Banka Dokümanları Uyum ve Denetim Rapor Portalı

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

1. Temel Kavramlar: Ne Yapıyoruz?

Amaç: Bir bankanın müşteri sözleşmeleri (örneğin 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.

Çözüm: Bu dokümanları bilgisayara okutup, yapay zekaya denetletmek.

2. Adım Adım Çalışma Süreci (Pipeline)

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

A. Veri Hazırlığı ve "Chunking" (Parçalama)

(İlgili Dosya: `src/banka_dokumanlari_preparation_pdf.py` ve `src/teblig_data_preparation.py`)

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

- **Chunking Nedir?** Uzun bir metni, anlam bütünlüğünü bozmadan küçük parçalara (örneğin her biri bir sözleşme maddesi olacak şekilde) bölme işlemidir.
- **Neden Yaptık?**

1. **Odaklanma:** Yapay zeka sadece ilgili maddeye odaklansın, dikkati dağılmasın.
2. **Arama Başarısı:** "Kart aidatı" diye arattığımızda, 200 sayfalık dosya yerine sadece aidatla ilgili paragrafı bulmak isteriz.

B. İndeksleme: ChromaDB ve BM25

(İlgili Dosya: `src/chroma_tool.py` - Ingest Modülü)

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

1. **ChromaDB (Vektör Veritabanı):**
   - **Nedir?** Kelimelerin "anlamını" sayılara (vektörlere) çevirir.
   - **Örnek:** Siz "ek masraf" diye ararsınız, sistem bunun "ilave ücret" veya "komisyon" ile benzer anlama 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.
   - **Örnek:** "Madde 12/A" diye ararsak, vektör veritabanı bunu "Madde 12/B" ile karıştırabilir (sayılar birbirine benzerdir). Ancak BM25, tam olarak "12/A" yazan yeri bulur.

Teknik Detay: Türkçe için BM25 Stratejimiz (Neden Zemberek Kullanmadık?)

Türkçe, sondan eklemeli bir dildir (örn: *Banka Bankalar Bankaların*). Klasik arama motorlarında "Banka" diye aratınca "Bankaların" kelimesini bulmak için *Stemming/Lemmatization* (Kelime köküne inme - Zemberek vb.) yapılması önerilir.

Ancak biz bu projede **Basit Tokenizasyon** (sadece küçük harfe çevirme) kullandık.

Neden?

1. **Hız (Latency):** Kök bulma işlemi her sorguda sistemi yavaşlatır. Hackathon'da hız kritiktir.
2. **Hibrit Gücü:** BM25'in "eklerden dolayı kaçırdığı" kelimeleri, zaten anlamdan yakalayan **Vektör Arama (ChromaDB)** tamamlamaktadır. Yani ağır bir Türkçe kütüphanesi kullanmadan, Hibrit Mimari sayesinde aynı başarıyı çok daha hızlı elde ettik.

C. Hibrit Arama (Hybrid Search) ve RRF

(İlgili Dosya: `src/chroma_tool.py` - Query Modülü)

Banka sözleşmesindeki bir maddeyi denetlemek için, o maddeyle ilgili Mevzuat (Tebliğ) kuralını bulmamız gerekir.

- **Soru:** Banka sözleşmesindeki "Yıllık üyelik bedeli 500 TL" maddesi için hangi yasaya bakmalıyım?
- **İşlem:** Sistem bu maddeyi hem ChromaDB'de (anlam) hem BM25'te (kelime) aratır.
- **Birleştirme (RRF - Reciprocal Rank Fusion):** İki farklı arama sonucunu harmanlar. Hem anlam olarak en yakın, hem de kelime olarak en uyuşan sonuçları üst sıraya taşır. Böylece doğru yasayı bulma ihtimalimiz maksimuma çıkar.

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

(İlgili Dosya: `src/llm_compliance_check.py`)

Doğru yasayı bulduk (Hibrit Arama tamamlandı). Şimdi "Banka maddesi bu yasaya uygun mu?" sorusunu cevaplamalıyız.

