# Carbon Consulting GenAI Bootcamp Final Projesi Raporu

## Proje Raporu: Resmi Gazete ve Genel Haber Odaklı Agentic AI Chatbot

**Burak Kılıç**

**1. Giriş**

Proje, LangGraph framework'ü kullanılarak bir supervisor-agent mimarisi üzerine inşa edilmiş, kullanıcı etkileşimi için Streamlit arayüzü sağlanmış ve dağıtım kolaylığı için Docker ile paketlenmiştir. Sistemin temel amacı, kullanıcının sorusunun niteliğine göre en uygun bilgi kaynağını otomatik olarak seçip (Resmi Gazete veritabanı veya güncel haber API'si), ilgili yanıtı üretmektir.

NOT: Resmi Gazete dosyaları Kaggle Notebook'ta bge-m3 embedding modeli kullanılarak ChromaDB'ye kaydedilmiştir. (Lokalde bu işlemi yapmak çok uzun sürdüğü için). Ayrıca resmi gazete verilerinin pdf dosyalarının çok düzensiz ve yoğun görsel içermesi sebebiyle doğrudan pdf dosyaları üzerinden değil, içerisinden text verileri çekilerek .txt uzantılı dosyalara dönüştürülüp, bu txt dosyalarını embeddinglere dönüştürerek ChromaDB'ye kaydedilmiştir.

**2. Sistem Mimarisi**

Geliştirilen sistem, LangGraph kullanılarak modellenmiş bir **supervisor-agent mimarisine** dayanmaktadır. Bu mimaride, merkezi bir "supervisor" node'u gelen isteği analiz eder ve görevi uygun "agent" node'una yönlendirir.

- **Supervisor Node (classify_question_node):**
    1. Gelen kullanıcı sorusunu alır.
    2. Önceden eğitilmiş bir Büyük Dil Modeli'ni (LLM - deepseek-r1:14b) kullanarak soruyu üç kategoriden birine sınıflandırır:
        - resmi_gazete: Türkiye Cumhuriyeti Resmi Gazetesi içeriğiyle ilgili sorular.
        - general: Güncel olaylar, genel kültür veya Resmi Gazete dışındaki konular.
        - irrelevant: Anlamsız, uygunsuz veya yanıtlanamayan sorular.
    3. Sınıflandırma sonucunu LangGraph state'ine (AgentState) yazar. Bu state, iş akışındaki diğer node'lar arasında bilgi taşımak için kullanılır.
- **Agent Nodes:** Supervisor tarafından belirlenen sınıflandırmaya göre aşağıdaki agent node'larından biri tetiklenir:
    1. **Resmi Gazete RAG Agent (resmi_gazete_rag_node):**
        - resmi_gazete olarak sınıflandırılan soruları ele alır.
        - Kullanıcının sorusunu kullanarak, bge-m3:latest embedding modeli ile oluşturulmuş ve ChromaDB'de saklanan Resmi Gazete belgelerinin vektör temsillerinde bir anlamsal arama (Retrieval) gerçekleştirir.
        - En ilgili belge parçacıklarını (context) alır.
        - Bu context'i ve orijinal soruyu LLM'e (deepseek-r1:14b) sunarak, *yalnızca sağlanan context'e dayalı* bir yanıt üretmesini ister (Retrieval-Augmented Generation - RAG).
        - Üretilen yanıtı, kaynağı ("Resmi Gazete") ve kullanılan context'i state'e yazar.
    2. **Genel Bilgi Agent (general_knowledge_node):**
        - general olarak sınıflandırılan soruları ele alır.

