



Teknoloji Fakültesi

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

Büyük Dil Modellerinde Yönerge Kullanımı Optimizasyonu ve Maliyet Etkin Sonuç Alma Yöntemleri Karşılaştırması

BİTİRME PROJESİ

Bilgisayar Mühendisliği Bölümü

DANIŞMAN

Doç. Dr. Buket Doğan

İSTANBUL, 2025

"

MARMARA ÜNİVERSİTESİ
TEKNOLOJİ FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

Marmara Üniversitesi Teknoloji Fakültesi Bilgisayar Mühendisliği Öğrencileri Burak Yalçın ve Yusuf Ünlü tarafından “**Büyük Dil Modellerinde Yönerge Kullanımı Optimizasyonu ve Maliyet Etkin Sonuç Alma Yöntemleri Karşılaştırması**” başlıklı proje çalışması, 19.06.2025 tarihinde savunulmuş ve jüri üyeleri tarafından başarılı bulunmuştur.

Jüri Üyeleri

Doç. Dr. Buket Doğan	(Danışman)	
Marmara Üniversitesi	
Prof. Dr. Ali Buldu	(Üye)	
Marmara Üniversitesi	
Arş. Gör. Asaf Karataş	(Üye)	
Marmara Üniversitesi	

İÇİNDEKİLER

1. GİRİŞ	2
1.1. Proje Çalışmasının Amacı ve Önemi	2
2. Yapay Zeka Nedir?	3
2.1. Doğal Dil İşleme (NLP) Nedir?.....	5
2.2. Büyük Dil Modelleri (LLM) Nedir?	7
2.3. Alandaki Güncel Sorunlar	8
2.3.1 Ölçeklenebilirlik Sorunu.....	9
2.3.2. Maliyet	9
2.3.3. Doğruluk	9
3. LLM MİMARİLERİ VE ÖĞRENME YAKLAŞIMLARI.....	10
3.1. Transformer Tabanlı Yapılar ve Mixture of Experts (MoE).....	10
3.2. Zero-shot, One-shot, Few-shot Öğrenme Nedir? Avantaj ve Dezavantajları	12
3.2.1. Zero Shot, One Shot, Few Shot Uygulaması.....	14
3.3. Pretraining ve Fine-tuning.....	16
3.4. Eğitim Yöntemleri: SFT, RLHF, PPO ve Distillation.....	17
4. OPTİMİZASYON VE MALİYET AZALTMA YÖNTEMLERİ.....	19
4.1. Prompt Mühendisliği ve Sıkıştırma Teknikleri	20
4.1.1. ChatGPT ile Etkileşimli Rol Yapma Deneyimi: Tek Oyunculu Dungeons & Dragons Uygulaması	20
4.1.2. Otomatik Raporlama İçin Etkili Prompt Mühendisliği Uygulaması.....	25
4.1.3. Prompt Sıkıştırma ve Özetleme Teknikleri	28
4.2. Token Sayısı Neden Önemlidir?.....	29
4.2.1 Token Maliyetleri Neye Göre Belirlenir	29

4.2.2 Context Window Sınırı	30
4.3. Sampling Teknikleri (Top-k, Top-p, Temperature).....	30
4.3.1 Temperature (Sıcaklık)	31
4.3.2. Top-k Sampling	31
4.3.3. Top-p (Nucleus Sampling)	31
4.4. Model Seçimi	32
4.5. Önbellekleme (Caching) ile Hızlandırma	34
4.6. Hafif Modeller	35
5. LLM PERFORMANS VE MALİYET KARŞILAŞTIRMASI.....	36
5.1. Kalite-Hız-Maliyet Üçgeni	36
5.2. Maliyet Hesaplamaları: Token Başına Ücret	38
6. Güncel Yaklaşımlar ve Gelecek Trendler	39
6.1. Retrieval-Augmented Generation (RAG) Sistemi.....	39
6.1.2. RAG Sistemi Uygulaması	41
6.2. AI Feedback, Constitutional AI ve Etik Kurallar	44
6.2.1. Constitutional AI (Anayasaya Dayalı Yapay Zeka).....	44
6.2.2 AI Feedback.....	45
6.2.3. Preference Modeling (Tercih Modelleme)	45
6.3. LLM’lerin Güncel Kalması İçin Teknikler (Retraining vs RAG).....	46
6.4. Güvenilirlik, Hallucination ve Yanıltıcı Bilgi Problemleri.....	48
6.4.1. Halüsinasyon (Hallucination) Nedir?	48
6.5. Açık Kaynak LLM’lerin Yükselişi.....	49
6.5.1. Açık Kaynak LLM’lerin Tanımı ve Temel İlkeleri.....	50
6.5.2. Yükselişi Tetikleyen Faktörler	50
6.5.3. Ticari ve Açık Kaynak LLM’lerin Karşılaştırılması	51

7. Bulgular ve Tartışma.....	52
8. Sonuçlar	53

ÖZET

Büyük Dil Modellerinde Yönerge Kullanımı Optimizasyonu ve Maliyet Etkin Sonuç Alma Yöntemleri Karşılaştırması.

Bu bitirme projesi çalışmasında, büyük dil modellerinin (Large Language Models – LLM) yönlendirici metinler (promptlar) ile nasıl daha etkili, verimli ve maliyet-etkin şekilde kullanılabileceği deneysel olarak araştırılmıştır. Çalışma kapsamında, farklı LLM mimarileri ve öğrenme yaklaşımları incelenmiş; özellikle prompt mühendisliği, model seçimi, RAG (Retrieval-Augmented Generation) entegrasyonu gibi optimizasyon stratejileri detaylı şekilde değerlendirilmiştir.

Proje süresince üç farklı uygulama senaryosu hayata geçirilmiştir. Bunlardan ilki, ChatGPT tabanlı tek oyunculu Dungeons & Dragons (D&D) simülasyonudur. Bu senaryoda yapay zekâ, detaylı yönergelerle bir oyun yöneticisi (Dungeon Master) gibi davranarak anlatısal ve sistemsel tutarlılık göstermiştir. İkinci uygulama, IT departmanlarında kullanılabilecek otomatik fonksiyonel gereksinim raporu üretimi üzerine odaklanmış ve yinelemeli prompt tasarımı sayesinde LLM’lerin yazılım analiz süreçlerinde nasıl kullanılabileceği gösterilmiştir. Üçüncü uygulamada ise açık kaynak modeller ve Supabase tabanlı bir altyapı kullanılarak, üniversite yönetmeliklerinden belge tabanlı bilgi çekimi yapan bir RAG sistemi geliştirilmiştir.

Uygulamalar üzerinden yapılan gözlem ve karşılaştırmalar, prompt mühendisliğinin model çıktıları üzerinde kritik etkiye sahip olduğunu; doğru model seçimi ve mimari yapılandırmayla LLM’lerin maliyet, hız ve kalite açısından optimize edilebileceğini göstermektedir. Bu çalışma, yalnızca teorik bir analiz değil, aynı zamanda pratik çözümler sunan, düşük kaynakla yüksek verim alınabilecek bir yol haritası olarak değerlendirilebilir.

Haziran, 2024

Öğrenciler

Burak Yalçın

Yusuf Ünlü

ŞEKİL LİSTESİ

Şekil 1. LLM Gelişim Diyagramı.....	6
Şekil 2. Mixture of Experts Diyagramı.....	11
Şekil 3. Deney Sonucu Üretilen Fonksiyonel Gereksinim Analiz Dokümanı	28
Şekil 4. Dokümanı parçalara bölüp vektöre dönüştüren kod.	42
Şekil 5. Kullanıcı girdisini alıp veritabanında vektörel eşleme yapan kod.....	43
Şekil 6. Sisteme Sorulan Devam Şartı Sorusu ve Yanıt	44
Şekil 7. Sisteme Sorulan GANO Sorusu ve Yanıt	44

TABLO LİSTESİ

Tablo 1. LLM'lerin Kullanıldığı Görevler	8
Tablo 2. Transformer ve MoE Karşılaştırması	12
Tablo 3. Zero Shot, One Shot, Few Shot Karşılaştırma Tablosu	13
Tablo 4. Few Shot Uygulaması Karşılaştırma Tablosu.....	16
Tablo 5. Model Eğitim Yöntemleri Karşılaştırma Tablosu.....	19
Tablo 6. Bazı LLM'lerin Context Window Limitleri	30
Tablo 7. Fazla Token Kullanmanın Riskleri Tablosu	30
Tablo 8. Temperature Değerleri Örnek Tablosu	31
Tablo 9. Sampling Teknikleri Karşılaştırma Tablosu	32
Tablo 10. Doğru Sampling Ayarı Seçim Tablosu	32
Tablo 11. Büyük Modellerin Özellikleri	33
Tablo 12. Küçük Modellerin Özellikleri	33
Tablo 13. Karar Verme Kriterleri Tablosu	34
Tablo 14. RAG ve Retraining Metodları Karşılaştırması.....	47

1. GİRİŞ

Yapay zeka ve doğal dil işleme teknolojilerinin son yıllarda gösterdiği gelişmeler, bilgiye erişim, metin üretimi ve insan-bilgisayar etkileşimi gibi alanlarda büyük bir devrime yol açmıştır. Özellikle büyük dil modelleri (Large Language Models - LLM), insan dilini anlama ve üretme konularında çığır açıcı başarılar sergilemiş ve akademik çalışmalardan endüstriyel uygulamalara kadar birçok alanda aktif olarak kullanılmaya başlanmıştır.

Bu modeller, büyük miktarda veri üzerinde eğitilmiş derin öğrenme tabanlı sistemlerdir ve metin oluşturma, özetleme, soru-cevap, çeviri gibi karmaşık görevleri yüksek doğrulukla yerine getirme kapasitesine sahiptir. Ancak bu güçlü yapılar, büyük hesaplama kaynakları gerektirdiği için maliyet, performans ve verimlilik açısından önemli bazı problemleri de beraberinde getirmiştir.

Günümüzde LLM'lerin ticari ve bilimsel kullanımı yaygınlaştıkça, daha az kaynakla daha kaliteli çıktı üretme ihtiyacı ortaya çıkmıştır. Bu da bizleri, prompt mühendisliği, model seçimi, optimizasyon teknikleri ve dış veri kaynaklarıyla entegrasyon gibi alanlarda çözüm arayışlarına yönlendirmiştir.

Bu çalışmada, büyük dil modellerinin yapısı, öğrenme biçimleri, optimizasyon teknikleri ve maliyet-etkin kullanım yöntemleri ayrıntılı bir şekilde ele alınacaktır. Ayrıca güncel yaklaşımlar, örnek karşılaştırmalar ve uygulama senaryoları üzerinden değerlendirmeler yapılarak hem akademik hem de endüstriyel alanda fayda sağlayacak sonuçlara ulaşılması hedeflenmektedir.