Bu motor, **her zaman ve kesinlikle sırasıyla** çalışan 3 adımdan oluşur:

1. **Adım: Kural Motoru (Rule Engine) - [Hesaplama & Uyarı]**

- **Yapı:** Sadece basit Regex (Kelime Arama) değildir. **Regex + Python Mantığı** birlikte çalışır.
- **Nasıl Çalışır?**

1. **Veri Çıkarma (Regex):** Metin içindeki sayıları ("5 TL"), oranları ("%10") ve anahtar kelimeleri ("Onaysız") Regex ile çekip çıkarır.
2. **Mantıksal Kıyas (Python):** Çıkarılan bu sayıları matematiksel olarak karşılaştırır (`if banka_ucreti > yasa_limiti`).

- **Örnek:** Yasa "EFT max 2 TL" derken Banka "5 TL" demişse, Regex "2" ve "5" sayılarını bulur, Python kodu "5 > 2" işlemini yapıp hatayı yakalar.
- **Kritik Nokta:** Kural motoru burada işlemi bitirmez. Tespit ettiği hatayı bir "UYARI ETİKETİ" (Warning Label) haline getirir ve bir sonraki aşamaya (LLM'e) paslar.

2. **Adım: Büyük Dil Modeli (LLM - Gemma) - [Akıl Yürütme & Karar]**

- Yapay zeka devreye girer ve kendisine sunulan **TÜM** verileri okur:
- Girdi 1: Banka Maddesi
- Girdi 2: İlgili Yasa (Mevzuat)
- Girdi 3: **Kural Motorundan Gelen Uyarı** ("Dikkat: Burada matematiksel limit aşımı 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) - [Güvenlik 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 yasağı görürse kararı ezer (**Overrule**) ve sonucu "UYUMSUZ" olarak değiştirir.

Özet Sıralama: Hibrit Arama -> Kural Uyarısı -> LLM Kararı -> Son Kontrol (Override). Bu sıra asla değişmez.

E. Raporlama ve Chatbot

(İlgili Dosya: `src/streamlit_compliance_viewer.py` ve `src/report_generator_pdf.py`)

Tüm maddeler tek tek bu süreçten geçtikten sonra:

1. **Dashboard:** Sonuçlar renkli grafiklerle ekrana yansıtılır. Kırmızı (Risk), Yeşil (Uyumlu).
2. **Otomatik Özet:** Sistem sonuçları analiz eder ve yöneticiye "Toplam 5 risk var, en kritik olanı EFT ücreti" şeklinde bir özet yazar. Bu özet Yapay Zeka (LLM) tarafından değil, önceden hazırlanmış akıllı şablonlar (Template) ile, çıkan istatistiklere göre anlık üretilir.
3. **Chatbot:** Tüm analiz bittikten sonra, eğer aklınıza takılan bir şey olursa Chatbot'a sorabilirsiniz. Chatbot, hem orijinal yasayı hem de bizim analiz sonuçlarımızı bilir. "Neden EFT maddesine riskli dedin?" diye sorarsanız, size dayanağını (Madde 12/3 ve hesapladığı tutar farkı) açıklar.

Özet: Neden 3 Farklı Teknoloji (Vektör, Kural, LLM) Kullanıyoruz?

- Tek başına **LLM** kullansaydık matematik hatası yapabilirdi (LLM'ler sayı saymakta bazen zorlanır).
- Tek başına **Kural (Rule)** kullansaydık, "Müşteri mağdur edilemez" gibi yoruma açık cümleleri anlayamazdı.
- Tek başına **Vektör Arama** kullansaydık, sayıları (Madde 12 ile 13'ü) karıştırabilirdi.

Hepsini birleştirerek (Hybrid Pipeline), **hem matematiksel kesinliği, hem hukuksal yorumlama yeteneğini hem de doğru yasaya ulaşma hızını** tek bir potada erittik.