TÜİK İstatistik Chatbot

RAG Tabanlı Akıllı Soru-Cevap Sistemi

Teknolojiler:
OpenAI GPT-4o Mini | LangChain | ChromaDB | RAGAS | Streamlit

Türkiye Gençlik, Aile ve Yaşlı İstatistikleri
Retrieval-Augmented Generation ile Desteklenmiş

Proje Özeti

TÜİK İstatistik Chatbot, Türkiye İstatistik Kurumu'nun gençlik, aile ve yaşlı istatistiklerine yönelik sorulara doğru ve güvenilir cevaplar veren, RAG (Retrieval-Augmented Generation) teknolojisi ile geliştirilmiş akıllı bir soru-cevap sistemidir.

> **Temel Özellikler**
>
> - **Akıllı Döküman Retrieval:** Yıl ve kategori bazlı filtreleme ile optimize edilmiş arama
> - **Metadata Tabanlı Sınıflandırma:** Her chunk otomatik olarak yıl ve kategori bilgisi ile etiketlenir
> - **Kavram Uyumsuzluk Kontrolü:** LLM'in yanlış bilgi üretmesini engelleyen guard mekanizması
> - **Karşılaştırma Desteği:** Çoklu yıl sorgularında gelişmiş retrieval stratejisi
> - **RAGAS Entegrasyonu:** Performans değerlendirmesi için kapsamlı metrik sistemi
> - **Kalıcı Vektör Veritabanı:** ChromaDB ile embedding'lerin disk üzerinde saklanması

Proje Hedefleri

- TÜİK PDF dokümanlarından otomatik bilgi çıkarımı yapılması
- Kullanıcı sorularına hızlı ve doğru yanıtlar verilmesi
- Yıl bazlı karşılaştırma ve trend analizi imkanı sağlanması
- Halüsinasyon (yanlış bilgi üretme) önleme mekanizmalarının geliştirilmesi
- Ölçülebilir ve sürekli iyileştirilebilir performans sunulması

☐ Teknik Mimari

RAG Pipeline Akışı

```
PDF Dokümanlar → Chunk'lama → Embedding
                                  ↓
ChromaDB ↔ Retriever → LLM → Cevap
```

Teknoloji Stack'i

Katman	Teknoloji	Kullanım Amacı
LLM	OpenAI GPT-4o Mini	Cevap üretimi ve akıl yürütme
Embedding	text-embedding-3-small	Semantik vektör temsili
Vector DB	ChromaDB	Vektör saklama ve arama
Framework	LangChain	RAG pipeline orkestrasyon
UI	Streamlit	Web arayüzü
Evaluation	RAGAS	Performans ölçümü
PDF Parser	PyPDF	PDF okuma ve işleme

Yapılandırma Parametreleri

Parametre	Değer	Açıklama
CHUNK_SIZE	600	Her chunk'ın karakter boyutu
CHUNK_OVERLAP	120	Chunk'lar arası örtüşme (%20)
TOP_K	4 (12)	Döndürülecek chunk sayısı (karşılaştırmada 12)
TEMPERATURE	0	Deterministik sonuçlar için

Önemli Özellikler

1. Metadata Tabanlı Akıllı Retrieval

Sistem, her PDF dokümanından otomatik olarak kategori ve yıl bilgisi çıkarır:

```
def extract_metadata(filename: str) -> Dict[str, str]: metadata = {"kategori": "bilinmiyor", "yil": "bilinmiyor"} filename_lower = filename.lower() if "genclik" in filename_lower: metadata["kategori"] = "genclik" elif "yasli" in filename_lower: metadata["kategori"] = "yasli" elif "aile" in filename_lower: metadata["kategori"] = "aile" year_match = re.search(r'_(\d{2})\.pdf', filename) if year_match: metadata["yil"] = f"20{year_match.group(1)}" return metadata
```

Avantajlar:

- Her chunk, hangi yıl ve kategoriye ait olduğunu bilir
- Yıl bazlı filtreleme ile alakasız dökümanlar elenir
- Karşılaştırma sorularında çoklu yıl araması yapılabilir
- Kategori bazlı sorgularda daha hassas sonuçlar elde edilir

