KARABÜK ÜNİVERSİTESİ ÇOKLU AJANLI VE RAG TABANLI BİLGİ ASİSTANI: KÜBAS

2025

BİLGİSAYAR MÜHENDİSLİĞİ

BİTİRME PROJESİ TEZİ

BÜŞRA UZUN

ERENCAN EKİNCİ

KARABÜK ÜNİVERSİTESİ ÇOKLU AJANLI VE RAG TABANLI BİLGİ ASİSTANI: KÜBAS

BÜŞRA UZUN

ERENCAN EKİNCİ

Karabük Üniversitesi

Mühendislik Fakültesi

Bilgisayar Mühendisliği (%100 İngilizce)

Bölümünde

Bitirme Projesi Tezi

Olarak Hazırlanmıştır.

KARABÜK

2025

TEŞEKKÜR

Bu tez çalışmasının planlanmasından yürütülmesine kadar tüm aşamalarında değerli bilgi ve tecrübeleriyle bizlere rehberlik eden, her zaman destek ve ilgisini esirgemeyen, yönlendirmeleriyle çalışmamızın daha nitelikli ve amacına uygun şekilde ilerlemesine katkı sağlayan kıymetli hocamız Dr. Ferhat Atasoy'a içten teşekkürlerimizi sunarız. Kendisine bizlere kattığı bilgi birikimi ve sağladığı yol göstericilik için ayrıca teşekkür ederiz.

İÇİNDEKİLER

Sayfa

ÖZET 5
ABSTRACT 6
GİRİŞ 7
Konunun Önemi 7
Projenin Amacı 8
LİTERATÜR TARAMASI 9
MATERYAL-METOT 10
Retrieval-Augmented Generation (RAG) Sistemlerinin Kullanımı 10
RAG Mimarisi 10
LangGraph ve Çok Ajanlı Sistem Mimarileri 12
UYGULAMA 14
Genel Sistem Mimarisi 14
RAG Yaklaşımı 15
LangGraph Temelli Akış Kontrolü 16
Web Scraping ve Güncel İçerik Getirme 19
Ön Uç Geliştirme 20
API Hizmetleri 20
Prompt Mühendisliği 21
Dağıtım ve Performans Optimizasyonu 22
DENEYSEL SONUÇLAR 23
TARTIŞMA 24
KAYNAKÇA 25

ÖZET

Günümüzde teknolojinin hızlı gelişimi, eğitim kurumlarında bilgiye hızlı, doğru ve erişilebilir bir şekilde ulaşma ihtiyacını daha da önemli hâle getirmiştir. Özellikle üniversitelerde artan öğrenci sayısı, çeşitlenen akademik yapı ve yoğun bilgi akışı, geleneksel bilgilendirme yöntemlerinin yetersiz kalmasına yol açmaktadır. Bu çalışma, Karabük Üniversitesi Mühendislik Fakültesi bünyesinde kullanılmak üzere, üniversiteyle ilgili çeşitli konularda kullanıcı sorularını yanıtlayabilen bir sanal asistan uygulamasının geliştirilmesini kapsamaktadır. Geliştirilen sistem, güncel yazılım mimarisi yaklaşımları temel alınarak iki katmanlı bir yapı üzerinde inşa edilmiştir. Kullanıcı deneyimini önceliklendiren ön yüz (frontend), React.js teknolojisiyle tasarlanmış; hızlı ve güvenilir veri akışı sağlamak amacıyla arka uç (backend), FastAPI altyapısıyla yapılandırılmıştır. Sistemin doğal dil işleme yetenekleri OpenAI tarafından geliştirilen GPT-4o modeliyle sağlanmış; ayrıca çoklu ajan (multi-agent) mimarisini destekleyen LangGraph entegrasyonu sayesinde bağlamsal doğruluk güçlendirilmiştir. Sanal asistan, kullanıcıların hem üniversite hem de mühendislik fakültesi hakkında bilgi edinmesini sağlarken, aynı zamanda fakülteye ait internet sitelerinde yayımlanan duyurular ve etkinlik içeriklerini de sunabilmektedir. Bu bilgiler, web kazıma (web scraping) yöntemleriyle dinamik olarak toplanmakta ve modele entegre edilmektedir. Bu yapı, klasik bir RAG (Retrieval-Augmented Generation**)** sistemine benzer şekilde çalışmakta; modele statik bilgi yerine güncel veri sağlama imkânı sunmaktadır. Böylece, büyük dil modellerinin yeniden eğitilmesine (fine-tuning) gerek kalmaksızın, düşük maliyetli ve bağlama uygun bilgi sunumu mümkün hâle gelmiştir. Sonuç olarak geliştirilen sistem, üniversitenin dijitalleşme sürecine katkı sunmakta; benzer yapay zekâ tabanlı çözümler için bütüncül, ölçeklenebilir ve optimize edilmiş bir örnek teşkil etmektedir.

Anahtar Sözcükler: Sanal Asistan, RAG, LangGraph, FastAPI, React.js, Web Scraping, Çok Ajanlı Sistemler, Web Tabanlı Sohbet Botu

ABSTRACT

In today's world of rapidly evolving technology, the need for fast, accurate, and accessible information has become increasingly important within educational institutions. Especially in universities, the growing number of students, diversified academic structures, and intense information flow have rendered traditional communication methods insufficient. This study focuses on the development of a virtual assistant application designed for Karabük University’s Faculty of Engineering, aiming to provide instant answers to users' questions related to the university.

The system is built upon a two-layer architecture based on modern software development practices. The frontend is developed using React.js with a focus on user experience, while the backend is implemented with FastAPI to ensure fast and reliable data flow. The virtual assistant’s natural language processing capabilities are powered by OpenAI’s GPT-4o model, and its contextual accuracy is enhanced through the integration of LangGraph, which enables a multi-agent architecture.

The assistant provides users with information about both the university and the faculty, while also delivering up-to-date content such as announcements and events published on official university websites. These dynamic data are collected through web scraping techniques and integrated into the system in real time. This structure operates similarly to a Retrieval-Augmented Generation (RAG) approach by delivering up-to-date information to the model without requiring static fine-tuning of the language model. As a result, it enables low-cost, context-aware information delivery.

In conclusion, the developed system contributes to the university’s digital transformation efforts and presents a comprehensive, scalable, and optimized example for similar AI-based solutions.

