

Projeyi Calisma Mantigi ve Teknik Detaylar

Bu dokuman, sistemin nasıl çalıştığını, kullanılan teknolojileri ve "neden" kullanıldıklarını hiç bilmeyen birine anlatır gibi sade ve detaylı bir şekilde açıklar.

1. Temel Kavramlar: Ne Yapıyoruz?

Amac: Bir bankanın müşteri sözleşmeleri (örnegin Kredi Kartı Sözleşmesi) ile devletin belirlediği kurallar (Tebliğ/Mevzuat) arasındaki uyumu denetlemek.

Sorun: Sözleşmeler çok uzun (200+ sayfa) ve kurallar çok karmaşık (Tebliğ). İnsanlar yavaş okur ve gözden kaçırır.

Cözüm: Bu dokumanları bilgisayara okutup, yapay zeka ile denetletmek.

2. Adım Adım Çalışma Sureci (Pipeline)

Sistemin "Baslat" düğmesine bastığınızda arka planda şu işlemler sırasıyla gerçekleşir:

A. Veri Hazırlığı ve "Chunking" (Parçalama)

Bilgisayarlar (ve Yapay Zeka modelleri), 200 sayfalık bir kitabı tek seferde hafızasında tutamaz. Tipki bir insanın kitabı sayfa sayfa veya paragraf paragraf okuması gerektiği gibi, biz de dokumanları küçük parçalara böleriz.

- * **Chunking Nedir?** Uzun bir metni, anlam bütünlüğünü bozmadan küçük parçalara (örnegin her biri bir sözleşme maddesi olacak şekilde) böleme işlemidir.
- * **Neden Yaptık?**
- 1. **Odaklanma:** Yapay zeka sadece ilgili maddeye odaklı olmalıdır, dikkati dağılmaması gerekmektedir.
- 2. **Arama Başarısı:** "Kart aidati" diye arattığımızda, 200 sayfalık dosya yerine sadece aidatla ilgili paragrafi bulmak isteriz.

B. İndeksleme: ChromaDB ve BM25

Parçaladığımız bu küçük metinleri (Chunk), daha sonra hızla bulabilmek için özel bir kütüphaneye (veritabanına) kaydederiz. Burada iki farklı teknoloji kullanıyoruz:

1. **ChromaDB (Vektor Veritabanı):**
 - * **Nedir?** Kelimelerin "anlamını" sayılar (vektörlere) çevirir.
 - * **Ornek:** Siz "ek masraf" diye ararsınız, sistem bunun "ilave ücret" veya "komisyon" ile benzer anlamda geldiğini bilir ve onları bulur. Kelime aynı olmasa bile *anlam* aynıysa bulur.
2. **BM25 (Anahtar Kelime İndeksi):**
 - * **Nedir?** Klasik "Google Araması" gibi çalışır. Kelimelerin birebir kendisini arar.
 - * **Ornek:** "Madde 12/A" diye ararsak, vektor veritabanı bunu "Madde 12/B" ile karıştırabilir (sayilar birbirine benzerdir). Ancak BM25, tam olarak "12/A" yazan yeri bulur.

Projeyi Calisma Mantigi ve Teknik Detaylar

- > **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 önerilir.
- >
- > 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 (Tebliğ) kuralini bulmamiz gereklidir.

- * **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) arar.
- * **Birlestirme (RRF - Reciprocal Rank Fusion):** Iki farkli arama sonucunu harmanlar. Hem anlam olarak en yakin, hem de kelime olarak en uyusan sonuclar 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 & Uyarı]**
 - * **Yapi:** Sadece basit Regex (Kelime Arama) degildir. **Regex + Python Mantigi** birlikte calisir.
 - * **Nasil Calisir?**
 1. **Veri Cikarma (Regex):** Metin icindeki sayilar ("5 TL"), oranlari ("%10") ve anahtar kelimeleri ("Onaysiz") Regex ile cekip cikarir.
 2. **Mantiksal Kiyas (Python):** Cikarilan bu sayilar matematiksel olarak karsilastirir (`if banka_ureti > yasa_limiti`).
 - * **Ornek:** Yasa "EFT max 2 TL" derken Banka "5 TL" demisse, Regex "2" ve "5" sayilarini bulur, Python kodu "5 > 2" islemi 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