2. Kavram Uyumsuzluk Kontrolü

LLM'in benzer ama farklı kavramları karıştırmasını önleyen kontrol mekanizması:

```
def guard_mismatch(question: str, docs: List[Document]) -> bool: q = question.lower() ctx = " ".join(d.page_content for d in docs).lower() # Yaşlı nüfus oranı vs yaşlı bağımlılık oranı if "yaşlı nüfus oran" in q and "yaşlı bağımlılık oran" in ctx: if "yaşlı nüfus oran" not in ctx: return True # Beklenen yaşam süresi türleri if "beklenen yaşam süresi" in q: if "doğuşta" in q and ("65 yaş" in ctx): if "doğuşta" not in ctx: return True return False
```

Önlenen Hatalar:

- "Yaşlı nüfus oranı" ≠ "Yaşlı bağımlılık oranı"
- "Doğuşta beklenen yaşam süresi" ≠ "65 yaşında beklenen yaşam süresi"
- Kavram uyumsuzluğu tespit edilirse sistemden uyarı verilir

3. Gelişmiş Karşılaştırma Desteği

Sistem, karşılaştırma sorularını otomatik olarak tespit eder ve retrieval stratejisini optimize eder:

```
def retrieve_docs(question: str) -> List[Document]: years = extract_years(question) # Karşılaştırma veya çoklu yıl → k'yı artır k = 12 if len(years) >= 2 else TOP_K docs_all = [] # Her yıl için ayrı arama if years and st.session_state.vector_store: for year in years: retriever = st.session_state.vector_store.as_retriever( search_kwargs={"k": k, "filter": {"yil": year}} ) docs_all.extend(retriever.invoke(question))
```

Dinamik Retrieval Stratejisi

Soru Tipi	K Değeri	Strateji
Tek yıl sorgusu	4	Yıl filtresi + standart arama
Çoklu yıl (2+)	12	Her yıl için ayrı arama + birleştirme
Yetersiz sonuç	4-12	Fallback: genel arama

4. Context Formatting

LLM'in chunk'ları doğru yorumlaması için özel formatlama uygulanır:

```
def format_docs(docs: List[Document]) -> str: formatted = [] for i, doc in enumerate(docs, 1): kategori = doc.metadata.get('kategori', 'bilinmiyor').upper() yil = doc.metadata.get('yil', 'bilinmiyor') formatted.append( f"[Kaynak {i} - {kategori} {yil}]\n{doc.page_content}\n" ) return "\n".join(formatted)
```

Örnek Formatlanmış Context:

```
[Kaynak 1 - GENCLIK 2018] 2018 yılında gençlerde işsizlik oranı %20,3 oldu... [Kaynak 2 - GENCLIK 2020] 2020 yılında gençlerde işsizlik oranı %25,9 oldu...
```

RAGAS Performans Değerlendirmesi

RAGAS (Retrieval-Augmented Generation Assessment), RAG sistemlerinin performansını objektif olarak ölçen bir framework'tür.

Değerlendirilen Metrikler

Metrik	Açıklama	Yüksek Değer Anlamı
Context Precision	Getirilen bağlamın ne kadar ilgili olduğu	Gereksiz chunk yok
Context Recall	Gerekli bilginin ne kadarının getirildiği	Eksik chunk yok
Faithfulness	Cevabın bağlama ne kadar sadık olduğu	Halüsinasyon yok
Answer Relevancy	Cevabın soruyla ne kadar ilgili olduğu	Gereksiz bilgi yok

Ground Truth Test Soruları

Sistem performansını ölçmek için 7 adet ground truth sorusu tanımlanmıştır:

1. 2020 yılında genç nüfus oranı nedir?
2. 2023 yılında akraba evliliği oranı nedir?
3. 2014 yılında boşanan çift sayısı kaçtır?
4. 2018 yılında gençlerde işsizlik oranı nedir?
5. 2020 yılında ne eğitimde ne istihdamda olan gençlerin oranı nedir?
6. 2023 yılında internet kullanan gençlerin oranı nedir?
7. 2024 yılında yaşlı nüfus kaç kişidir?

> **RAGAS Kullanımı:** Ana chatbot sayfasında en az bir soru sorduktan sonra, sağ alt köşedeki "RAGAS Değerlendirmesine Git" butonuna tıklayarak performans değerlendirmesi sayfasına erişebilirsiniz.