Keywords: Virtual Assistant, RAG, LangGraph, FastAPI, React.js, Web Scraping, Multi-Agent Systems, Web-Based Chatbot

Giriş

Konunun Önemi

Günümüz üniversitelerinde, öğrenci ve personelin çeşitli bilgilere hızlı ve güvenilir bir şekilde erişim ihtiyacı giderek artmaktadır. Özellikle üniversitelerin büyüklüğü, uluslararası öğrenci çeşitliliği ve öğrenci sayısındaki artış, bilgi akışının hızlı ve etkili bir biçimde yürütülmesini zorlaştırmaktadır. Ayrıca üniversiteye yeni başlayan ve oryantasyon süreçlerinde eksiklikleri olan öğrenciler, duyuru, yönetmelik veya yönerge okumaktansa doğrudan erişebildikleri akademik veya idari personele soru sorarak çözüm aramayı tercih etmektedir. Bu durum, hem zaman kaybına hem de olası yanlış yönlendirmeler sonucunda farklı problemlere yol açabilmektedir. Fakülte içerisinde yapılan gözlemler de birçok öğrencinin, bir sorunla karşılaştıklarında, etkinlikler hakkında bilgi sahibi olmadıklarında veya sınavlar sonrası süreçlerle ilgili bilgi almak istediklerinde yine yönetmelikler ve duyurulara başvurmadan doğrudan kendi bölümünden sorumlu idari personele başvurduğunu göstermektedir. Bunun yanında, öğrenciler idari personele başvurmadan önce Karabük Üniversitesi'nin internet sitesinden gerekli kaynaklara ulaşsa dahi, yönetmelikte öğrencinin aradığı bilgiden çok daha fazlası yer aldığından, tüm dökümanı okumak istemeyebilir. Ayrıca, yönetmeliklerin çok fazla hukuki terim içermesi, öğrencinin okuduğunu anlamasını zorlaştırabilir. Uluslararası Öğrenci Değerlendirme Programı'nın (Programme for International Student Assessment, PISA) 2022 yılı sonuçlarına bakıldığında, Türkiye'deki öğrencilerin %30'u orta uzunluktaki metinlerin ana düşüncesini belirleyebilmekte ve gerekli bilgilerin "açıkça" verildiği durumlarda orta uzunluktaki belirli detaylar üzerine düşünebilecek düzeydeyken, %30'luk bir bölümü ise bu düzeyin de altında kalmaktadır [1]. Bu verilere dayanarak üniversite içerisindeki eğitim-öğretim ve sınav yönetmeliklerinin uzunlukları ve yapıları ele alındığında öğrencilerin okuduklarını anlamakta zorlanması beklenilen bir durumdur. Bu gibi durumlarda öğrencilerin aradıkları bilgilere hızla ve kolayca ulaşmalarını sağlamak, hem onların zamanını verimli kullanmalarına hem de üniversite kaynaklarının etkinliğini artırmaya yardımcı olur. Ancak geleneksel bilgi erişim yöntemleri, özellikle yoğun bilgi içeren ve hukuki terimler barındıran dokümanlarda yetersiz kalabilir. İşte bu noktada, YZ destekli çözümler devreye girer. Literatürdeki araştırmalar, YZ destekli sohbet asistanlarının yükseköğretim

kurumlarında öğrencilere hızlı ve erişilebilir bilgi sağlama konusunda nasıl devrim yaratabileceğini ortaya koymaktadır. Örneğin, "Adoption of AI-Chatbots to Enhance Student Learning Experience in Higher Education" başlıklı çalışma [2], YZ destekli sohbet asistanlarının, öğrencilerin ihtiyaç duydukları bilgiye zahmetsizce ulaşmalarına imkan tanıyarak bilgi erişim sürecini nasıl kolaylaştırdığını vurgulamaktadır. Buna benzer şekilde, "Are Chatbots the Future of Higher Education? Unveiling Their Impact, Challenges, and Prospects" adlı çalışma [3], YZ tabanlı sohbet asistanlarının üniversite ortamında öğrencilerin sorularını anında yanıtlayarak, hem öğrenci deneyimini geliştirdiğini hem de yoğun dönemlerde idari personelin iş yükünü azalttığını belirtmektedir. Bu nedenle, hızla gelişen YZ çağına ayak uydurmak, sadece öğrencilerin bilgiye erişimini kolaylaştırmakla kalmayıp, üniversite ortamını daha dinamik ve erişilebilir hale getirmek için de kaçınılmaz bir gereklilik olarak öne çıkmaktadır.

Projenin Amacı

Bu projenin amacı, Karabük Üniversitesi Mühendislik Fakültesi bünyesinde, öğrenci ve personelin bilgi taleplerine anında yanıt verebilecek, danışmanlık sağlayabilecek bir sohbet asistanı sistemi geliştirmektir. Sistem, öğrencilere akademik takvim, güncel duyurular, etkinlikler ve okul yönetmelikleri gibi temel bilgileri hızlı bir şekilde sunarak bilgi akışını hızlandırmayı, fakülte bünyesindeki personelin iş yükünü azaltmayı ve öğrenci memnuniyetini artırmayı amaçlamaktadır. Ayrıca, çok dilli bir model ile yabancı öğrencilerin bilgiye daha kolay erişimini sağlamak da projenin önemli amaçları arasında yer almaktadır.

Hedefler:

- Anında Bilgi Sağlama: Sohbet asistanı, öğrenci ve personelin bilgi taleplerine hızlı bir şekilde yanıt vererek zamandan tasarruf sağlayacak ve bilgi akışını hızlandıracaktır.
- Kapsamlı Bilgi Desteği: Akademik takvim, yönetmelikler, etkinlikler ve kampüs hizmetleri gibi üniversite ile ilgili çeşitli bilgileri doğru ve hızlı bir şekilde sunacaktır
- Çoklu Dil Desteği: Yabancı öğrencilerin bilgiye daha kolay erişimini sağlayacak ve onların üniversite içindeki deneyimlerini iyileştirecektir.

- Dönemsel Sorulara Yanıt Verme: Sınav tarihleri ve kayıt dönemleri gibi zamanla değişen bilgileri güncel tutarak öğrencilere doğru zamanda, doğru bilgiyi sunacaktır.
- Kullanıcı Memnuniyetini Artırma: Sohbet asistanının kullanıcı dostu arayüzü ve hızlı yanıt kapasitesi sayesinde öğrenci ve personelin memnuniyetini artıracak, bilgiye erişim sürecini kolaylaştıracaktır.
- Doğru Kaynağa Yönlendirme: Sistem, cevap veremediği sorular karşısında kullanıcıyı doğru kaynaklara yönlendirerek bilgiye ulaşma sürecini etkin bir şekilde destekleyecektir.

LİTERATÜR TARAMASI

Son yıllarda büyük dil modelleri (LLM) ve doğal dil işleme (NLP) teknolojilerindeki gelişmeler, kullanıcıların doğal dilde sorular sorarak anlamlı ve bağlamsal olarak doğru yanıtlar alabildiği sanal asistan sistemlerinin farklı alanlarda yaygınlaşmasını sağlamıştır. Eğitim kurumlarında bu sistemler, yalnızca öğrenme süreçlerini desteklemek için değil, aynı zamanda öğrencilerin idari ve akademik bilgilere daha hızlı erişimini sağlamak amacıyla da kullanılmaktadır.