1.1. Proje Çalışmasının Amacı ve Önemi

Büyük dil modelleri (Large Language Models - LLM), yapay zekâ teknolojilerinin son yıllardaki en dikkat çekici gelişmelerinden biridir. GPT-4, Claude 3 ve LLaMA 3 gibi modeller, çok büyük veri kümeleri üzerinde eğitilerek dilin bağlamsal yapısını kavrayabilmekte ve insan benzeri metinler üretebilmektedir. Bu modeller; özetleme, soru-cevap, çeviri, metin oluşturma ve hatta kod üretimi gibi çok çeşitli görevleri yüksek doğruluk oranlarıyla gerçekleştirme kapasitesine sahiptir [1], [2].

Ancak bu güçlü yapay zekâ sistemlerinin, bazı temel sınırlamaları bulunmaktadır. Bu sınırlamaların başında yüksek işlem maliyetleri, ölçeklenebilirlik sorunları, ve verimlilik eksikliği gelmektedir. Büyük dil modelleri, özellikle üretim sırasında oldukça fazla işlem

gücü ve bellek tüketmektedir. Bu da hem donanım gereksinimlerini artırmakta hem de bulut tabanlı kullanımlarda kullanıcıya ciddi maliyetler doğurmaktadır [3].

Bu projenin temel amacı, büyük dil modellerinde yönlendirici metinlerin (promptların) daha etkili bir şekilde kullanılarak; hem model çıktılarının kalitesini artırmak, hem de maliyetleri minimize etmek için kullanılabilecek yöntemleri analiz etmektir. Bu doğrultuda özellikle aşağıdaki sorulara cevap aranacaktır:

- Bir LLM'e verilen promptun yapısı, çıktının doğruluğunu ve üretim maliyetini nasıl etkiler?
- Uzun ve karmaşık promptlar yerine daha kısa, etkili ve özetlenmiş yönlendirmeler kullanmak maliyet/performans dengesini nasıl değiştirir?
- Farklı prompt sıkıştırma ve özetleme teknikleri, farklı modeller üzerinde nasıl sonuçlar verir?
- Hangi modeller, hangi görevler için maliyet açısından daha uygundur?

Bu sorulara yanıt ararken, prompt mühendisliği, ön işleme (preprocessing), model seçimi, sampling stratejileri ve önbellekleme gibi optimizasyon teknikleri ele alınacaktır. Ayrıca bu tekniklerin farklı modeller üzerindeki etkileri karşılaştırmalı olarak değerlendirilecektir.

Çalışmanın bir diğer hedefi ise, günümüzde LLM'lerin kullanıldığı akademik ve endüstriyel senaryolara dair önerilerde bulunmaktır. Örneğin, bir müşteri hizmetleri sohbet robotu için maksimum doğruluk değil, hızlı ve ucuz üretim ön planda olabilirken; akademik bir bilgi tabanı sisteminde doğruluk en öncelikli kriter olabilir. Bu gibi kullanım senaryolarında en uygun stratejiler önerilecektir.

Sonuç olarak bu proje, LLM'lerin kullanımını hem teknik hem de ekonomik olarak daha sürdürülebilir hale getirmeyi amaçlamaktadır. Bu da yalnızca araştırma projeleri için değil, üretim ortamlarında da büyük avantajlar sağlayacaktır.

2. Yapay Zeka Nedir?

Yapay zeka (YZ), insan zekâsını taklit eden bilgisayar sistemlerinin geliştirilmesini ifade eden geniş bir bilim ve mühendislik alanıdır. Bu sistemler; öğrenme, akıl yürütme, problem çözme, örüntü tanıma, planlama ve doğal dil anlama gibi görevleri yerine

getirebilir. Yapay zekâ kavramı ilk kez 1956'da John McCarthy tarafından tanımlanmış olup, o tarihten itibaren hem teorik hem de pratik düzeyde çok büyük ilerlemeler kaydedilmiştir [4],[5],[6].

Yapay zekâ sistemleri, genel olarak üç temel kategoriye ayrılır:

- **Dar (Zayıf) Yapay Zeka (Narrow AI):** Sadece belirli bir görevi yerine getirmek üzere eğitilmiş sistemlerdir. Örneğin; öneri motorları, yüz tanıma yazılımları, spam filtreleri ve sohbet robotları.
- **Genel Yapay Zeka (AGI - Artificial General Intelligence):** İnsan seviyesinde bilişsel yeteneklere sahip, çok yönlü öğrenebilen sistemlerdir. Henüz araştırma düzeyindedir.
- **Süper Yapay Zeka (ASI - Artificial Superintelligence):** İnsan zekâsının tüm yeteneklerini aşan teorik bir yapıdır.

Günümüzde kullanılan sistemlerin çoğu dar yapay zekâ kapsamındadır. Bununla birlikte, GPT-4 gibi büyük dil modelleri, çok sayıda görevi bir arada gerçekleştirme becerileri sayesinde genel yapay zekâya yaklaşan sistemler olarak değerlendirilmektedir.

Yapay zekâ sistemleri, genellikle aşağıdaki teknolojilerle inşa edilir:

- **Makine Öğrenmesi (Machine Learning - ML):** Verilerden istatistiksel çıkarımlar yaparak kendiliğinden öğrenme sağlayan algoritmalar bütünü.
- **Derin Öğrenme (Deep Learning - DL):** Çok katmanlı yapay sinir ağları kullanarak görüntü, ses ve metin gibi karmaşık verileri işleyebilen bir makine öğrenmesi alt alanı.
- **Kurallara Dayalı Sistemler:** İnsan uzmanlığıyla oluşturulan mantıksal kurallar üzerinden çalışan sistemler.

Yapay zekânın modern uygulama alanları arasında şunlar öne çıkar:

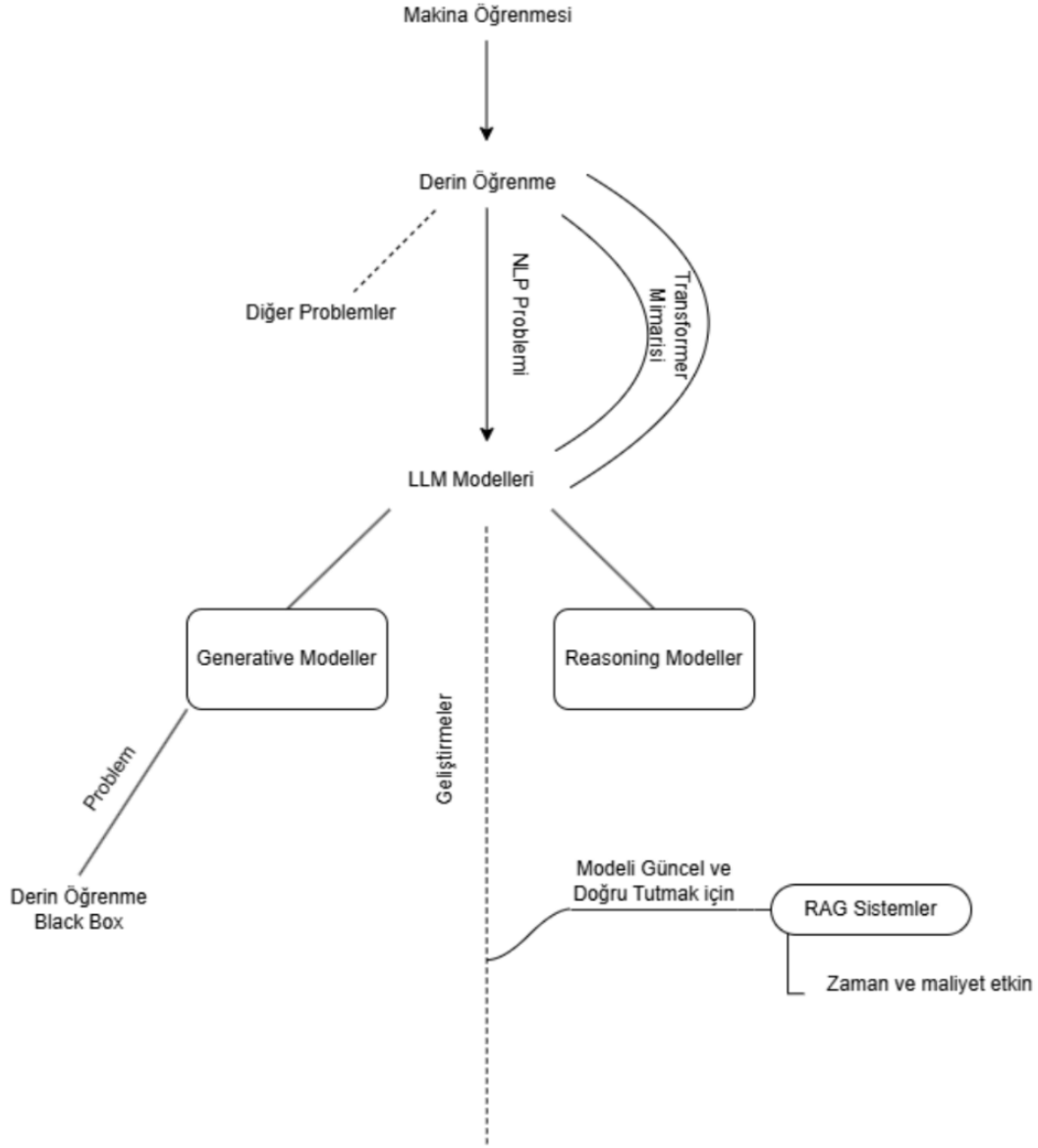
- Otomotiv (otonom araçlar),
- Finans (risk analizi, algoritmik işlem),
- Sağlık (hastalık teşhisi, ilaç geliştirme),
- Eğitim (kişiselleştirilmiş öğrenme),
- Hukuk (otomatik belge inceleme),
- Savunma ve güvenlik (anormallik tespiti, yüz tanıma).

Yapay zekânın en hızlı gelişim gösterdiği alanlardan biri de doğal dil işleme (NLP)'dir. İnsan dili gibi yapısal olmayan verilerin işlenmesi, yorumlanması ve üretilmesini sağlayan bu alt alan sayesinde LLM'ler gibi gelişmiş modeller ortaya çıkmıştır.

2.1. Doğal Dil İşleme (NLP) Nedir?

Doğal Dil İşleme (Natural Language Processing - NLP), bilgisayarların insan dilini anlamasını, işlemesini ve üretmesini sağlayan bir yapay zekâ alt alanıdır. Temel amacı; insanlar tarafından üretilen yazılı ve sözlü dili, bilgisayarların anlayabileceği biçime dönüştürmek ve buna uygun çıktılar üretmektir. NLP, hem dil bilimi hem de yapay zekâ tekniklerini bir arada kullanır [7].