Kullanım Senaryoları

Senaryo 1: Basit Sorgular

> **Kullanıcı:** "2020 yılında genç nüfus oranı nedir?"
>
> **Chatbot:** "2020 yılında genç nüfus, toplam nüfusun %15,4'ünü oluşturdu."
>
> **İşleyiş:**
>
> 1. Sorgudan yıl çıkarılır: 2020
> 2. Vektör veritabanında yıl filtresi ile arama yapılır
> 3. En alakalı 4 chunk getirilir
> 4. LLM, chunk'lardan cevap üretir

Senaryo 2: Karşılaştırma Sorguları

> **Kullanıcı:** "2018 ve 2020 yıllarında genç işsizlik oranı nasıl değişti?"
>
> **Chatbot Cevabı:**
>
> | Yıl | İşsizlik Oranı |
> |---|---|
> | 2018 | %20,3 |
> | 2020 | %25,9 |
>
> 2018'den 2020'ye +5,6 puanlık artış görülmüştür.
>
> **İşleyiş:**
>
> 1. Çoklu yıl tespiti: [2018, 2020]
> 2. K değeri 12'ye çıkarılır
> 3. Her yıl için ayrı arama yapılır
> 4. Tüm chunk'lar birleştirilerek kapsamlı cevap üretilir

Senaryo 3: Kavram Uyumsuzluğu

> **Kullanıcı:** "Yaşlı nüfus oranı nedir?"
>
> **Retrieval:** Yanlışlıkla "yaşlı bağımlılık oranı" chunk'ları getirildi
>
> **Guard Mechanism:** Uyumsuzluk tespit edildi ✖
>
> **Chatbot:** "Bu bilgi dokümanlarda bulunmamaktadır."

> **Sonuç:** Yanlış bilgi üretimi başarıyla engellenmiştir ✅

Kurulum ve Çalıştırma

1. Gereksinimler

- Python 3.8 veya üzeri
- OpenAI API anahtarı
- 5-10 GB disk alanı (vektör veritabanı için)

2. Kurulum Adımları

```
# Repository'yi klonlayın git clone <repository-url> cd CHATBOT_DERSI # Bağımlılıkları yükleyin pip install -r requirements.txt # .env dosyası oluşturun echo "OPENAI_API_KEY=your_api_key_here" > .env # PDF'leri data/ klasörüne ekleyin # Dosya formatı: kategori_yıl.pdf # Örnek: genclik_20.pdf (2020 Gençlik İstatistikleri)
```

3. Ana Kütüphaneler

Kütüphane	Kullanım Amacı
streamlit	Web arayüzü
langchain	RAG pipeline
langchain-openai	OpenAI entegrasyonu
langchain-chroma	Vektör veritabanı
pypdf	PDF okuma
ragas	Performans değerlendirmesi

4. Uygulamayı Çalıştırma

```
streamlit run pdf_gpt_3.py
```

> **İlk Çalıştırma:** Uygulama açıldığında PDF'ler otomatik olarak yüklenecek ve vektör veritabanı oluşturulacaktır. Bu işlem ilk seferde birkaç dakika sürebilir. Sonraki çalıştırmalarda mevcut veritabanı kullanılacağı için uygulama hızlı açılır.

Sorun Giderme

Problem 1: "OPENAI_API_KEY bulunamadı"

> **Çözüm:**
>
> - .env dosyasının proje kök dizininde olduğundan emin olun
> - API anahtarınızın doğru formatta olduğunu kontrol edin
> - .env dosyasında OPENAI_API_KEY=your_key_here formatını kullanın

Problem 2: "PDF bulunamadı"

> **Çözüm:**
>
> - data/ klasörünün proje kök dizininde bulunduğundan emin olun
> - PDF dosyalarının doğru formatta adlandırıldığını kontrol edin
> - Dosya formatı: kategori_yil.pdf (örn: genclik_20.pdf)
> - En az bir PDF dosyasının data/ klasöründe olması gerekir

Problem 3: Vektör veritabanı yavaş yükleniyor

> **Çözüm:**
>
> - İlk yüklemede normaldir, embedding işlemi zaman alır
> - Sonraki çalıştırmalarda chroma_db/ klasörü kullanılır ve hızlıdır
> - Veritabanını yeniden oluşturmak için chroma_db/ klasörünü silin