Labadze ve arkadaşları (2023), yapay zekâ tabanlı sohbet botlarının öğrencilerin ihtiyaç duyduğu bilgilere hızlıca ulaşabilmelerini sağladığını; Altun ve Seferoğlu (2024) ise bu sistemlerin eğitim ortamlarında rehberlik ve yönlendirme işlevi gördüğünü vurgulamıştır. [4-5]

Ancak Karabük Üniversitesi özelinde değerlendirildiğinde, öğrencilerin sınav tarihleri, yönetmelikler, etkinlikler ve duyurular gibi temel bilgilere erişiminde hâlen geleneksel yöntemlerin kullanıldığı görülmektedir. Bu nedenle geliştirilen sanal asistan uygulaması, üniversiteye özgü güncel ve resmi bilgilerin öğrencilere hızlı, doğru ve kolay erişimle sunulmasını sağlayarak önemli bir ihtiyaca çözüm sunmayı hedeflemektedir.

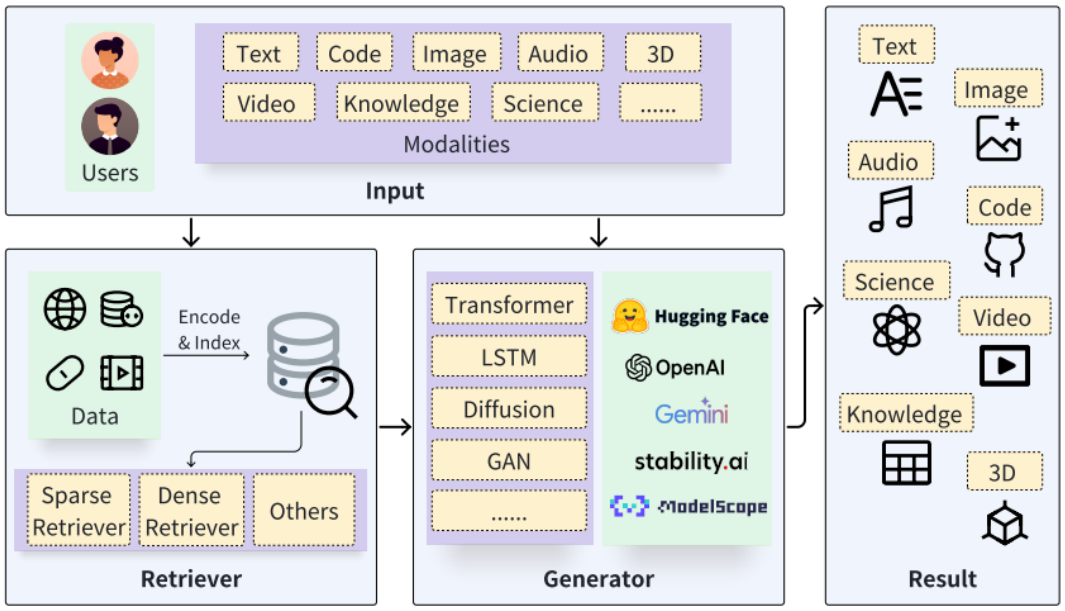
MATERYAL-METOT

Bu bölümde projede kullanılan teknikler ve araçlar hakkında yapılan literatür taraması sonucu elde edilen bilgiler ve teknik detaylar aşağıda yer almaktadır.

Retrieval-Augmented Generation (RAG) Sistemlerinin Kullanımı

Büyük dil modellerinin sınırlı bilgi kesitine (genellikle eğitim verisiyle sınırlı ve güncelliği kısıtlı) sahip olması, **bilgiye dayalı** sorularda yetersiz kalmalarına ve uydurma (halüsinasyon) üretmelerine yol açabilmektedir [6]. Bu sorunu aşmak için geliştirilen **RAG** yaklaşımları, dil modellerini harici bilgi kaynaklarıyla destekleyerek yanıtların güncel ve doğrulanabilir olmasını hedefler.

RAG Mimarisi



Şekil 1 - RAG Şeması [7]

RAG, büyük dil modeli tabanlı yapay zekâ sistemlerinin harici bilgi kaynaklarından yararlanarak daha isabetli ve güncel yanıtlar üretmesini sağlayan bir mimari yaklaşımdır. RAG mimarisi, bir **bilgi getirici** bileşen (retriever) ile bir **metin üretici model**in (generator) entegre çalışmasından oluşur: Kullanıcı sorgusuyla ilişkili dokümanlar veya veriler öncelikle bir bilgi tabanından otomatik olarak çekilir ve ardından bu içerik, modelin girişine eklenerek modele o ana kadarki eğitim verilerinde bulunmayan güncel veya özel alan bilgisi sağlanır. Bu sayede dil modeli, kendi eğitim verilerinde olmayan gerçeklerle de beslendiğinden, özellikle kuruma

özgü ya da güncel bilgiler gerektiren konularda daha tutarlı ve doğru çıktılar üretebilir. RAG yaklaşımı, model cevaplarının güvenilir bilgi kaynaklarına dayandırılmasını sağladığı için, soru-cevap sistemleri ve destek amaçlı sohbet botları gibi uygulamalarda yaygın olarak kullanılmaktadır [8-9].

RAG sistemlerinin kullanımına dair literatürde öne çıkan noktalar şöyledir:

