgili Yasa (Mevzuat)
* Girdi 3: **Kural Motorundan Gelen Uyari** ("Dikkat: Burada matematiksel limit asimi var!")
* **Nihai Karar:** LLM, bu uc veriyi birlestirir. Kural motorunun uyarisini dikkate alarak ve yasanin sozel yorumunu da ekleyerek kararini verir: "UYUMSUZ (NOT_OK)".

3. **Adim: Son Kontrol (Safety Override) - [Guvenlik Kilidi]**
* LLM kararini verdikten sonra, devreye son bir guvenlik kodu girer.
* **Neden?** Bazen LLM cok karmasik, sinsi ifadeleri (orn: "imzayla sigortayi kabul etmis sayilirsiniz") "Uyumlu" sanabilir.
* Kod, bu tip yasakli kelime kaliplarini (Regex) kontrol eder. Eger LLM "Uyumlu" dese bile, kod bu yasagi gorurse karari ezer (**Overrule**) ve sonucu "UYUMSUZ" olarak degistirir.

Ozet Siralama: Hibrit Arama -> Kural Uyarisi -> LLM Karari -> Son Kontrol (Override). Bu sira asla degismez.

> **Gorsel Sema Icin Oneri (Diagram Flow):**
> Eger bu akisi ciziyorsaniz, kutucuklari su sirayla baglamalisiniz:
> 1. `[Hibrit Arama Sonuclari]` --> ok --> `[Kural Motoru (Regex)]`
> 2. `[Kural Motoru]` --> (Cikti: Uyarilar) --> `[LLM (Gemma)]`
> 3. `[LLM]` --> (Cikti: Ham Karar) --> `[Son Kontrol (Override)]`
> 4. `[Son Kontrol]` --> `[Nihai Karar (JSON)]`

E. Raporlama ve Chatbot

Tum maddeler tek tek bu surecten gectikten sonra:

1. **Dashboard:** Sonuclar renkli grafiklerle ekrana yansitilir. Kirmizi (Risk), Yesil (Uyumlu).
2. **Otomatik Ozet:** Sistem sonuclari analiz eder ve yoneticiye "Toplam 5 risk var, en kritik olani EFT ucreti" seklinde bir ozet yazar. Bu ozet Yapay Zeka (LLM) tarafindan degil, onceden hazirlanmis akilli sablonlar (Template) ile, cikan istatistiklere gore anlik uretilir.
3. **Chatbot:** Tum analiz bittikten sonra, eger aklinizа takilan bir sey olursa Chatbot'a sorabilirsiniz. Chatbot, hem orijinal yasayi hem de bizim analiz sonuclarimizi bilir. "Neden EFT maddesine riskli dedin?" diye sorarsaniz, size dayanagini (Madde 12/3 ve hesapladigi tutar farki) aciklar.

Ozet: Neden 3 Farkli Teknoloji (Vektor, Kural, LLM) Kullaniyoruz?

* Tek basina **LLM** kullansaydik matematik hatasi yapabilirdi (LLM'ler sayi saymakta bazen zorlanir).
* Tek basina **Kural (Rule)** kullansaydik, "Musteri magdur edilemez" gibi yoruma acik cumleleri anlayamazdi.
* Tek basina **Vektor Arama** kullansaydik, sayilari (Madde 12 ile 13'u) karistirabilirdi.

Hepsini birlestirerek (Hybrid Pipeline), **hem matematiksel kesinligi, hem hukuksal yorumlama yetenegini hem de dogru yasaya ulasma hizini** tek bir potada erittik.