Projeyi Calisma Mantigi ve Teknik Detaylar

- * Girdi 2: Ilgili Yasa (Mevzuat)
 - * Girdi 3: **Kural Motorundan Gelen Uyarı** ("Dikkat: Burada matematiksel limit asımı var!")
 - * **Nihai Karar:** LLM, bu üç veriyi birleştirir. Kural motorunun uyarısını dikkate alarak ve yasanın sözel yorumunu da ekleyerek kararını verir: "UYUMSUZ (NOT_OK)".
 - 3. **Adım: Son Kontrol (Safety Override) - [Guvenlik Kilidi]**
 - * LLM kararını verdikten sonra, devreye son bir güvenlik kodu girer.
 - * **Neden?** Bazen LLM çok karmaşık, sinsi ifadeleri (örn: "imzayla sigortayı kabul etmiş sayılırsınız") "Uyumlu" sanabilir.
 - * Kod, bu tip yasaklı kelime kalıplarını (Regex) kontrol eder. Eğer LLM "Uyumlu" dese bile, kod bu yasığı gorurse kararı ezer (**Overrule**) ve sonucu "UYUMSUZ" olarak değiştirecektir.
- **Özet Sıralama:** Hibrit Arama -> Kural Uyarısı -> LLM Kararı -> Son Kontrol (Override). Bu sıra asla degişmez.
- > **Görsel Sema İçin Öneri (Diagram Flow):**
- > Eğer bu akışı çiziyorsanız, kutucukları su sırayla bağlamalısınız:
- > 1. `[Hibrit Arama (Chroma + BM25)]` --> ok --> `[RRF Birleştirme (Context)]`
 - > 2. `[RRF Sonucusu]` --> ok --> `[Kural Motoru (Regex)]`
 - > 3. `[Kural Motoru]` --> (Çıktı: Uyarılar) --> `[LLM (Gemma)]`
 - > 4. `[LLM]` --> (Çıktı: Ham Karar) --> `[Son Kontrol (Override)]`
 - > 5. `[Son Kontrol]` --> `[Nihai Karar (JSON)]`

E. Raporlama ve Chatbot

Tüm maddeler tek tek bu surecten geçtikten sonra:

1. **Dashboard:** Sonuçlar renkli grafiklerle ekrana yansıtılır. Kırmızı (Risk), Yeşil (Uyumlu).
2. **Otomatik Özeti:** Sistem sonuçları analiz eder ve yöneticiye "Toplam 5 risk var, en kritik olani EFT ücreti" şeklinde bir özeti yazar. Bu özeti Yapay Zeka (LLM) tarafından değil, onceden hazırlanmış akıllı şablonlar (Template) ile, cıkan istatistiklere göre anlık uretilir.
3. **Chatbot:** Tüm analizi bittikten sonra, eğer akliniza takılan bir şey olursa Chatbot'a sorabilirsiniz. Chatbot, hem orijinal yasayı hem de bizim analizi sonuçlarımızı bilir. "Neden EFT maddesine riskli dedin?" diye sorarsanız, size dayanagini (Madde 12/3 ve hesapladigi tutar farki) açıklar.

Ozet: Neden 3 Farklı Teknoloji (Vektor, Kural, LLM) Kullanıyoruz?

- * Tek başına **LLM** kullanışlık matematik hatası yapabilirdi (LLM'ler sayı saymakta bazen zorlanır).
- * Tek başına **Kural (Rule)** kullanışlık, "Müşteri magdur edilemez" gibi yorumu açık cümleleri anlayamazdı.
- * Tek başına **Vektor Arama** kullanışlık, sayıları (Madde 12 ile 13'u) karıştırabilirdi. Hepsini birleştirerek (Hybrid Pipeline), **hem matematiksel kesinliği, hem hukuki yorumlama yeteneğini hem de doğru yasaya ulaşma hızını** tek bir potada erittik.