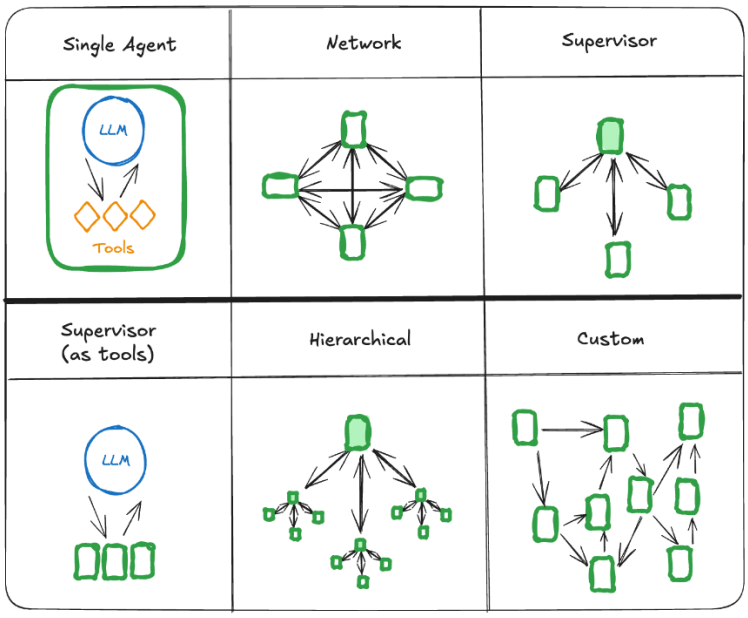
- **RAG Kavramı ve Yararları:** Lewis ve arkadaşlarının (2020) sunduğu çalışmada, RAG modeli ilk kez bilgi yoğun NLP görevlerinde tanıtılmıştır. Bu yaklaşıma göre, büyük bir dil modeli (parametrik bellek) bir **dış bilgi tabanından** (ör. Wikipedia) gelen doküman parçalarını arayıp alarak yanıt üretir. RAG, geleneksel salt parametrik modellere kıyasla açık uçlu soru-cevap görevlerinde duruma göre **son teknoloji başarımı** yakalamış; üretilen metinlerin daha spesifik, çeşitli ve gerçeklere dayalı olduğunu göstermiştir [10]. Böylelikle RAG, dil modellerinin kendi eğitim verilerinde bulunmayan güncel veya alan özelindeki bilgilere erişmesini mümkün kılarak, halüsinasyon eğilimini azaltmakta ve yanıtların güvenilirliğini artırmaktadır [11].
- **Uygulama ve Mühendislik Boyutu:** RAG sistemleri akademik araştırmaların yanı sıra endüstride de yapay zekâ destekli sohbet botlarında hızla benimsenmektedir. NVIDIA araştırmacıları tarafından yakın zamanda sunulan bir çalışma, kurumsal ortamlarda RAG tabanlı sohbet botları geliştirmenin **içerik güncelliği**, uygun mimari tasarımı, maliyet-etkinlik, test süreçleri ve güvenlik gibi bir dizi kritik boyutu olduğunu vurgulamıştır. Etkin bir RAG sistemi kurmak için vektör veritabanlarından ilgili dokümanların çekilmesi, sorguların yeniden biçimlendirilmesi, sonuçların yeniden sıralanması ve her adımda istemlerin (prompt) dikkatlice tasarlanması gibi pek çok mühendislik adımı gerektiği belirtilmektedir [12]. Bu da RAG'nin yalnızca bir model değil, uçtan uca bir **boru hattı (pipeline)** olarak ele alınması gerektiğine işaret eder. Sonuç olarak, doğru uygulandığında RAG, büyük dil modellerini yeniden eğitmeye gerek kalmaksızın harici bilgiyi entegre ederek daha güncel ve bağlamsal olarak doğru yanıtlar üretmeye olanak tanıyan güçlü bir yöntemdir [13].

LangGraph ve Çok Ajanlı Sistem Mimarileri

Sohbet botlarının ve genel olarak dil modeli tabanlı uygulamaların karmaşık görevleri yerine getirebilmesi için **çok ajanlı (multi-agent) mimariler** giderek önem kazanmaktadır. LangChain çatısı altında geliştirilen **LangGraph** kütüphanesi, birden fazla dil modelinin veya aracın düğümler (ajanlar) olarak tanımlandığı ve bir grafik yapısı halinde birbirine bağlandığı esnek bir kontrol akışı sunmaktadır [14].

LangGraph

LangGraph, LangChain ekibi tarafından geliştirilen açık kaynaklı bir yapay zeka ajan çerçevesidir. Bu teknoloji, büyük dil modellerini kullanarak çok adımlı veya çok etmenli (multi-actor) iş akışlarını bir grafik yapısı biçiminde tanımlamaya ve yürütmeye olanak tanır [15]. LangGraph, düşük seviye bir orkestrasyon kütüphanesi olarak birden fazla yapay zeka ajanı ya da bileşeninin etkileşimlerini denetleyip yönlendirebilmeyi sağlar. Bununla birlikte, özelleştirilebilir mimariler, uzun süreli bellek (durum takibi) ve insan denetimi (human-in-the-loop) gibi özellikler sunarak karmaşık görevlerin güvenilir şekilde gerçekleştirilmesine imkân tanır. Bu sayede geliştiriciler, her bir ajanın belirli bir rol üstlendiği ve birlikte çalıştığı ölçeklenebilir çoklu ajan sistemleri oluşturabilirler [16].



Şekil 2 - LangGraph kütüphanesi ile oluşturulabilen farklı çoklu ajan sistemleri [17].

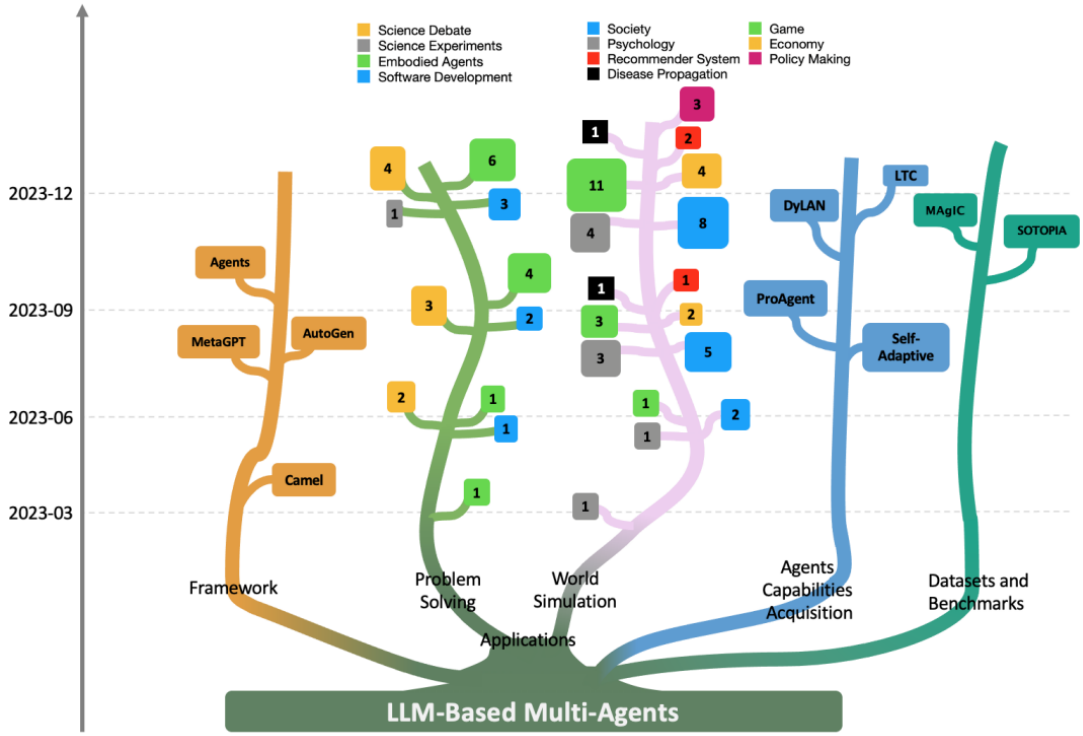
Bu yapı ile ilgili teknik açıklamalar ve literatürdeki bir değerlendirme aşağıda sunulmuştur:

Çok Ajanlı Mimari Nedir: LangGraph'in geliştiricileri, "çok ajanlı" terimini her biri bağımsız bir görev için özelleşmiş istemlere ve hatta farklı dil modeline sahip birden fazla **dil modeli aktörünün** birlikte çalışması olarak tanımlamaktadır. Bu yaklaşımda her bir ajan bir grafik düğümü olup, ajanlar arası bilgi akışı kenarlar üzerinden kontrollü bir şekilde gerçekleşir. Böyle bir yapı, klasik bir **durum makinesi** gibi düşünülebilir; her bir ajan bir durumu temsil ederken, aralarındaki geçişler grafik üzerinde tanımlanan akışa göre yürütülür [14].

Çok Ajanlı Tasarımın Avantajları: Birden fazla ajanın birlikte çalışması, tek bir ajanın tek başına üstesinden gelemeyeceği karmaşık problemleri alt görevlere bölerek çözme imkânı tanır. LangGraph blogunda vurgulandığı üzere, görev ve araçların gruplandırılarak farklı ajanlara paylaştırılması her bir ajanın **odaklandığı işe özel daha başarılı** olmasını sağlar; ayrıca her ajana ayrı istemler ve örnekler verilebildiğinden, gerektiğinde her biri farklı özelleşmiş veya ince ayarlı model kullanabilir. Bu modüler yapı geliştirme sürecini de kolaylaştırır: her bir ajanın performansı ayrı ayrı test edilip iyileştirilebilir, böylece bütünsel sistemi bozmadan parça bazında optimizasyon yapılabilir. Nitekim, *"bir ajan tek başına yetersiz kalıyorsa birden çok ajanın işbirliği işe yarar"* prensibiyle çok ajanlı tasarım konseptinin ortaya çıktığı ve pratikte başarılı sonuçlar verdiği bildirilmektedir [14].

Araştırma ve Uygulama Gelişmeleri: LLM tabanlı çok ajanlı sistemler, araştırma dünyasında da giderek daha fazla dikkat çekmektedir. Guo ve çalışma arkadaşlarının (2024) kapsamlı derlemesi, tek bir dil modelinin karar verici ajan olarak kullanıldığı ilk yaklaşımlardan, birden fazla LLM'nin ortak çalışmasına uzanan süreçte **karmaşık problem çözme** ve **dünya simülasyonu** gibi görevlerde önemli ilerlemeler kaydedildiğini aktarmaktadır. Bu çok ajanlı sistemlerde ajanların hangi ortamlarda çalıştığı, nasıl profillendiği, nasıl iletişim kurduğu ve yeteneklerinin nasıl geliştirildiği gibi sorular aktif araştırma konuları haline gelmiştir. Örneğin, farklı ajanların belirli roller üstlenerek (planlama, yürütme, değerlendirme gibi) bir arada çalıştığı senaryolar, zor görevlerde tek bir modelin performansını aşan sonuçlar verebilmektedir. Sonuç olarak, LangGraph gibi araçlar pratiğe çok ajanlı mimarileri getirmekte, akademik çalışmalar da bu yaklaşımların ilkelerinden doğan yeni yöntem

ve protokolleri incelemektedir. Çok ajanlı mimari, geliştirilen sanal asistanın bağlamsal tutarlılığını ve görev başarısını artırmada önemli bir bileşen olarak görülmektedir [18].



Şekil 3 - LLM Tabanlı Çoklu Ajanlar. Her düğüm üzerindeki sayı, söz konusu makalenin yayımlandığı dönemde o kategoriye ait makale sayısını ifade etmektedir [18].

UYGULAMA

Genel Sistem Mimarisi

Bu çalışmada geliştirilen üniversite sohbet botu, modern yapay zeka teknolojileri ve web teknolojilerinin bir arada kullanıldığı çok katmanlı bir mimariye sahiptir.

Sistem mimarisi üç ana bileşenden oluşmaktadır:

1. **Ön Uç (Frontend)**: React kütüphanesi kullanılarak geliştirilen, kullanıcıların sistemle etkileşime geçtiği arayüz.

2. **Arka Uç (Backend):** FastAPI çerçevesi ile geliştirilmiş, istemci isteklerini işleyen ve yapay zeka bileşenlerini yöneten API hizmetleri.

3. **Yapay Zeka Modülleri:** Büyük Dil Modelleri (LLM), vektör veritabanları ve yapay zeka araçlarını içeren bilgi işleme katmanı.

RAG Yaklaşımı

Geliştirilen sistem, temel olarak Retrieval Augmented Generation (RAG) mimarisi üzerine inşa edilmiştir. RAG, büyük dil modellerinin mevcut bilgilerini, harici veri kaynaklarından alınan bilgilerle zenginleştirerek daha doğru ve güncel yanıtlar üretmesini sağlayan bir yaklaşımdır. Bu yaklaşımın temel adımları şunlardır:

1. Veri Toplama ve İşleme: Üniversite ile ilgili bilgiler, fakülte ve bölüm web sitelerinden toplanarak yapılandırılmış bir formatta saklanır.

2. Vektör Dönüşümü: Toplanan metinler, OpenAI'nin "text-embedding-3-large" modeli kullanılarak vektör temsillerine dönüştürülür.

3. Vektör Veritabanı: Elde edilen vektörler, ChromaDB vektör veritabanında depolanır ve benzerlik aramaları için indekslenir.

4. Bilgi Çıkarımı: Kullanıcı soruları vektöre dönüştürülerek, en alakalı bilgiler veritabanından çıkarılır.

5. Yanıt Üretimi: Çıkarılan bilgiler, Büyük Dil Modeline (LLM) bağlam olarak sunularak, soruya özel ve doğru yanıtlar üretilir.