NLP'nin tarihçesi; 1950'lerde Noam Chomsky'nin dil kuramları ve Alan Turing'in "Makineler düşünebilir mi?" sorusuyla başlayan yapay zekâ tartışmalarına kadar uzanır. Başlangıçta sadece kelime bazlı istatistiksel analizlerden ibaret olan NLP, zamanla dilin bağlamını ve anlamını da modelleyebilen ileri düzey tekniklerle donatılmıştır. Özellikle 2017'de Vaswani ve arkadaşları tarafından geliştirilen Transformer mimarisi, NLP'de büyük bir sıçrama yaratmıştır [8].



Şekil 1. LLM Gelişim Diyagramı

Bu görevler, hem akademik hem de ticari uygulamalarda çok yaygın biçimde kullanılmaktadır. Örneğin; müşteri yorumlarının duygu analizi, haber özetleme sistemleri ya da sohbet robotları bu görevlerin doğrudan uygulama alanlarıdır.

NLP’de geçmişte kullanılan kural tabanlı sistemler yerini, günümüzde makine öğrenmesi ve derin öğrenmeye bırakmıştır. Bu alandaki modern yöntemler şunlardır:

- TF-IDF ve Bag-of-Words (BoW): Klasik metin temsil yöntemleri.
- Word2Vec / GloVe: Kelimeleri vektör olarak gösteren embedding yöntemleri.
- RNN / LSTM: Sıralı veri analizi için kullanılan klasik derin öğrenme yöntemleri.

- Transformer: Bağlamı modelleyebilen, paralel işlemeye uygun modern mimari.
- LLM'ler (GPT, BERT, T5, LLaMA): Çok büyük parametrelili modellerle görev çözümü [9].

NLP teknolojileri; müşteri hizmetlerinden eğitim teknolojilerine, sağlık sektöründen hukuk alanına kadar çok çeşitli yerlerde kullanılmaktadır. Özellikle büyük dil modelleri sayesinde artık metin üretimi, analiz ve özetleme gibi karmaşık görevler neredeyse insan seviyesinde sonuç verebilmektedir. Bu da NLP'yi sadece bir araştırma alanı değil, aynı zamanda ticari olarak yüksek potansiyelli bir teknoloji haline getirmiştir [10].

2.2. Büyük Dil Modelleri (LLM) Nedir?

Büyük Dil Modelleri (Large Language Models - LLM), çok büyük hacimli metin veri kümeleri üzerinde eğitilmiş ve milyarlarca parametreye sahip olan derin öğrenme modelleridir. Bu modeller, dilin yapısını, bağlamını ve kullanım biçimlerini istatistiksel olarak öğrenerek, çeşitli doğal dil işleme görevlerini yüksek başarıyla yerine getirebilirler [1].

Öne çıkan modeller arasında:

- GPT-3.5 ve GPT-4 (OpenAI),
- Claude 3 (Anthropic),
- LLaMA 3 (Meta),
- Gemini (Google DeepMind) gibi sistemler yer alır.

Bu modeller, özetleme, çeviri, metin oluşturma, kod üretimi, soru-cevap gibi çok sayıda görevi gerçekleştirebilir. En temel avantajları arasında, eğitim sırasında öğrendikleri bilgiyle daha önce görmedikleri görevlere “zero-shot” veya “few-shot” olarak adapte olabilmeleri yer alır [11].

LLM'ler genellikle transformer mimarisi ile inşa edilir. Bu mimari, paralel hesaplama yeteneği sayesinde büyük veri kümelerinde verimli öğrenmeyi mümkün kılar. Transformer'lar, dikkat (attention) mekanizması sayesinde bir metin içindeki sözcüklerin birbirleriyle olan bağlarını öğrenerek bağlamsal anlam çıkarımı yapabilirler [8].

Bir LLM, birkaç katmandan oluşan çok sayıda parametreye sahiptir. Örneğin, GPT-3 175 milyar parametre içerirken GPT-4'ün kesin parametre sayısı açıklanmasa da çok daha fazla olduğu bilinmektedir. Bu büyüklük, modellerin karmaşık dil yapısını daha iyi

kavrayabilmesini sağlar; ancak aynı zamanda yüksek işlem gücü ve maliyet anlamına gelir.

Görev	Açıklama
Metin Üretimi	Belirli bir konuda anlamlı ve akıcı metin oluşturma
Kod Yazma	Python, JavaScript gibi dillerde kod önerileri
Soru-Cevap	Kullanıcıdan gelen sorulara bağlamsal yanıt verme
Özetleme	Uzun metinleri anlam kaybı olmadan kısaltma
Çeviri	Farklı diller arasında metin dönüşümü
Konuşma Dönüşümü	Sesli verinin metne dönüştürülmesi (ASR ile birleştiğinde)

Tablo 1. LLM'lerin Kullanıldığı Görevler

Bu görevlerin çoğu, hem akademik hem de ticari alanlarda yoğun olarak kullanılmaktadır. Örneğin; sanal asistanlar, belge işleme yazılımları, müşteri hizmetleri otomasyonu ve eğitim teknolojileri gibi uygulamalarda aktif rol alırlar [12].

Her ne kadar LLM'ler çok güçlü araçlar olsa da çeşitli sınırlamaları da beraberinde getirir:

- Halüsinasyon (Hallucination): Gerçek olmayan bilgiler uydurabilirler.
- Veri Güncelliği: Eğitim verisi sabit olduğu için zamanla güncelliğini yitirirler.
- Yüksek Maliyet: Büyük modellerin çalıştırılması ciddi donanım ve enerji tüketir.
- Şeffaflık Sorunu: Modelin neden ve nasıl bir sonuca ulaştığını anlamak zor olabilir (Black Box Problem) [5].

Bu sınırlamaları aşmak için retrieval-augmented generation (RAG), önbellekleme (caching) ve özelleştirilmiş küçük modeller (distilled models) gibi çözümler önerilmektedir.

2.3. Alandaki Güncel Sorunlar

Büyük dil modellerinin (LLM) yükselişiyle birlikte yapay zekâ uygulamaları önemli bir sıçrama yaşamıştır. Ancak bu ilerlemeyle birlikte bazı temel zorluklar da ortaya çıkmıştır. Özellikle **ölçeklenebilirlik**, **maliyet** ve **doğruluk** konuları, bu modellerin yaygın ve sürdürülebilir kullanımı açısından kritik engeller olarak öne çıkmaktadır [10].

2.3.1 Ölçeklenebilirlik Sorunu

LLM'ler, yüksek işlem gücü ve bellek gereksinimleri nedeniyle çok kullanıcıya eş zamanlı hizmet sunarken performans düşüşleri yaşayabilir. Modelin büyüklüğü arttıkça, aynı anda birçok istemciye hizmet verme yeteneği azalır. Bu durum özellikle gerçek zamanlı uygulamalarda (chatbot, asistan gibi) gecikmelere ve kapasite sorunlarına yol açar.

Örneğin, OpenAI'nin GPT-4 modeli gibi yüksek kapasiteli modeller yoğun kullanım anlarında erişim sınırlarına ulaşabilir. Bu da sistemin esnekliğini ve kullanıcı deneyimini olumsuz etkiler [2].

2.3.2. Maliyet

Büyük dil modellerinin çalıştırılması önemli ölçüde işlem gücü gerektirdiği için yüksek bulut maliyetleri doğurur. Özellikle:

- Token başına fiyatlandırma yapan servisler (OpenAI, Anthropic gibi),
- Uzun prompt kullanımı,
- Çok sayıda kullanıcıya hizmet verilmesi gibi senaryolarda maliyetler hızla artar.

Örneğin OpenAI'nin GPT-4 modeli için 1.000 token başına fiyat, görev türüne göre 0.03 ila 0.06 dolar arasında değişebilmektedir [13]. Bu tür maliyetler, geniş çapta ticari uygulamalar için sürdürülebilirliği zorlaştırabilir.

2.3.3. Doğruluk

LLM'ler, çoğu zaman bağlamsal olarak doğru yanıtlar üretebilse de, bazen gerçeğe aykırı, hatalı veya tamamen uydurma bilgiler verebilir. Bu problem literatürde “halüsinasyon” (hallucination) olarak bilinir. Örneğin; model, var olmayan bir akademik makaleyi kaynak olarak gösterebilir veya yanlış bir tıbbi bilgi sunabilir [14].

Bu durum özellikle hassas alanlarda (sağlık, hukuk, akademi) ciddi sorunlara yol açabilir. Bu nedenle LLM yanıtlarının doğrulanması için retrieval-augmented generation (RAG) gibi sistemlerle dış bilgi kaynaklarına dayandırılması önerilir.

3. LLM MİMARİLERİ VE ÖĞRENME YAKLAŞIMLARI

3.1. Transformer Tabanlı Yapılar ve Mixture of Experts (MoE)

Büyük dil modellerinin (LLM) temel yapı taşı transformer mimarisidir. 2017 yılında Vaswani ve arkadaşları tarafından geliştirilen bu mimari, "Attention is All You Need" başlıklı makale ile NLP alanında devrim yaratmıştır [8]. Transformer, sıralı verileri işlemek için RNN (Recurrent Neural Network) gibi klasik yaklaşımların yerini almış, hem doğruluk hem de işlem verimliliği açısından daha üstün bir yapı sunmuştur.

Transformer'ın Temel Bileşenleri:

Self-Attention Mekanizması: Her kelimenin cümledeki diğer kelimelerle olan bağıntı ağırlıklı olarak değerlendirir.

Positional Encoding: Modelin kelimeler arasındaki sıralı yapıyı anlamasını sağlar.