        - **Öncelikli olarak:** NEWSDATA_API_KEY ortam değişkeni tanımlıysa, NewsData.io API'sini kullanarak soruyla ilgili güncel Türkçe haberleri arar.
        - **Haber Bulunursa:** Bulunan haberlerin başlık ve özetlerini bir context olarak formatlar. Bu haber context'ini ve orijinal soruyu LLM'e sunarak, hem haberlerden hem de kendi genel bilgisinden yararlanarak bir yanıt üretmesini ister. Kaynak olarak "Genel Bilgi (NewsData.io)" belirtilir.
        - **Haber Bulunamazsa / API Hatası / API Anahtarı Yoksa:** Haber arama adımını atlar veya oluşan hatayı not eder. Soruyu doğrudan LLM'e sorarak, modelin kendi dahili bilgisine dayalı bir yanıt üretmesini ister. Kaynak olarak "Genel Bilgi (LLM)" veya ilgili hata durumu belirtilir.
        - Üretilen yanıtı ve kaynağı state'e yazar. (Bu agent RAG context'i döndürmez).
    3. **Fallback Agent (fallback_node):**
        - irrelevant olarak sınıflandırılan soruları ele alır.
        - Soruya yanıt veremediğini belirten standart, kibar bir mesaj üretir.
        - Yanıtı ve kaynağı ("Yanıt Yok") state'e yazar.
- **Yönlendirme Mantığı (route_question):** Supervisor node'undan sonra çalışır. State'deki classification değerine bakarak iş akışını ilgili agent node'una (resmi_gazete_agent, general_agent, fallback_agent) yönlendirir.
- **Durum Yönetimi (AgentState):** LangGraph'ın TypedDict tabanlı state'i, question, classification, context (Resmi Gazete RAG için), answer, source ve error gibi bilgileri node'lar arasında taşır ve iş akışının sonunda nihai sonucu içerir.
- **Sonlanma (END):** Her agent node'u görevini tamamladıktan sonra iş akışı sona erer.

**3. Kullanılan Teknolojiler ve Araçlar**

- **Streamlit:** Kullanıcıların soru sormasını ve chatbot yanıtlarını (kaynak ve RAG context detayları dahil) görmesini sağlayan basit ve etkileşimli bir web arayüzü oluşturmak için kullanıldı. Chat geçmişi st.session_state içinde tutulmaktadır.
- **LangGraph:** Supervisor-agent mimarisini uygulamak, durum takibi yapmak ve node'lar arasında koşullu yönlendirmeyi sağlamak için kullanılan ana framework'tür. Durum makinesi (StateGraph) tanımlanarak iş akışı (workflow) oluşturulmuştur.
- **LangChain:** LangGraph ile entegre çalışan ve LLM'ler, embedding modelleri ve vektör veritabanları ile etkileşimi kolaylaştıran kütüphanedir. Spesifik olarak ChatOllama, OllamaEmbeddings ve Chroma (vektör deposu arayüzü) bileşenleri kullanılmıştır.
- **Ollama:** Büyük Dil Modellerini (LLM - deepseek-r1:14b) ve Embedding Modellerini (bge-m3:latest) lokal olarak çalıştırmak ve sunmak için kullanılmıştır. Kod, Ollama'nın hem lokalde hem de Docker container içinden (host makineye erişimle) çalışmasını destekleyecek şekilde yapılandırılmıştır (OLLAMA_BASE_URL ayarı).
- **ChromaDB:** Resmi Gazete belgelerinden çıkarılan metin parçacıklarının embedding'lerini (vektörlerini) depolamak ve anlamsal arama (similarity search) yapmak için kullanılan açık kaynaklı vektör veritabanıdır. Veritabanı dosyaları (chroma_db klasörü) proje dizininde saklanmakta ve uygulama başlatıldığında yüklenmektedir.

- **NewsData.io API:** Genel bilgi soruları için güncel haberleri çekmek amacıyla kullanılan harici bir API'dir. NewsDataApiClient kütüphanesi ile entegrasyon sağlanmıştır. API anahtarının (NEWSDATA_API_KEY) ortam değişkeni olarak ayarlanması gerekmektedir; aksi takdirde bu özellik devre dışı kalır.
- **Python:** Projenin ana programlama dilidir.
- **Docker:** Uygulamanın tüm bağımlılıkları ile birlikte paketlenmesi, taşınabilirliğin sağlanması ve farklı ortamlarda kolayca çalıştırılabilmesi için kullanılmıştır. Dockerfile, gerekli sistem kütüphanelerini (örn: build-essential for chroma-hnswlib), Python bağımlılıklarını kurar, uygulama kodunu ve ChromaDB verilerini kopyalar ve Streamlit uygulamasını başlatır.