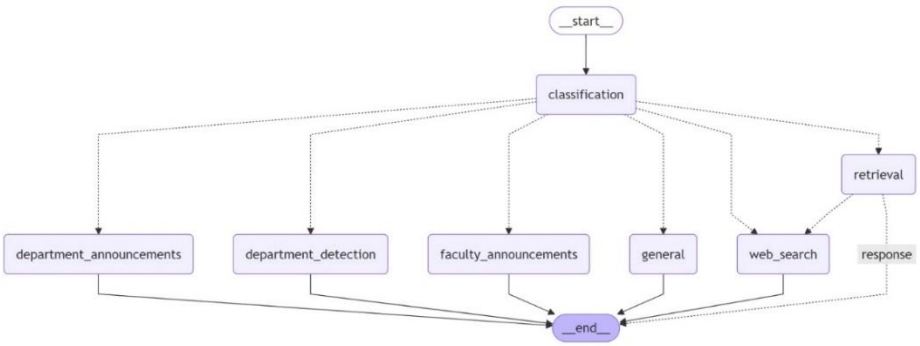
Sistemimizde kullanılan vektör veritabanı, aşağıda görülen kod parçasıyla oluşturulmuştur:

```python
def create_vectordb():
    load_dotenv()

    dosya_adi = f"C:/Users/{username}/Desktop/Vectordb.txt"
    paragraflar = paragraf_okuyucu(dosya_adi)

    documents = []
    for paragraph in paragraflar:
        documents.append(
            Document(
                page_content=paragraph,
                metadata={"source": paragraph.split(':', 1)[0]}
            )
        )

    vectorstore = Chroma.from_documents(
        documents=documents,
        embedding=OpenAIEmbeddings(model="text-embedding-3-large")
    )

    return vectorstore
```

Şekil 4 - Vektör Veritabanını Oluşturan Kod Parçası

LangGraph Temelli Akış Kontrolü

Sistemin merkezinde, LangGraph kütüphanesi kullanılarak geliştirilen yönlendirilmiş grafik yapısı bulunmaktadır. Bu grafik, kullanıcı sorularını farklı işleme aşamalarına yönlendirerek, sorunun içeriğine göre en uygun yanıt stratejisini belirlemektedir. Grafik akışı aşağıda gösterilmiştir.



Şekil 5 - Kullanıcının Sorgusunu İşleyen Grafik Yapısı

Sistemde kullanılan ana grafik bileşenleri şunlardır:

1. Sınıflandırma (Classification) Düğümü: Kullanıcı sorusunu analiz ederek kategori belirler.

2. Bilgi Çıkarımı (Retrieval) Düğümü: Vektör veritabanından ilgili bilgileri çeker.

3. Duyuru ve Etkinlik Düğümleri: Fakülte ve bölüm duyurularını işler.

4. Web Arama Düğümü: Veritabanında bilgi bulunamadığında web sitesinden bilgi çeker.

5. Genel Yanıt Düğümü: Basit sorulara ve selamlaşmalara yanıt verir.

Bu grafik yapısı, aşağıdaki gibi oluşturulmuştur:

```python
def create_agent():

    workflow = StateGraph(AgentState)


    workflow.add_node("classification", classification)
    workflow.add_node("retrieval", retrieval)
    workflow.add_node("general", general)
    workflow.add_node("faculty_announcements", faculty_announcements)
    workflow.add_node("department_announcements", department_announcements)
    workflow.add_node("department_detection", department_detection)
    workflow.add_node("web_search", web_search)


    workflow.set_entry_point("classification")


    workflow.add_node("response", lambda x: x)


    workflow.add_conditional_edges(
        "classification",
        lambda x: x["next"],
        {
            "general": "general",
            "retrieval": "retrieval",
            "faculty_announcements": "faculty_announcements",
            "department_announcements": "department_announcements",
            "department_detection": "department_detection",
            "web_search": "web_search"
        }
    )


    workflow.add_edge("general", "response")
    workflow.add_edge("retrieval", "response")
    workflow.add_conditional_edges(
        "retrieval",
        lambda x: "web_search" if x["retrieval_failed"] else "response",
        {
            "web_search": "web_search",
            "response": "response"
        }
    )


    workflow.add_edge("web_search", "response")
    workflow.add_edge("faculty_announcements", "response")
    workflow.add_edge("department_announcements", "response")
    workflow.add_edge("department_detection", "department_announcements")


    return workflow.compile()
```

Şekil 6 - LangGraph Grafik Yapısını Oluşturan Kod Parçası

Soru Sınıflandırma Mekanizması

Sistemde, kullanıcı sorularının içeriğini anlayarak doğru işleme sürecine yönlendirmek için gelişmiş bir sınıflandırma mekanizması kullanılmıştır. Bu mekanizma, soruları beş ana kategoriye ayırmaktadır:

1. GENEL_FAKULTE: Mühendislik Fakültesi hakkında genel bilgi soruları.

2. BOLUM_SPESIFIK: Belirli bir mühendislik bölümüyle ilgili spesifik sorular.

3. DUYURU_ETKINLIK: Fakülte veya bölümlerin duyuruları veya etkinlikleri hakkındaki sorular.

4. RETRIEVAL: Özel bilgiler gerektiren ve vektör veritabanından yanıtlanması gereken sorular.

5. GENEL: Genel konuşma, selamlaşma veya yukarıdaki kategorilere girmeyen üniversite ile ilgili basit sorular.

Sınıflandırma süreci şu şekilde gerçekleştirilmektedir:

```python
def classify_query(question: str) -> str:



    llm = create_llm()
    prompt = ChatPromptTemplate.from_template(CLASSIFICATION_TEMPLATE)


    chain = prompt | llm


    result = chain.invoke({"query": question})


    response = result.content.strip().upper()

    if "GENEL_FAKULTE" in response:
        return "GENEL_FAKULTE"
    elif "BOLUM_SPESIFIK" in response:
        return "BOLUM_SPESIFIK"
    elif "DUYURU_ETKINLIK" in response:
        return "DUYURU_ETKINLIK"
    elif "RETRIEVAL" in response:
        return "RETRIEVAL"
    else:
        return "GENEL"
```

Şekil 7 - Sınıflandırma Sistemini Gerçekleştiren Kod Parçası

Ayrıca, hangi spesifik bölümle ilgili soru sorulduğunu belirlemek için bölüm algılama mekanizması geliştirilmiştir:

```python
def detect_department(question: str) -> str:

    department_key = find_department_from_text(question)
    if department_key:
        return department_key

    llm = create_llm()

    department_list = "\n".join([f"- {name}" for name in DEPARTMENTS.values()])

    prompt = ChatPromptTemplate.from_template(DEPARTMENT_DETECTION_TEMPLATE)

    chain = prompt | llm

    result = chain.invoke({"question": question, "departments": department_list})
```

Şekil 8 - Bölümler Hakkında Gelen Sorunun Hangi Bölüme Ait Olduğunu Belirleyen Kod Parçası

Web Scraping ve Güncel İçerik Getirme

Sistemin en önemli özelliklerinden biri, üniversitenin web sitesinden güncel bilgileri otomatik olarak çekebilmesidir. Bu sayede kullanıcılara en güncel duyurular, etkinlikler ve haberler sunulabilmektedir. Web kazıma (scraping) işlemleri için Python'un requests ve BeautifulSoup kütüphaneleri kullanılmıştır.