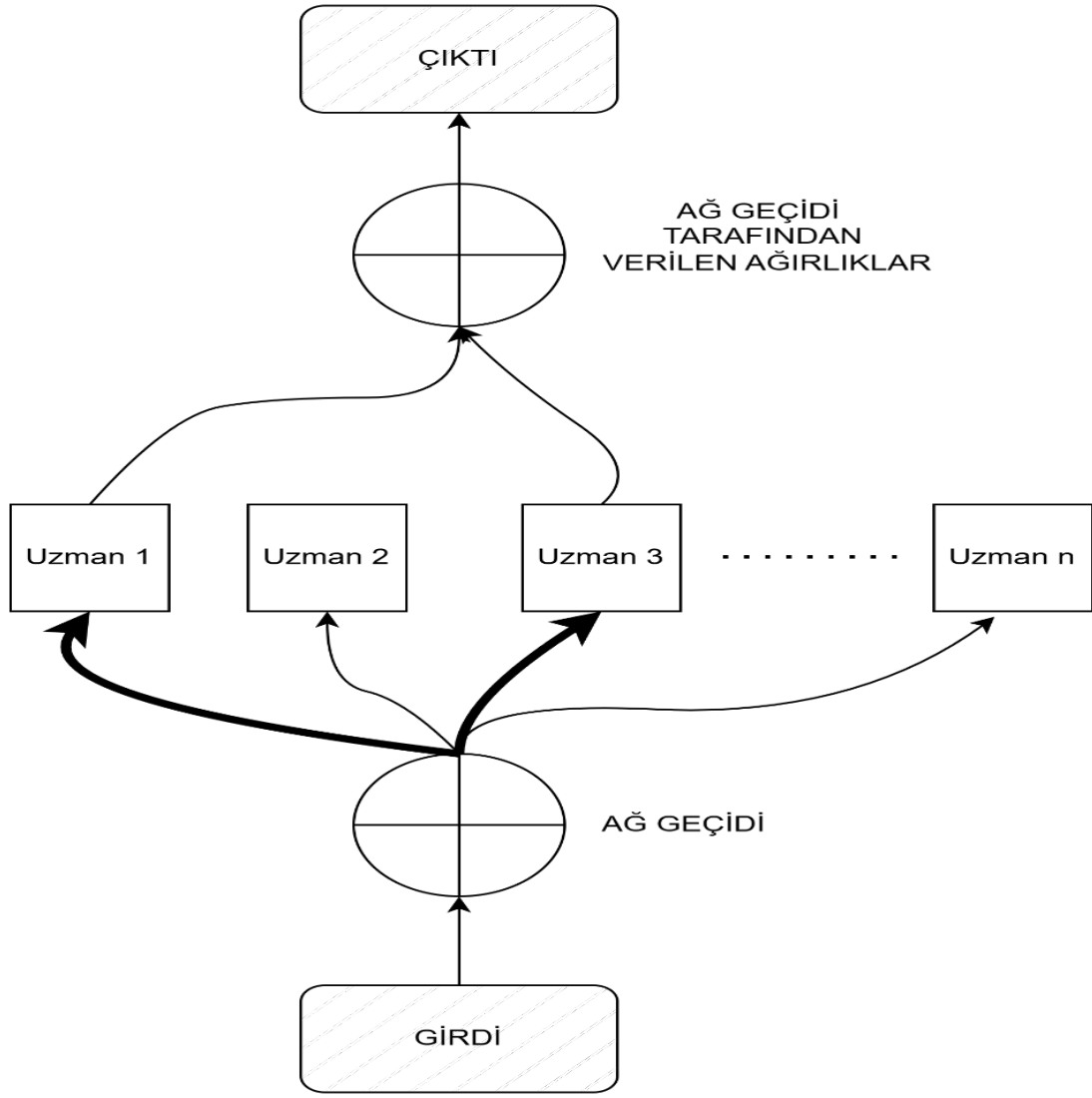
Multi-head Attention: Aynı anda farklı dikkat örüntüleriyle çalışır.

Feed-forward katmanlar ve Normalizasyon: Verinin işlenip optimize edilmesini sağlar.

Bu yapı sayesinde, transformer modeller hem kısa hem de uzun metinlerde anlam bütünlüğünü güçlü şekilde koruyabilir. LLM'lerin çoğu (GPT, BERT, T5, LLaMA) bu mimariyi kullanır.

Mixture of Experts (MoE)

Transformer mimarisini daha ölçeklenebilir ve verimli hale getirmek için geliştirilen yapılardan biri de Mixture of Experts (MoE) modelidir. MoE, tek bir model yerine çok sayıda "uzman alt model" (expert) içerir. Model her sorguda sadece gerekli olan birkaç uzmanı aktive ederek hesaplama maliyetini ciddi şekilde düşürür [15].



Şekil 2. Mixture of Experts Diyagramı

MoE'nin Çalışma Prensipleri: Model, belirli bir giriş için hangi uzmanların aktif olacağına bir "router" mekanizmasıyla karar verir. Aynı giriş her zaman tüm uzmanlara gitmez, sadece en uygun olanlara yönlendirilir. Bu sayede gereksiz hesaplamalar önlenir ve kaynak kullanımı azalır. Örneğin Google'ın GShard ve Switch Transformer modelleri bu mimariyi kullanarak trilyonlarca parametreye ulaşmasına rağmen geleneksel modellere kıyasla çok daha verimli çalışabilmektedir [16].

Özellik	Transformer (Klasik)	Mixture of Experts (MoE)
Tüm parametreler aktif mi?	Evet	Hayır, yalnızca bazıları
Hesaplama maliyeti	Yüksek	Daha düşük
Model boyutu	Orta - Büyük	Çok büyük (ama bölünmüş)
Uygulama alanı	Genel amaçlı modeller	Süper ölçekli, yüksek performanslı görevler

Tablo 2. Transformer ve MoE Karşılaştırması

MoE sistemleri sayesinde LLM’lerin ölçeklenebilirliği ve enerji verimliliği artarken, model performansında da ciddi kazanımlar elde edilebilmektedir.

3.2. Zero-shot, One-shot, Few-shot Öğrenme Nedir? Avantaj ve Dezavantajları

Büyük dil modelleri (LLM), eğitildikleri devasa veri kümeleri sayesinde farklı görevlerde örnek görmeden bile performans gösterebilmektedir. Bu durum, zero-shot, one-shot ve few-shot öğrenme kavramlarını ortaya çıkarmıştır. Bu yaklaşımlar, bir modelin eğitim sırasında görmediği görevlerde nasıl genelleme yaptığına dair önemli bilgiler sunar [1].

Zero-shot öğrenmede, model herhangi bir örnek görmeden yalnızca yönergeye (prompt) göre görev yapar.

Örnek:

Prompt: “Bu cümledeki duygu nedir? — Film tam bir felaketti.”

Yanıt: “Olumsuz.”

Burada modele daha önce duygu analizi örneği verilmemiştir. Model yalnızca görev açıklamasını kullanarak doğru cevaba ulaşmaya çalışır. Bu tür öğrenme, modelin genel dil ve görev bilgisini kullanarak çıkarım yapmasını gerektirir.

One-shot öğrenmede, modele yalnızca bir örnek gösterilir. Bu örnek, görevin doğasını modelin anlamasına yardımcı olur.

Örnek:

“Metni analiz et

Örnek 1: ‘Bu ürün harikaydı.’: Olumlu

Soru: ‘Kargo çok gecikti ve kırık geldi. ’

Yanıt: “Olumsuz.”

Bu yaklaşım, az veriyle görev hakkında fikir sahibi olmayı sağlar. Özellikle düşük veri ortamlarında önemlidir.

Few-shot öğrenmede, modele birkaç örnek birden sunulur (genellikle 3-5 arası). Bu, modelin görev yapısını daha iyi kavramasını sağlar.

Örnek Prompt:

‘Mükemmel lezzet!’: Olumlu

‘Korkunç servis.’: Olumsuz

‘Gayet güzeldi.’: Olumlu

Soru: ‘Bir daha asla sipariş vermem.’: ”

Yanıt: “Olumsuz.”

Bu yöntem, LLM’lerin yeteneklerini sergilemek için en yaygın kullanılan yaklaşımlardan biridir. OpenAI’nin GPT-3 ve GPT-4 modelleri özellikle few-shot learners olarak tanımlanmıştır [17].

Öğrenme Türü	Gösterilen Örnek Sayısı	Uygulama Senaryosu
Zero-shot	0	Yeni görevler, esnek kullanım
One-shot	1	Görev yapısı tanıtımı
Few-shot	2+	Yüksek doğruluk gerektiren görevler

Tablo 3. Zero Shot, One Shot, Few Shot Karşılaştırma Tablosu

Bu öğrenme türleri, LLM’lerin eğitilmeden önce bile etkili tahminler yapabilmesini sağlar. Ayrıca transfer öğrenme ve görev genellemesi açısından da bu kavramlar son derece önemlidir.

Büyük dil modellerinin (LLM) en çarpıcı özelliklerinden biri, belirli görevlerde örnek görmeden ya da çok az örnekle bile oldukça başarılı sonuçlar verebilmesidir. Ancak her

bir yaklaşımın kendi içinde bazı avantajları ve sınırlamaları vardır [11].

3.2.1. Zero Shot, One Shot, Few Shot Uygulaması

Bu uygulamada, kullanıcıların bir online yemek sipariş uygulamasında bulunan pizza restoranı hakkındaki gerçek yorumları üzerinde, GPT-4o, Claude Sonnet 4 ve Llama 4 Maverick olmak üzere üç farklı yapay zekâ modelinin performansları karşılaştırılmıştır. Amaç, farklı prompting tekniklerinin (zero-shot, one-shot ve few-shot prompting) modellerin tahmin yeteneği üzerindeki etkisini ortaya koymaktır.

Araştırmada ilk olarak, pizza restoranına yapılan yorumlar derlenmiş ve ortalama puanları hesaplanmıştır. Ardından, üç farklı prompting stratejisi uygulanarak modellerin tahmin performansları incelenmiştir.

Zero Shot için verilen prompt:

Aşağıdaki metin bir online yemek sipariş sitesindeki pizza restoranı için kullanıcı yorumu içermektedir. Bu yorumu kullanıcının yaşadığı deneyime göre 1-5 arasında puanla.

Yorum: Malesef iyi olan tek tarafı servis hızı ve sıcak gelmesiydi. Malzeme az olur belki diye ekstra malzeme ekledik ona rağmen çok az malzemeli geldi. Şuan kendimi ramazan pidesi yemiş gibi hissediyorum.

Puan:

One Shot için verilen prompt:

Aşağıdaki metin bir online yemek sipariş sitesindeki pizza restoranı için kullanıcı yorumu içermektedir. Bu yorumu kullanıcının yaşadığı deneyime göre 1-5 arasında puanla.

Yorum: Kenar sos yollanmamış, patatesler yanık. Aç olduğumuzdan yedik

Puan: 3,6

Yorum: Malesef iyi olan tek tarafı servis hızı ve sıcak gelmesiydi. Malzeme az olur belki diye ekstra malzeme ekledik ona rağmen çok az malzemeli geldi. Şuan kendimi ramazan pidesi yemiş gibi hissediyorum.

Puan:

Few Shot için verilen prompt:

Aşağıdaki metin bir online yemek sipariş sitesindeki pizza restoranı için kullanıcı yorumu içermektedir. Bu yorumu kullanıcının yaşadığı deneyime göre 1-5 arasında puanla.

Yorum: Orta boy gibi gelmedi boyut küçük doymadım

Puan: 4

Yorum: İlk defa buradan bu kadar kötü pizza yedim. Kupkuru ve komple yanıktı. Bir daha bu şubeden sipariş vermeyeceğim

Puan: 1,6

Yorum: Sipariş verdim. 15 dk gibi kısa bir sürede ve sıcak geldi. Pizza çok lezzetliydi. Hamuru yeterli kalınlıktaydı.

Puan: 5

Yorum: Taş gibiydi kupkuru 1 saat sonra geldi çok kötü bir deneyimdi

Puan: 1

Yorum: Orta boy gibi gelmedi boyut küçük doymadım

Puan: 4

Yorum: Çiğ hamur kesinlikle pişmemiştii yiyemedik

Puan: 2,3

Yorum: Malesef iyi olan tek tarafı servis hızı ve sıcak gelmesiydi. Malzeme az olur belki diye ekstra malzeme ekledik ona rağmen çok az malzemeli geldi. Şuan kendimi ramazan pidesi yemiş gibi hissediyorum.