Problem 4: Yanlış cevaplar alıyorum

> **Çözüm:**
>
> - CHUNK_SIZE ve CHUNK_OVERLAP değerlerini ayarlayın
> - TOP_K değerini artırarak daha fazla context getirin
> - PDF'lerin doğru formatlandırıldığından emin olun
> - Sorunuzun net ve açık olduğundan emin olun

Proje Yapısı

```
CHATBOT_DERSI/
│
├── .vscode/ # VS Code yapılandırma dosyaları
├── chroma_db/ # Kalıcı vektör veritabanı (otomatik oluşturulur)
├── data/ # PDF dokümanlar (TÜİK istatistikleri)
├── pages/ # Streamlit sayfaları
│ └── ragas_evaluation.py # RAGAS değerlendirme sayfası
│
├── .env # Ortam değişkenleri (OPENAI_API_KEY)
├── .gitignore # Git ignore dosyası
├── pdf_gemini_1.py # Alternatif implementasyon (Gemini)
├── pdf_gpt_2.py # Alternatif implementasyon (GPT v2)
├── pdf_gpt_3.py # ⭐ Ana uygulama dosyası
├── requirements.txt # Python bağımlılıkları
└── README.md # Proje dokümantasyonu
```

PDF Dosya İsimlendirme

PDF dosyaları data/ klasörüne aşağıdaki formatta eklenmelidir:

Format	Örnek	Açıklama
kategori_yıl.pdf	genclik_20.pdf	2020 Gençlik İstatistikleri
kategori_yıl.pdf	yasli_23.pdf	2023 Yaşlı İstatistikleri
kategori_yıl.pdf	aile_18.pdf	2018 Aile İstatistikleri

> **Not:** Yıl bilgisi iki haneli olarak verilmelidir (_20.pdf = 2020, _23.pdf = 2023). Sistem otomatik olarak 20XX formatına çevirir.

Chunk Boyutu Optimizasyonu

> **Chunk Size (600 karakter):**
>
> - 500-800 arası değerler genelde iyi sonuç verir
> - Daha küçük parçalar context kaybına neden olur
> - Daha büyük parçalar gürültü oluşturur
>
> **Overlap (120 karakter - %20):**
>
> - Chunk'lar arası bağlam sürekliliği sağlar
> - CHUNK_SIZE'ın %20'si optimal oran
> - Daha hassas sorgular için artırılabilir

Sonuç ve Gelecek Çalışmalar

Proje Başarıları

> - Yıl bazlı filtreleme ile yüksek doğruluk oranı
> - Kavram uyumsuzluk kontrolü ile halüsinasyon önleme
> - Karşılaştırma sorularında dinamik retrieval
> - RAGAS ile ölçülebilir performans
> - Kalıcı vektör veritabanı ile hızlı başlatma

Gelecek Geliştirmeler

- **Çoklu Dil Desteği:** İngilizce ve diğer dillerde soru-cevap
- **Grafik ve Görselleştirme:** İstatistiklerin otomatik grafik oluşturma
- **API Entegrasyonu:** REST API ile dış sistemlere entegrasyon
- **Daha Fazla Kategori:** Diğer TÜİK istatistik kategorileri
- **Gelişmiş Filtreleme:** Bölge, şehir bazlı filtreleme
- **Kullanıcı Geri Bildirimi:** Cevap kalitesi değerlendirme sistemi

Teknik İyileştirmeler

Alan	İyileştirme Önerisi
Embedding Model	text-embedding-3-large ile daha iyi semantik arama
LLM Model	GPT-4 ile daha karmaşık analiz yetenekleri
Reranking	Cohere Rerank ile daha iyi sonuç sıralaması
Caching	Sık sorulan sorular için cache mekanizması

Eğitim Amaçları

Bu proje, RAG (Retrieval-Augmented Generation) teknolojisinin gerçek dünya uygulamasını göstermek amacıyla geliştirilmiştir. Öğrenilen temel konular:

- Vektör veritabanı tasarımı ve kullanımı
- Embedding ve semantik arama
- LLM prompt mühendisliği
- Metadata tabanlı filtreleme
- Performans değerlendirme metrikleri
- Halüsinasyon önleme teknikleri

İletişim ve Katkı

Sorularınız veya önerileriniz için GitHub üzerinden issue açabilirsiniz.

Bu proje eğitim amaçlıdır ve açık kaynak topluluk katkılarına açıktır.