Sistem, aşağıdaki içerik türlerini otomatik olarak çekebilmektedir:

1. Fakülte genel duyuruları
2. Bölüm spesifik duyuruları
3. Üniversite geneli etkinlikler
4. Akademik takvim bilgileri
5. Spesifik web sayfası içerikleri

Web kazıma işlemleri, aşağıdaki fonksiyonlar aracılığıyla gerçekleştirilmektedir:

```
def get_announcements(department_key: str = None) -> List[Dict]: ···
def get_events(department_key: str = None) -> List[Dict]: ···
def get_latest_announcements(department_key: str = None) -> str: ···
def get_latest_events(department_key: str = None) -> str: ···
def get_faculty_announcements() -> str: ···
def search_website(query: str, department_key: str = None) -> str: ···
def get_announcement_details(url: str) -> Dict: ···
def get_event_details(url: str) -> Dict: ···
```

Şekil 9 - Web Kazıma İşlemlerini Gerçekleştiren Fonksiyonlar

Ön Uç Geliştirme

Sistemin kullanıcı arayüzü, modern web teknolojileri kullanılarak geliştirilmiştir. Arayüz, kullanıcı deneyimini en üst düzeye çıkarmak için tasarlanmış olup, aşağıdaki teknolojiler kullanılmıştır:

1. React: Kullanıcı arayüzünün temel yapısını oluşturan JavaScript kütüphanesi.

2. CSS3: Arayüzün stillendirilmesi için kullanılan teknoloji.

3. React Icons: Arayüzdeki ikonlar için kullanılan kütüphane.

API Hizmetleri

Ön uç ve yapay zeka bileşenleri arasındaki iletişim, FastAPI çerçevesi kullanılarak geliştirilmiş REST API üzerinden sağlanmaktadır. API yapısı, aşağıdaki ana bileşenlerden oluşmaktadır:

1. Router'lar: İstemci isteklerini ilgili hizmetlere yönlendiren bileşenler.

2. Şemalar: Veri transferi için kullanılan Pydantic modelleri.

3. Hizmetler: İş mantığını içeren servis sınıfları.

4. Konfigürasyon: Sistem ayarlarını içeren yapılandırma dosyaları.

Ana `/api/chat` endpoint'i, kullanıcı sorularını işleyerek yanıt üretmektedir:

```python
@router.post("/chat")
async def chat(request: QuestionRequest, fastapi_request: Request):
    try:
        question = request.question
        session_id = request.session_id

        history = request.history

        logger.info(f"[{session_id}] Soru: {question}")
        result = get_answer_with_graph(question, session_id, history)

        if "error" in result:
            logger.error(f"[{session_id}] Hata: {result['error']}")
            raise HTTPException(
                status_code=StatusCodes.INTERNAL_SERVER_ERROR,
                detail={
                    'message': ErrorMessages.PROCESSING_ERROR,
                    'error': result['error']
                }
            )

        logger.info(f"[{session_id}] Yanıt oluşturuldu")
        return {
            "answer": result["answer"],
            "session_id": session_id,
            'status_code': StatusCodes.SUCCESS
        }
    except Exception as e:
        logger.error(f"Chat router hatası: {str(e)}")
        raise HTTPException(
            status_code=StatusCodes.INTERNAL_SERVER_ERROR,
            detail= {
                'message': ErrorMessages.PROCESSING_ERROR,
                'error': str(e)
            }
        )
```

Şekil 10 - Kullanıcı Sorgularını İşleyen "/chat" Uç Noktası

Prompt Mühendisliği

Sistemin yanıt kalitesini ve doğruluğunu artırmak için gelişmiş promptlar tasarlanmıştır. Bu promptlar, LLM'in doğru bağlamda ve istenen formatta yanıt üretmesini sağlamaktadır. Sistemde kullanılan promptlara örnekler:

1. Genel Sistem Promptu: LLM'in temel rolünü ve davranışlarını tanımlar.

2. Retrieval Promptu: Vektör veritabanından çıkarılan bilgileri içeren prompt.

3. Duyuru ve Etkinlik Promptu: Web'den çekilen güncel bilgileri içeren prompt.

4. Sınıflandırma Promptu: Soruları kategorize etmek için kullanılan prompt.

5. Bölüm Algılama Promptu: Soruda hangi bölümden bahsedildiğini belirlemek için kullanılan prompt.

Dağıtım ve Performans Optimizasyonu

Sistem, performansı ve kullanılabilirliği artırmak için çeşitli optimizasyon teknikleri ile geliştirilmiştir:

1. Önbellek (Caching): LLM ve VektörDB örneğini önbellekte tutarak tekrarlı oluşturmaları engelleme.

2. Asenkron İşlemler: FastAPI'nin asenkron yapısı kullanılarak eşzamanlı istek işleme.

3. Lazım Olduğunda Yükleme (Lazy Loading): API kaynaklarının ihtiyaç duyulduğunda yüklenmesi.

4. Oturum Yönetimi: Kullanıcı oturumlarını ve mesaj geçmişlerini verimli şekilde saklama.

5. Bileşen MemoRizasyonu: React bileşenlerinin gereksiz yeniden render'larını önleme.

Sistem, Uvicorn WSGI sunucusu üzerinde çalıştırılmaktadır:

Şekil 11- Uygulamayı Uvicorn İle Ayağa Kaldıran Kod Parçası

DENEYSEL SONUÇLAR

Geliştirilen sanal asistan sistemi, Karabük Üniversitesi Mühendislik Fakültesi öğrencilerinin bilgiye erişim süreçlerini kolaylaştırmak amacıyla farklı senaryolarda test edilmiştir. Testlerde kullanılan promptlar, öğrencilerin en sık yönelttiği sorulara (örneğin: sınav tarihleri, yönetmelik maddeleri, ders geçme kuralları, danışman bilgileri, etkinlik detayları) dayalı olarak belirlenmiştir. Bu senaryolar, hem okulun mevcut web sitesindeki bilgilerle hem de sistemin sunduğu yanıtlarla karşılaştırılmıştır.

Testler sonucunda sistemin güçlü yönleri şunlar olmuştur:

- Duyuru ve etkinliklerle ilgili sorularda sistem, güncel verileri başarıyla çekerek kullanıcıya doğru bilgi sunmuştur.
- Akademik takvim veya yönetmelik gibi belgelerde geçen belirli bilgilere ulaşım konusunda vektör tabanlı bilgi çıkarımı mekanizması oldukça etkili çalışmıştır.
- Bölüm algılama ve sınıflandırma yapısı, sorunun ait olduğu kategoriyi doğru belirleyerek en uygun iş akışını tetiklemiştir.
- Sistem, LangGraph temelli çok ajanlı yapı sayesinde, bilgi eksikliği durumunda web arama bileşenini devreye alarak yanıt üretmeye devam edebilmiştir.