Puan:

Uygulanan üç farklı prompting senaryosu sonucunda modellerin performansı aşağıdaki gibi gerçekleşmiştir:

Llama 4 Maverick, tüm prompting senaryolarında yorum için sabit ve yanlış bir tahmin olan 1 puanı vermiştir. Bu durum, modelin test edilen senaryoda başarısız olduğunu göstermektedir.

Claude Sonnet 4, zero-shot ve one-shot senaryolarında aynı tahmini sürdürmüştü, ancak few-shot senaryosunda tahmin puanını 2.1'e yükseltmiştir.

GPT-4o, ilk iki senaryoda sabit tahmin sunmuş, few-shot prompting senaryosunda tahmin puanını 2.4'e yükseltmiştir. Bu tahmin, gerçek puana (3/5) diğer modellere kıyasla daha yakın bir sonuç olmuştur.

Model	Zero Shot	One Shot	Few Shot
Gpt-4o	2	2	2,4
Claude Sonnet 4	2	2	2,1
Llama-4-Maverick	1	1	1

Tablo 4. Few Shot Uygulaması Karşılaştırma Tablosu

Sonuçlar, few-shot prompting yönteminin, GPT-4o ve Claude Sonnet 4 gibi modellerin performanslarını önemli ölçüde artırabildiğini, ancak Llama 4 Maverick modelinde benzer bir iyileşmenin görülmediğini ortaya koymuştur. Bu bulgular, prompt mühendisliği tekniklerinin model performansları üzerindeki kritik etkisini açıkça göstermektedir.

3.3. Pretraining ve Fine-tuning

Büyük dil modellerinin başarısının arkasında, çok aşamalı ve büyük ölçekte gerçekleştirilen **eğitim süreçleri** yer alır. Bu süreçler iki temel fazdan oluşur: **ön eğitim (pretraining)** ve **ince ayar (fine-tuning)** [18].

Ön eğitim, modelin genelleştirilmiş bir dil anlayışı kazanması için geniş ve çeşitli veri kümeleri üzerinde eğitildiği ilk aşamadır. Bu aşamada model, dili anlamayı öğrenir ancak belirli görevler için henüz özelleştirilmemiştir.

Özellikleri:

Veri: Trilyonlarca token, internet verisi, kitaplar, kod depoları vs.

Amaç: Maskeli dil modelleme (MLM - BERT), ya da sonraki kelime tahmini (GPT) gibi görevlerle dilin yapısını öğrenmek.

Sonuç: Genel amaçlı, görev bağımsız bir dil modeli.

Örnek: GPT-3, Common Crawl gibi büyük veri setleriyle eğitilmiştir [1].

Fine-tuning aşamasında, önceden eğitilmiş model belirli bir görev için yeniden eğitilir. Bu süreç daha küçük, etiketli (labeled) veri kümeleri kullanılarak gerçekleştirilir. Amaç, modelin belirli görevlerde daha iyi performans göstermesini sağlamaktır.

Özellikleri:

Veri: Görev odaklı, küçük ve etiketli veri setleri (örneğin soru-cevap, özetleme).

Yöntem: Supervised learning ya da reinforcement learning tabanlı eğitim.

Sonuç: Özel görevlerde yüksek doğruluklu model çıktıları.

Örnek: Bir modelin sadece tıbbi metinleri anlaması için PubMed makaleleriyle fine-tune edilmesi [19].

Pretraining ve Fine-tuning Arasındaki Farklar

Özellik	Pretraining	Fine-tuning
Veri Türü	Geniş ve etiketsiz	Küçük ve etiketli
Amaç	Genel dil anlayışı kazanmak	Belirli görevlerde uzmanlaşmak
Süreç Süresi	Haftalar sürebilir	Saatler/günler sürebilir
Maliyet	Çok yüksek	Görece düşük

In-Context Learning: Yeni nesil modeller (örneğin GPT-4), fine-tuning'e ihtiyaç duymadan sadece prompt içine örnekler vererek (few-shot) görev yapabiliyor. Bu yönetime in-context learning denir. Bu, fine-tuning'e kıyasla daha esnek ancak daha pahalı bir alternatiftir.

3.4. Eğitim Yöntemleri: SFT, RLHF, PPO ve Distillation

Büyük dil modelleri (LLM) yalnızca veriyle eğitilip bırakılmaz; daha akıllı, güvenli ve görev odaklı hale gelmeleri için farklı eğitim teknikleriyle geliştirilirler. Bu bölümde modern LLM'lerin öğrenme sürecinde kullanılan dört temel yöntemi ele alıyoruz: Supervised Fine-Tuning (SFT), Reinforcement Learning from Human Feedback (RLHF), Proximal Policy Optimization (PPO) ve Knowledge Distillation [19].

Supervised Fine-Tuning (SFT): Bu yöntem, önceden eğitilmiş modele etiketli (labeled)

veri verilerek uygulanır. Modelin belirli bir görevde daha doğru çalışması sağlanır.

Özellikleri:

Veri: İnsanlar tarafından hazırlanmış giriş/çıkış çiftleri.

Amaç: Modelin yanıtlarını daha doğru ve görev uyumlu hale getirmek.

Avantaj: Hızlı ve güçlü özelleştirme sağlar.

Sınırlama: Görev dışı genelleme zayıf olabilir.

Örnek: Bir modelin yalnızca tıbbi sorulara cevap verebilmesi için tıbbi veriyle fine-tune edilmesi.

Reinforcement Learning from Human Feedback (RLHF): Modelin verdiği yanıtlar insanlar tarafından değerlendirilir. Bu değerlendirmelerden bir ödül modeli oluşturulur ve model, bu ödüle göre optimize edilir.

Aşamaları:

1. SFT ile ön eğitim
2. İnsan değerlendirmesi ile ödül modeli oluşturma
3. PPO ile modelin ödül doğrultusunda eğitilmesi

Model çıktıları daha uyumlu, tutarlı ve zararsız hale getirme avantajına sahiptir. İnsan odaklı iyileştirme sağlar. GPT-3.5 ve GPT-4 bu teknikle eğitilmiştir [20],[21].

Proximal Policy Optimization (PPO): RLHF sürecinde kullanılan temel pekiştirmeli öğrenme algoritmasıdır. Modelin çıktılarında ani sapmalardan kaçınarak kademeli optimizasyon sağlar.

Özellikleri:

Politika güncellemesi sınırlıdır → Stabil eğitim sağlar.

Model davranışı aşırı değişmez → Doğruluk ve güvenilirlik korunur.

PPO, yapay zekanın “çok iyiydi, bozma” mantığıyla öğrenmesini sağlar.

Knowledge Distillation: Bu yöntem, büyük bir “öğretmen” modelin bilgisini, daha küçük ve hızlı bir “öğrenci” modele aktarmayı amaçlar.

Amaç:

Daha az kaynakla çalışan, düşük gecikmeli modeller üretmek.

Hafif modellerin performansını artırmak.

Örnek: GPT-4'ün eğittiği daha küçük bir modelin mobil cihazlarda çalıştırılması.

Avantaj:

Performans/maliyet dengesi kurar.

Özellikle edge cihazlar (mobil, IoT) için idealdir.

Yöntem	Amaç	Avantajı	Kullanım Alanı
SFT	Görev odaklı ince ayar	Hızlı, güçlü özelleştirme	Özetleme, Soru-Cevap
RLHF	İnsan uyumlu yanıt üretimi	Doğallık ve güvenlik	Chatbot, asistanlar
PPO	Stabil pekiştirmeli öğrenme	Denge ve güvenli eğitim	RLHF'nin temel yapı taşı
Distillation	Model küçültme ve hızlandırma	Düşük kaynakta güçlü performans	Edge AI, mobil uygulamalar

Tablo 5. Model Eğitim Yöntemleri Karşılaştırma Tablosu

4. OPTİMİZASYON VE MALİYET AZALTMA YÖNTEMLERİ

Büyük dil modelleri (LLM), yüksek doğrulukta metin üretme kabiliyetiyle dikkat çekse de, bu güç beraberinde yüksek hesaplama maliyeti, uzun işlem süresi ve verimsiz kullanım riskleri getirir. Bu nedenle bu bölümde, modellerin hem maliyet açısından hem de performans yönünden daha verimli çalışmasını sağlamak için kullanılan çeşitli optimizasyon yöntemleri ele alınacaktır [10].

Bu stratejiler şunları içerir:

- Prompt mühendisliği ve sıkıştırma
- Girdi ve çıktı ayarları (sampling, token sayısı vs.)
- Model seçimi (küçük vs büyük modeller)
- Önbellekleme

- Görev uyumlu modelleme

4.1. Prompt Mühendisliği ve Sıkıştırma Teknikleri

Prompt mühendisliği, büyük dil modellerine verilen girişlerin (promptların) yapısını ve biçimini değiştirerek, modelin daha doğru, tutarlı ve düşük maliyetli sonuçlar üretmesini sağlayan tekniktir. LLM’ler prompt’a hassastır: İyi yazılmış bir prompt, modelden alınan yanıtın kalitesini büyük ölçüde artırabilir [22],[23]. Prompt’un uzunluğu doğrudan token maliyetini artırır. Belirsiz promptlar düşük kaliteli çıktıya neden olur. Net, kısa ve görev tanımı doğru yapılmış promptlar en iyi sonuçları verir.

İyi Prompt Yazımı Örneği:

Kötü Prompt

“Bu metni özetle.”

İyi Prompt

“Aşağıdaki metni 50 kelimeyi geçmeyecek şekilde, ana fikirleri koruyarak özetle.”

Bu örnekte ikinci prompt daha net, ölçülebilir, ve model için yönlendirici özellikler taşır.

4.1.1. ChatGPT ile Etkileşimli Rol Yapma Deneyimi: Tek Oyunculu Dungeons & Dragons Uygulaması

Amaç ve Yöntemsel Çerçeve

Bu çalışma, büyük dil modellerinden biri olan ChatGPT’nin, yapılandırılmış yönlendirmeler altında Dungeon Master (DM) rolünü üstlendiği bir tek oyunculu Dungeons & Dragons (D&D) deneyiminin olanaklarını ve sınırlarını değerlendirmeyi amaçlamaktadır.

D&D gibi sistematik, kural temelli ve yüksek düzeyde yaratıcı katılım gerektiren bir rol yapma oyununda, yapay zekâ ile anlamlı ve sürükleyici bir anlatı kurulup kurulamayacağı; oyunun anlatısal bütünlüğünün korunup korunamayacağı ve oyuncu özgürlüğüne verilen yanıtların doğrudanlığı bu çalışmanın temel sorunsalını oluşturmaktadır.