**4. Uygulama Akışı (Adım Adım)**

1. **Başlatma:** Uygulama (app.py) çalıştırıldığında:
    - Ortam değişkenleri (.env) yüklenir (API Anahtarları, Ollama Host).
    - Ollama sunucusunun adresi belirlenir (Lokal/Docker).
    - Ollama Embedding ve LLM modelleri belirtilen adres üzerinden başlatılır.
    - ChromaDB veritabanı (./chroma_db) yüklenir ve bir retriever nesnesi oluşturulur.
    - NewsData.io API anahtarının varlığı kontrol edilir.
    - LangGraph iş akışı (app) derlenir.
    - Streamlit arayüzü başlatılır.
2. **Kullanıcı Etkileşimi:**
    - Kullanıcı Streamlit arayüzünden bir soru girer.
    - Soru, chat geçmişine eklenir ve ekranda gösterilir.
3. **LangGraph İş Akışı:**
    - Kullanıcının sorusu {"question": prompt} girdisiyle LangGraph app.invoke() fonksiyonuna verilir.
    - **Supervisor Node (classify_question_node):** Soruyu alır, LLM ile sınıflandırır (resmi_gazete, general, irrelevant).
    - **Yönlendirme (route_question):** Sınıflandırmaya göre akışı uygun agent'a yönlendirir.
    - **Agent Node (ilgili olan):**
        - *Resmi Gazete Agent:* ChromaDB'den RAG yapar, LLM ile yanıt üretir.
        - *Genel Bilgi Agent:* NewsData.io'yu dener (varsa), LLM ile yanıt üretir.
        - *Fallback Agent:* Standart yanıt üretir.
    - Seçilen agent, answer, source, context (varsa) ve error (varsa) bilgilerini içeren state'i güncelleyerek iş akışını tamamlar.
4. **Yanıt Gösterimi:**
    - LangGraph'tan dönen nihai state'deki answer kullanıcıya gösterilir.
    - source bilgisi (Resmi Gazete, Genel Bilgi (NewsData.io), Genel Bilgi (LLM), Yanıt Yok, Hata) yanıtın altında belirtilir.
    - Eğer kaynak "Resmi Gazete" ise ve context mevcutsa, bu RAG context'i "Detay: RAG Bağlamı" başlığı altında genişletilebilir bir alanda gösterilir.
    - Olası hatalar (error) varsa, st.error ile kullanıcıya bildirilir.
    - Asistanın yanıtı (tüm detaylarla birlikte) chat geçmişine eklenir.

**5. Dockerization**

Dockerfile aşağıdaki adımları gerçekleştirerek uygulamanın bir container imajını oluşturur:

1. python:3.11-slim temel imajını kullanır.
2. RUNNING_IN_DOCKER=true ortam değişkenini ayarlar (uygulama içindeki Ollama host mantığı için).
3. build-essential gibi derleme araçlarını yükler. Bu, chroma-hnswlib gibi C++ eklentileri içeren Python paketlerinin kurulumu için kritiktir.
4. requirements.txt dosyasını kopyalar ve pip kullanarak bağımlılıkları kurar.
5. Uygulama kodunu (app.py) ve ChromaDB veri klasörünü (chroma_db) container içine kopyalar.
6. Streamlit için 8501 portunu dışarıya açar.
7. Container başlatıldığında streamlit run app.py komutunu çalıştırarak uygulamayı tüm ağ arayüzlerinde (0.0.0.0) erişilebilir şekilde başlatır.

---