Ancak sistemin zayıf yönleri de gözlemlenmiştir:

- Bazı çok uzun ve çok anlamlı promptlar, modelin bağlamı korumasını zorlaştırmış, bu da bazı yanıtlarda anlamsal tutarsızlıklara neden olmuştur.
- Nadiren geçen terimler veya özel adlar (örneğin: belirli bir hocanın tam adı) vektör veri tabanında yetersiz temsil edildiğinden, yanlış veya eksik yanıtlar üretilebilmiştir.
- Web kazıma bileşeni, web sitesinde tasarımsal değişiklik olduğunda veri çekmekte sorun yaratabilir ve bu durum modelin yanıt oluşturamamasına sebep olabilir.

Bu deneysel süreçler, sistemin gerçek dünya kullanımında çoğu senaryoya uygun şekilde çalıştığını, ancak bazı sınır durumlarda hala iyileştirmeye açık alanlar bulunduğunu göstermiştir.

TARTIŞMA

Bu çalışmada geliştirilen Karabük Üniversitesi Bilgilendirme Asistanı (KÜBAS), literatürde sunulan yapay zekâ destekli sanal asistan sistemlerinin eğitim kurumlarında bilgiye erişimi hızlandırmadaki etkilerini destekler nitelikte sonuçlar vermiştir. KÜBAS'ın geliştirilmesiyle birlikte, Karabük Üniversitesi öğrencilerinin sınav tarihleri, yönetmelikler ve etkinlikler gibi bilgilere doğrudan ve hızlı bir şekilde erişebilmesi sağlanmıştır. Böylece literatürde belirtilen teorik katkılar, somut bir uygulama üzerinden başarıyla doğrulanmıştır.

Deneysel sonuçlar, sistemin genel olarak başarılı olduğunu ve büyük dil modelleri ile RAG mimarisinin entegrasyonunun bilgi güncelliği ve doğruluğu açısından büyük katkılar sağladığını göstermiştir. Ancak, uzun ve karmaşık sorularda modelin bağlamı korumakta zorlanması, vektör veritabanındaki belirli bilgi eksiklikleri ve web kazıma süreçlerinde yaşanan zorluklar gibi sınırlamalar da gözlemlenmiştir. Bu durum, mevcut literatürde RAG sistemlerinin mühendislik ve veri bütünlüğü açısından dikkatle yönetilmesi gerektiği yönündeki bulgularla uyumludur.

Geliştirilen sistemin literatüre katkısı, üniversiteye özgü güncel bilgilere dinamik erişim sağlayan, çok ajanlı bir yapıyı başarıyla entegre eden ölçeklenebilir bir çözüm sunmasıdır. Özellikle LangGraph kullanımı ile sağlanan çok ajanlı yapı, sistemin farklı bilgi kaynaklarından doğru veriyi seçebilmesini ve eksik bilgi durumlarında alternatif akışlar başlatabilmesini mümkün kılarak, literatürde önerilen çok ajanlı yaklaşımların pratikteki etkinliğini ortaya koymuştur.

Gelecek çalışmalarda, sistemin zayıf yönlerini geliştirmek amacıyla bazı iyileştirmeler yapılabilir. Özellikle:

- Uzun ve çok anlamlı sorularda bağlamı koruyacak daha gelişmiş prompt mühendisliği tekniklerinin geliştirilmesi,
- Vektör veritabanının zenginleştirilmesi,

- Kullanıcı geri bildirimlerine dayalı sürekli model iyileştirme döngüsünün kurulması, sistemin güvenilirliğini ve kullanıcı memnuniyetini artıracaktır.

Bu öneriler doğrultusunda ilerleyen çalışmalar, KÜBAS'ın üniversite çapında daha geniş bir kullanıcı kitlesine ulaşmasını ve yapay zekâ tabanlı bilgilendirme sistemlerinin eğitim alanındaki etkinliğinin daha da artırılmasını sağlayacaktır.

KAYNAKÇA

[1] Millî Eğitim Bakanlığı. (2024). PISA 2022 Türkiye Raporu.

[2] N. Sandu and E. Gide, "Adoption of AI-Chatbots to Enhance Student Learning Experience in Higher Education in India," 2019 18th International Conference on Information Technology Based Higher Education and Training (ITHET).

[3] K. Oqaidi, S. Aouhassi and K. Mansouri, "Are Chatbots the Future of Higher Education? Unveiling Their Impact, Challenges, and Prospects," 2024 4th International Conference on Innovative Research in Applied Science, Engineering and Technology (IRASET).

[4] Labadze, L., Grigolia, M., Machaidze, L. (2023). Role of AI chatbots in education: systematic literature review. International Journal of Educational Technology. https://educationaltechnologyjournal.springeropen.com/articles/10.1186/s41239-023-00426-1

[5] Altun, E., & Seferoğlu, S. S. (2024). Eğitimde sohbet robotlarının kullanımına yönelik literatür incelemesi. İnönü Üniversitesi Eğitim Fakültesi Dergisi. https://dergipark.org.tr/tr/pub/inuefd/issue/86749/1397735

[6]: OpenAI. (2021). *WebGPT: Browser-assisted question-answering with human feedback*. arXiv:2112.09332. https://arxiv.org/abs/2112.09332

[7] Zhao, P., Zhang, H., Yu, Q., Wang, Z., Geng, Y., Fu, F., Yang, L., Zhang, W., Jiang, J., Cui, B. (2024). Retrieval-Augmented Generation for AI-Generated Content: A Survey. arXiv:2402.19473. https://arxiv.org/abs/2402.19473

[8] https://www.databricks.com/glossary/retrieval-augmented-generation-rag

[9] https://www.ibm.com/architectures/patterns/genai-rag

[10] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., Kiela, D. (2021). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. arXiv:2005.11401. https://arxiv.org/abs/2005.11401

[11] Oreški, D., Vlahek, D. (2024). Retrieval Augmented Generation in Large Language Models: Development of AI Chatbot for Student Support

[12] NVIDIA. (2024). FACTS About Building Retrieval Augmented Generation-based Chatbots. arXiv:2407.07858. https://arxiv.org/abs/2407.07858

[13] https://www.codiste.com/agentic-retrieval-augmented-generation-rag-ai-workflow

[14] https://blog.langchain.dev/langgraph-multi-agent-workflows/

[15] https://www.ibm.com/think/topics/langgraph

[16] https://github.com/langchain-ai/langgraph

[17] https://langchain-ai.github.io/langgraph/concepts/multi_agent/

[18] Guo, T., Chen, X., Wang, Y., Chang, R., Pei, S., Chawla, N. V., Wiest, O., & Zhang, X. (2024). Large language model based multi-agents: A survey of progress and challenges. arXiv:2402.01680. https://arxiv.org/abs/2402.01680