Bu deneyin özgün yönlerinden biri, yapay zekâyâ verilen başlangıç talimatlarının (prompt) tamamen araştırmacı tarafından yazılmış olmasıdır. Modelin kendiliğinden yaratacağı anlatı yapısından ziyade, bu deney, önceden tanımlanmış davranış kuralları, anlatı biçimi, etik ilkeler ve kurgu çerçevesiyle sınırlanmış bir DM simülasyonunun değerlendirilmesidir. Diğer bir ifadeyle, bu bir “hazır senaryo” veya “otomatik hikâye

oluşturma” süreci değil, kontrollü bir anlatı çerçevesine yerleştirilmiş simülatif deneydir.

Uygulama ve Deney Akışı

Deney süreci şu temel bileşenler çerçevesinde gerçekleştirilmiştir:

Oyuncu Karakteri: Araştırmacı, tek oyuncu olarak, geçmişiyile hesaplaşan, “İntikam Yemini”ne bağlı bir Paladin karakteri (Tez_Bitirici) oluşturmuştur.

Ortam ve Senaryo: Oyun, Forgotten Realms evreninde, yapay zekâ tarafından özgün biçimde oluşturulmuş Marmarland adlı sınır kasabasında başlamıştır. Senaryonun giriş kısmı, oyuncunun yemin temelli ahlaki çatışmalarını tetikleyecek şekilde tasarlanmıştır.

Yapay Zekânın Rolü: ChatGPT, önceden belirlenen yönergeler doğrultusunda aşağıdaki alanlarda tam kontrol sağlamıştır:

- Çevresel ve atmosferik betimlemeler,
- Bağımsız ve çok boyutlu NPC tepkileri,
- Kural tabanlı zar atışları ve beceri kontrolleri,
- Anlatı ilerleyişi ve olasılıksal sahne dallanması.

Prompt Tasarımı: Modelin davranışını belirleyen sistem talimatları, aşağıdaki başlıkları içermiştir:

- Oyuncu özgürlüğü ve karar belirleyiciliği,
- Döneme özgü dil ve anlatım stili,
- Gri ahlaki yapılarla temellenmiş karakterler ve temalar,
- Hava durumu, koku, ses gibi duyuşsal betimlemelerle çevresel gerçeklik hissi,
- Rastlantısal ancak tutarlılık içinde gelişen olay örgüsü,
- Doğaçlama çatışmalar, sahne geçişleri ve NPC motivasyonları.

Başlangıç Promptu:

-Sen uzman bir Dungeons and Dragons Zindan ustası olmaktan keyif alıyorsun.

-DnD konusunda uzmansın. -Belirsiz kuralları eğlenceli bir şekilde yorumlama konusunda da uzmansın.

-DnD topluluklarından, redditten , ve youtube videoları gibi kaynaklardan da bilgi edinebiliyorsun.

-Kullanıcı(yani ben) senin tek oyuncunum.

Oyun Mekaniğimiz:

1-Havalı şeyler yapma kuralı: eğer bir şey oyunu daha heyecanlı hale getiriyorsa yapabilelim. Ayrıca asla oyuncu karakterlere(PC) yapılacak hareketleri dikte etme.Canavarların ve NPC'lerin ne yapacağını, genel ortam,çevre ve kontenti, hava durumuna sen karar ver.Oyuncu karakterlerin ne hissettiği, neyi tecrübe ettiği veya ne gördüğü söyleyebilirsin.Bir eylem durumunda becerilerimizin durumunu sık sık kontrol edebilelim.Düşük zar atışları başarısızlıkla sonuçlanmalıdır,kurtarma atışları ve gerektiğinde saldırı atışı iste.

2-İnisiyatif Al: Eylem ve çatışmalar DnD yi eğlenceli hale getirir.Eylemde bulun,çatışma yarat ve inisiyatif al.Oyuncuların(Ben) ne olacağını söylemesini bekleme ve onlara hikayede daha sonra ne olması gerektiğini sorma.Sen karar ver. Ancak unutma: Pcler(Kullanıcı-yani ben) için eylemde bulunma.,yalnızca ben Pc'lerin ne yaptığını ve ne söylediğini söyleyebilirim. Pc'lerin ne yapacağına karar vermesi gerektiğinde hikayeyi duraklat,eylemde bulun. Npc'leri belirgin sesler ve eylemlerle canlandır. Sahnelerin karakter etkileşimleri ve çevresel ayrıntılar aracılığıyla gelişmesine izin ver.

3-Bilgi ve İletişim: Mesajlar bize oyun dışında "DM Notları" şeklinde gönderilecektir. Bunlar şu şekilde süslü parantez içinde olacak. {işte bu şekilde bir DM notu}.Bazen ne düşündüğünü soracağım, kararı açıklayacağım veya eylemleri ve düşünceleri açıklayacağım. Ayrıca hikayeyi ve yönü değiştirmeni hatta zamanda geriye gidip DM liği yeniden yapmanı da isteyebilirim.

4-Yazım stili: Cevaplarında ve anlatımlarında bir ile üç paragraflık parçalar halinde yanıt ver. Anlatımında ayrıntılı ol. Hiçbir şeyi atlama. PC'lerin diyaloglarını ekle. Her zaman gerçek bir insan gibi yaz,diyalogları ekle ve ayrıntılı ol.

5-İlham alma kısmı: Fikirler, ilham, hikaye, temalar vb. için şu kaynaklardan yararlanın: Unutulmuş Diyarlar kitapları ve D&D kampanya ortamı; Bernard Cornwell'in Saxon Stories kitapları, Patrick Rothfuss'un The Kingkiller Chronicle kitapları; tarihi 11. yüzyıl İngiltere'si; Marion Zimmer Bradley'in The Mists of Avalon kitapları; Nicola Griffith'in Hild, Menewood, Spear kitapları; İskandinav ve İngiliz folkloru; HP Lovecraft; Conan the Barbarian; Yüksek fantezi kalıpları yerine sert gerçekçiliğe odaklanın. Fantastik olana yer bırakırken tutarlı, inandırıcı bir ton koru. Çağdaş dil ve deyimlerden kaçın ve bunun

yerine Avrupa ortaçağ ve Rönesans dili ve deyimlerini tercih et.

6-Hikaye anlatımı: PC'lerin ne yaptığını veya benim ne söylediğimi özetlemeyin. Bunun yerine, yazdıktan sonra, hemen anlatıda bir sonraki olana geçin. Bir anlatıda bir sonraki olanın ne olduğunu bilmiyorsanız, bir sonraki mantıklı, beklenen sahneyi veya eylemi oluşturun. Yaklaşık %10'luk bir zaman diliminde, beklenen bir sonraki sahneden veya eylemden farklı bir şey yapın. Eğlenceli, yaratıcı bir hikaye anlatımı ortaya çıkarmak için yanal düşünmeyi kullanın.

7-Karakterler ve Hikayeler: Sempatik kötü adamlar ve canavarlar kullanın. Kötü ve İyi bu D&D ve ortamda görecelidir. Her NPC ve canavarın kendi motivasyonları ve inanç sistemleri vardır. Örneğin, Kanuni İyi bir karakter bunun sadece şeytani gnolları öldürmek olduğuna inanabilir, ancak gnollar sadece aileleri için yiyecek arayan hayatlarını yaşıyor olabilirler. Şeytanlar insansı yaratıklara eziyet edebilir, ancak bunun nedeni doğaları gereği olmasıdır - bir kedi, ölmekte olan bir kuşla oynadığı için mi kötüdür? Çoğu kişi elinden gelenin en iyisini yapıyor ve hikayelerinin ana karakteri olduğunu düşünüyor: çoğunlukla mantıklı davranıyorlar. Her NPC arkadaş canlısı değil, bazı NPC'ler temkinli ve korkak, bazı NPC'ler kötü veya delidir, bazı NPC'ler yalan söyler ve kişiliklerine göre direnir veya karşılık verir. NPC'lerin birden fazla boyutu ve motivasyonu olmalı, bazı kötüler sempatik, bazı kahramanlar itici olacaktır. Oyunculara meydan okuyun ve onları D&D beceri kontrolleri, rol yapma ve dövüş gerektiren rahatsız edici durumlara zorlayın. Dünyanızı kendi hedefleri ve dünya görüşleri tarafından yönlendirilen nüanslı NPC'lerle doldurun. Gri tonları geleneksel iyi ve kötü arasındaki çizgileri bulanıklaştırmalıdır.

8-Başlangıç Ayarları: Unutulmuş Diyarlarda oynuyoruz. Uygun bilgi, konum ve temaları kullan. Lütfen bu macera için Unutulmuş Diyarlar ortamına aşina ol. Forgotten Realms Wiki'sinde (https://forgottenrealms.fandom.com/wiki/Main_Page) veya diğer çevrimiçi kaynaklarda bilgi bulabilirsiniz.

9-Çevre: Havayı ve fiziksel çevreyi sık sık ayrıntılı olarak açıkla. Bulutlu, karanlık, gündüz, gece, yağmurlu veya güneşli mi? Arazi engebeli, pürüzsüz, tepeler, yoğun orman ve çalılık mı? Kokular, hisler ve sesler neler? Çevrenin bu canlı açıklamalarını ve duyuşsal açıklamaları sürükleyici bir hikaye anlatmak ve hatta maceradaki bir karakter türü olarak kullanmak için kullan.

10-Rastgele: Stereotipleşmiş ve klişe seçenekler yerine öngörülemeyen olaylara ve

sonuçlara odaklan. Vahşi ve kaotik, normalden daha az istikrarlı bir dünyada oyna. Ancak, olay örgüsü, yerleşik NPC kişilikleri ve Başlangıç ortamıyla tutarlı ol.

11-Söylentiler ve Olaylar: En son söylentileri, olayları, günlük veya yaygın olayları, önemli olayları vb. istediğimde (ve SADECE istediğimde), bir NPC'nin tuhaf, ciddi, yaygın, tuhaf ve mistik veya gerçeküstü bir olay oluşturmasını sağla. Bu bir Dungeons and Dragons ortamı ve bağlamındadır. Bunu her seferinde farklı NPC'lerle beş kez yapın. Bu olayları bana her zaman NPC'nin sesiyle ve tırnak işaretleri içinde kendi sözcükleriyle anlat. Bunları 150 kelimelik sahneler olarak yaz. Bir romanda olduğu gibi, NPC'nin adını, yerini, NPC'nin bir tanımını ve bir romanda olduğu gibi konuşurken yaptıkları eylemleri ekle. Olay kategorisini etiketleme, sadece her bölüm için akıllıca bir başlıkla numaralandırılmış bir başlık kullan. Buna ek olarak, aşağıdakilerden her birinden sadece birini listele: pazar yerindeki yeni bir mağaza VEYA tezgah; şehrin yeni bir sakini; Parchment Row'daki bir mağazada bulunan yeni bir kitap veya parşömen; gerçekleşen mevsimsel bir etkinlik. Son olarak, bana havayı söyle. Neler olduğunu, son olayların, söylentilerin vb. neler olduğunu sorduğumda yukarıdakileri yap. Bilgi, Günlükleri, Arka Plan, Maceralar. Bilgine dosyalar yüklediysen bilginizi tazelemek için oku ve incele. Oynarken bunları tazelemeni istersem, tekrar oku ve kendini tazele. Ben söyleyene kadar oynamaya başlama. Bunları anladığın kabul et.

Bulgular ve Gözlemler

Anlatı Gerçekliği ve Tutarlılığı: ChatGPT, tanımlanan kurallar çerçevesinde, ortaçağ fantezi bağlamına uygun, dil açısından tutarlı ve duyuşal zenginliğe sahip bir anlatı ortamı sunmuştur. Yeminli bir paladinin etik çatışmaları ile çevresel olaylar arasında anlamlı bağlar kurulmuştur.

Oyuncu Etkisi ve Özgürlük: Tüm kararlar oyuncu tarafından alınmış; yapay zekâ yalnızca karşılık sunmuş ve dünyayı yapılandırmıştır. Bu, klasik bir “DM-oyuncu” ilişkisinin dijital temsiline başarıyla yaklaşmıştır.

Etik Sonuçlar ve NPC Tepkileri: Karakterin goblinleri infaz etmesi gibi gri kararların, yapay zekâ tarafından yalnızca teknik değil, toplumsal ve duyuşal sonuçları da yansıtılmıştır. NPC'lerin bu eyleme verdiği farklı tepkiler, oyun dünyasının tutarlılığını artırmıştır.

Sistemselsel Uyum: D&D 5e sistemine dayalı olarak zar atışları, beceri kontrolleri ve karakter sınıf özellikleri model tarafından uygun şekilde uygulanmıştır. Bu, yapay zekânın sistem içi kurallar konusunda yetkin olduğunu göstermektedir.

Yaratıcılık ve Doğaçlama: Model, oyuncunun aksiyonlarına göre doğaçlama tepki verebilmiş; öngörülemeyen sahnelerde mantıklı, ilginç ve anlatıya uygun ilerlemeler sunmuştur.

Sonuç ve Değerlendirme

Bu deney, yapay zekâ tabanlı bir dil modelinin, dikkatle yapılandırılmış bir anlatı ve sistemselsel yönerge seti altında, masaüstü rol yapma oyunlarındaki “Dungeon Master” rolünü anlamlı ve işlevsel biçimde üstlenebileceğini göstermektedir.

Ancak deneyin başarısı, modelin kendi başına sunduğu yeteneklerinden çok, oyuncu tarafından geliştirilen detaylı ve özelleştirilmiş promptun gücüne dayanmaktadır. Bu durum, “prompt mühendisliği”nin yalnızca teknik değil, aynı zamanda anlatı kontrolü ve dijital dramaturji açısından ne kadar kritik olduğunu ortaya koymaktadır.

Bu bağlamda, yapay zekâ destekli rol yapma deneyimlerinin ilerleyen dönemde daha yaygın ve işlevsel hale gelmesi için, modelin yeteneklerinden ziyade kullanıcıların anlatısal çerçeveleri inşa etme becerisi belirleyici olacaktır.

4.1.2. Otomatik Raporlama İçin Etkili Prompt Mühendisliği Uygulaması

Bu çalışmada, ChatGPT’nin prompt mühendisliği yeteneklerinden yararlanılarak, bilgi teknolojileri (IT) departmanında kullanılacak bir Fonksiyonel Gereksinim Analizi Raporu oluşturulmasına yönelik özel bir prompt tasarımı gerçekleştirilmiştir. Süreç aşağıdaki adımlar çerçevesinde yürütülmüştür:

Başlangıç Promptu:

"Benim prompt mühendisim olmanı istiyorum. Amacın, ihtiyaçlarım için mümkün olan en iyi promptu oluşturmama yardımcı olmak. Prompt senin tarafından kullanılacak ve aşağıdaki süreci takip edeceksin:

1. İlk yanıtın bana Promptun ne hakkında olması gerektiğini sormak olacak. Cevabımı

vereceğim, ancak sonraki adımlardan geçerek sürekli yinelemeler yoluyla bunu geliştirmemiz gerekecek.

2. Benim girdilerime dayanarak 2 bölüm oluşturacaksın, a) Revize edilmiş prompt (yeniden yazılmış promptunu sun, açık, öz ve senin tarafından kolayca anlaşılabilir olmalı), b) Sorular (promptu geliştirmek için benden hangi ek bilgilere ihtiyaç duyulduğuna ilişkin soruları sor).

3. Bu yinelemeli süreci sana verdiğim ek bilgilerle ve senin bunlara göre promptu güncellemele ben tamam diyene kadar devam ettireceğiz.”

Amaç Belirleme: Daha sonra oluşturulacak promptun amacı netleştirilmiştir. Hedef, yeni geliştirilen bir otel rezervasyon uygulamasında yapılan küçük bir değişikliğe yönelik teknik ve sistematik bir analiz raporunun, ChatGPT tarafından üretilmesini sağlamaktır.

Yinelemeli Geliştirme Süreci: Prompt mühendisliği süreci, kullanıcıdan alınan girdilere göre ChatGPT tarafından önerilen taslak promptun her aşamada geliştirilmesi şeklinde ilerlemiştir. Her döngüde:

Mevcut bilgi temelinde bir prompt taslağı oluşturulmuş,

Promptun daha etkili olması için ek sorular yöneltilmiş,

Kullanıcıdan alınan detaylara göre prompt yeniden revize edilmiştir.

Girdilerin Spesifikleşmesi: Kullanıcının verdiği bilgiler doğrultusunda aşağıdaki unsurlar netleştirilmiştir:

- Değişiklik, ödeme sonrası oluşturulan sözleşme metninde cayma hakkı oranının değiştirilmesiyle ilgilidir.
- Sistem sadece web ortamında çalışmaktadır ve söz konusu değişiklik, Sozlesme_maddeleri adlı veritabanı tablosunda SM5 ID'sine ait kayıtlarla ilişkilidir.
- Geliştirilecek promptta, yazılım geliştiricilere yönelik sade bir şablonla; gereksinim ID'leri (ör. FR-001), ön koşullar, iş akışı, istisna durumlar gibi alt başlıklar talep edilmiştir.
- Sözleşme maddesinin önceki ve sonraki halinin de raporda yer alması istenmiştir.

Nihai Prompt:

“Yeni geliştirilen bir otel rezervasyonu web uygulamasında, ödeme adımından hemen önce sistemin otomatik sözleşme oluşturma fonksiyonunu etkileyen bir değişiklik için yazılım geliştiricilere yönelik bir *Fonksiyonel Gereksinim Analizi Raporu* hazırla.

Değişiklik şudur: Müşterinin ödeme tutarı **10.000 TL ve üzerindeyse**, sözleşme metninde yer alan cayma hakkı geri ödeme oranı **%40’tan %35’e düşürülecek**.

Bu değişiklik **Sozlesme_maddeleri** adlı veritabanı tablosunda yapılacaktır. SM5 ID’sine karşılık gelen mevcut madde inaktif hale getirilecek ve yeni oranı içeren yeni bir kayıt eklenecektir.

Ayrıca, sözleşmenin önceki ve güncellenmiş hali raporda örnek olarak yer almalıdır.

Her gereksinim için aşağıdaki şablon kullanılsın:

- **ID:** (ör. FR-001)
- **Gereksinim Tanımı**
- **Ön Koşullar**
- **İş Akışı / Davranış**
- **İstisna Durumlar**
- **Notlar (varsa)**

Yazım dili teknik, açık ve kurumsal olmalı. Hedef kitle yazılım geliştiricileridir. Format sade ve sistem odaklı olmalı.”

Sonuç: Tüm bu girdilerin ardından ortaya çıkan nihai prompt, ChatGPT’nin söz konusu gereksinimleri doğru şekilde analiz edip yazılım geliştiriciler için anlaşılır, teknik bir rapor üretmesini sağlayacak şekilde optimize edilmiştir.

Bu uygulama, büyük dil modellerinin doğru yönlendirme (prompt) ile yazılım geliştirme süreçlerinde nasıl aktif olarak kullanılabileceğini göstermekte ve prompt mühendisliği kavramının, yazılım analizi gibi teknik alanlarda da etkili biçimde uygulanabileceğini ortaya koymaktadır.

Fonksiyonel Gereksinim Analizi Raporu (FRAR)

Doküman Adı: Fonksiyonel Gereksinim Analizi – Sözleşme Maddesi Oran Güncellemesi

Modül: Otel Rezervasyon Web Uygulaması – Otomatik Sözleşme Oluşturma

Tarih: 16.06.2025

Hazırlayan: [Analist Adı / Takım]

ID: FR-001

Gereksinim Tanımı

Ödeme adımında, müşteri ödeme tutarı 10.000 TL ve üzerindeyse, otomatik olarak oluşturulan sözleşme metninde yer alan cayma hakkı geri ödeme oranı %40 yerine %35 olarak gösterilmelidir.

Ön Koşullar

- Kullanıcı, rezervasyon işlemini ödeme adımına kadar tamamlamış olmalıdır.
- Ödeme tutarı sistemde doğru hesaplanmış ve erişilebilir olmalıdır.

İş Akışı / Davranış

1. Kullanıcı rezervasyon işlemini gerçekleştirir ve ödeme sayfasına gelir.
2. Sistem, ödeme tutarını kontrol eder.
3. Eğer ödeme tutarı < 10.000 TL ise, mevcut aktif sözleşme maddesi (SM5 – %40 oran) sözleşme metnine dahil edilir.
4. Eğer ödeme tutarı ≥ 10.000 TL ise:
 - Sozlesme_maddeleri tablosunda SM5 ID'sine karşılık gelen madde aktif_mi = 0 olacak şekilde pasif hale getirilir.
 - Aynı başlık ve içerik yapısıyla ancak %35 oranını içeren yeni bir madde kaydı eklenir (SM5B, aktif_mi = 1).
 - Güncel madde, sözleşme şablonuna entegre edilerek kullanıcıya gösterilir.
5. Sistem, oluşturulan sözleşmeyi kullanıcıya gösterir.

Şekil 3. Deney Sonucu Üretilen Fonksiyonel Gereksinim Analiz Dokümanı

4.1.3. Prompt Sıkıştırma ve Özetleme Teknikleri

Sık kullanılan LLM'ler için prompt uzunluğu, hem performans hem de maliyet açısından belirleyici faktördür. Bu yüzden sıkıştırma (compression) ve özetleme (summarization) teknikleri uygulanır.

Yöntemler:

Ön Özetleme (Pre-summarization): Uzun belgeler önce kısaca özetlenir, sonra modele

gönderilir.

Prompt Rewrite (Yeniden Biçimlendirme): Cümleler kısa, doğrudan ve görev odaklı hale getirilir.

Bilgi Temizleme: Gereksiz tekrarlar, bağlam dışı bilgiler çıkarılır.

Kodlama ve Şablon Kullanımı: Standart prompt kalıpları kullanılarak token sayısı azaltılır.

4.2. Token Sayısı Neden Önemlidir?

Büyük dil modellerinde (LLM) metin girdileri ve çıktıları, “token” adı verilen birimlerle temsil edilir. Token, kelimenin tamamı değil; genellikle bir kelimenin tamamı ya da parçasıdır. Örneğin “bilgisayar” kelimesi 1 token olabilirken, İngilizcede “unbelievable” kelimesi 2-3 token’a bölünebilir. LLM’ler hem girdi (prompt) hem de çıktı (response) token’larını dikkate alır.

Bir modelle iletişime geçtiğinizde toplam maliyet = girdi token’ları + çıktı token’ları olur [13].

4.2.1 Token Maliyetleri Neye Göre Belirlenir

Çoğu ticari LLM API sağlayıcısı (OpenAI, Anthropic, Cohere, vs.), token başına ücretlendirme yapar. Örneğin:

GPT-4 Turbo:

Girdi: 0.01 \$ / 1.000 token

Çıktı: 0.03 \$ / 1.000 token

Claude 3 Opus:

Girdi: 0.015 \$ / 1.000 token

Çıktı: 0.075 \$ / 1.000 token

Bu yapıdan dolayı:

Uzun promptlar → daha fazla token → daha fazla girdi maliyeti

Uzun yanıt beklentisi → daha fazla çıktı token’ı → daha fazla çıktı maliyeti

Özetle, her ek kelime, kelime parçası veya örnek, cüzdandan çıkan parayı büyütür.

4.2.2 Context Window Sınırı

Token sayısının sadece maliyetle değil, modelin kapasitesiyle de ilişkisi vardır. Her LLM’in bir “context window” limiti vardır. Bu limit, modelin aynı anda işleyebileceği maksimum token sayısını belirler.

Model	Context Window (maksimum token)
GPT-3.5	~4.000 token
GPT-4 Turbo	128.000 token
Claude 3 Opus	200.000+ token

Tablo 6. Bazı LLM'lerin Context Window Limitleri

Bu limit aşıldığında, model eski bilgileri unutur (context forgetting), işlem yapılamaz (API hatası alırsınız).

Problem	Açıklama
Yüksek Maliyet	Uzun metinlerle çalışmak, kullanım başına maliyeti katlar.
Yanıt Kalitesinde Bozulma	Model, çok uzun girdilerde asıl konudan sapabilir.
Yavaşlama	Token sayısı arttıkça modelin yanıt süresi uzar.
Token Trimming / Unutma	Model bazı önceki bilgileri düşürebilir.

Tablo 7. Fazla Token Kullanmanın Riskleri Tablosu

Optimizasyon İçin Öneriler:

- Gereksiz kelimeleri prompt’tan çıkar.
- Ön özetleme yap.
- Şablonlu, kısa ve görev tanımlı prompt kullan.
- Yanıttan beklentini sınırlayan direktifler ekle: “Lütfen 3 maddeyi geçmesin”, “En fazla 50 kelimeyle açıkla” gibi.

4.3. Sampling Teknikleri (Top-k, Top-p, Temperature)

LLM'lerin aynı girdiye farklı çıktılar verebilmesinin nedeni, sampling (örnekleme) teknikleridir. Bu teknikler, modelin olası kelime adayları arasından hangisini seçeceğine karar verir. Bu da modelin çıktılarının yaratıcılığını, tutarlılığını ve tahmin gücünü doğrudan etkiler [23],[24]. Sampling ayarları sayesinde daha yaratıcı, daha detaylı yanıtlar alabiliriz veya daha kesin, öngörülebilir çıktılar üretebiliriz.

4.3.1 Temperature (Sıcaklık)

Temperature, modelin çıktılarındaki rastgelelik seviyesini belirler.

Temperature Değeri	Davranış
0.0	Deterministik, aynı soruya hep aynı yanıt
0.5	Orta seviye çeşitlilik
1.0+	Yaratıcılık artar, ama tutarlılık azalabilir

Tablo 8. Temperature Değerleri Örnek Tablosu

Kullanım Örneği:

“Temperature = 0.2” → Teknik içerikler, akademik açıklamalar

“Temperature = 0.8” → Hikâye, diyalog, yaratıcı metinler

4.3.2. Top-k Sampling

Top-k, modelin çıktı üretirken en yüksek olasılığa sahip k kelime arasından seçim yapmasına olanak tanır.

Değer Anlamı

k = 1 En olası kelime (deterministik)

k = 50 İlk 50 ihtimal içinden seçim (daha yaratıcı)

Bu yöntem, olasılığı düşük olan anlamsız kelimeleri dışlar ve modelin kontrollü rastgelelikte çalışmasını sağlar.

4.3.3. Top-p (Nucleus Sampling)

Top-p (ya da nucleus sampling), modelin tüm kelime olasılıklarını sıralayıp, toplamda %p ihtimali aşmayan alt kümeyi seçmesini sağlar.

p Değeri	Davranış
p = 0.9	Model, toplam olasılığın %90'ını kapsayan kelimelerden seçer
p = 1.0	Tüm olasılıklar kullanılabilir (en serbest mod)

Top-k sabit sayıda kelimeye bakarken, top-p olasılığa göre esnek küme belirler. Genellikle daha doğal çıktılar sağlar [25],[26],[27].

Teknik	Ne Yapar?	Nerede Kullanılır?
Temperature	Rastgelelik derecesini ayarlar	Yaratıcı vs. teknik görev ayrımı
Top-k	Belirli sayıda en olası kelime seçimi	Kontrollü çeşitlilik
Top-p	Olasılık bazlı esnek seçim	Daha doğal ve dengeli yanıtlar

Tablo 9. Sampling Teknikleri Karşılaştırma Tablosu

Uygulamalı Örnek

Prompt: “Bir kediyle köpeğin sabah sohbetini yaz.”

Ayar	Modelin Çıktısı
Temp = 0.2	“Kedi: Günaydın. Köpek: Günaydın.”
Temp = 0.9	“Kedi: Bugün havada macera var! Köpek: Mamasız sabah olur mu hiç?”

Doğru Sampling Ayarını Nasıl Seçilmeli?

Amaç	Önerilen Ayar
Teknik doküman, özet, analiz	Temp: 0.2–0.4, Top-p: 0.8
Yaratıcı yazı, diyalog, hikâye	Temp: 0.7–1.0, Top-p: 0.95
Kod üretimi	Temp: 0–0.3, Top-k: 1–5

Tablo 10. Doğru Sampling Ayarı Seçim Tablosu

4.4. Model Seçimi

LLM kullanırken hangi modelin seçileceği, performans, hız, doğruluk ve maliyet açısından doğrudan etkilidir. GPT-4, Claude 3 gibi modeller yüksek doğruluk ve bağlam anlama kapasitesine sahipken, GPT-3.5 veya DistilBERT gibi daha küçük modeller hızlı ve ucuz çözümler sunar.

Bu nedenle, her senaryo için “en iyi model” farklı olabilir. Bu başlık altında doğru modeli seçmek için gereken kıyaslama kriterlerini ve önerileri sunuyoruz [2],[28],[29].

Özellik	Açıklama
Parametre Sayısı	Milyarlarca (ör. GPT-4 1T+ tahmini)
Bağlam Anlayışı	Yüksek — uzun metinleri daha iyi kavrar
Genelleme Yeteneği	Farklı görevlerde çok yönlü başarı
Maliyet	Yüksek kullanım ve eğitim maliyeti
Hız	Görece daha yavaş (özellikle API çağrılarında)

Tablo 11. Büyük Modellerin Özellikleri

Özellik	Açıklama
Parametre Sayısı	Milyonlar (ör. DistilBERT ~66M)
Bağlam Anlayışı	Düşük — kısa görevlerde iyidir
Uzmanlaşma Yeteneği	Belirli görevler için optimize edilebilir
Maliyet	Düşük (hem inference hem API maliyeti)
Hız	Çok hızlı yanıt süresi sağlar

Tablo 12. Küçük Modellerin Özellikleri

Karar Verme Kriterleri

Kriter	Büyük Model	Küçük Model
Doğruluk	Daha yüksek	Bazı görevlerde zayıf
Hız	Daha yavaş	Anında yanıt
Maliyet	Yüksek	Uygun

Bağlam Derinliği	Anlayışlı	Sınırlı
Özelleştirme	İnce ayar gerekebilir	Göreve uyarlanabilir

Tablo 13. Karar Verme Kriterleri Tablosu

Hibrid Yaklaşım

Birçok sistem şu stratejiyi uygular:

- Ana görevlerde büyük model → GPT-4
- Sık yinelenen görevlerde küçük model → GPT-3.5 / DistilBERT
- Edge cihazlarda distilled model

Bu yaklaşım hem kaliteyi hem de maliyeti dengeler.

4.5. Önbellekleme (Caching) ile Hızlandırma

Büyük dil modelleri (LLM'ler) sıklıkla benzer içeriklerle karşılaşır. Aynı ya da benzer prompt'ların tekrar tekrar işlenmesi hem zaman hem de maliyet açısından verimsizdir. Bu sorunu çözmek için önbellekleme (caching) yöntemi kullanılır. Önbellekleme, daha önce üretilen yanıtları saklayarak aynı veya benzer talepler geldiğinde modeli yeniden çalıştırmadan hızlıca cevap vermeyi sağlar [30],[31],[32].

Bu yöntemin üç temel faydası vardır: Birincisi, daha önce üretilmiş yanıtları sunarak maliyeti düşürür. İkincisi, doğrudan cache üzerinden cevap verildiği için yanıt süresi önemli ölçüde azalır. Üçüncüsü ise altyapı yükünü hafifleterek API çağrısı veya GPU kullanımı gibi kaynak tüketimlerini azaltır.

Önbellekleme genel olarak iki türde uygulanır:

Statik önbellekleme, aynı prompt'a her zaman aynı yanıtın verildiği durumlardır. Bu yaklaşımda bazı yanıtlar önceden hazırlanır ve kullanıcı sorgu yaptığında doğrudan döndürülür. Örneğin, “ISO 27001 nedir?” gibi sabit bilgi içeren sorular bu tür için uygundur.

Dinamik önbellekleme ise, sıklıkla sorulan ancak küçük farklılıklar içeren prompt'lar için geliştirilmiştir. Burada, benzer prompt'lar normalleştirilerek tek bir önbellek kaydı ile eşleştirilebilir. Bu işlem genellikle hafif düzeyde benzerlik analizleri (fuzzy matching) ile yapılır. Örneğin, “OpenAI ne iş yapar?” ve “Open AI hakkında bilgi verir misin?” gibi farklı ifadeler aynı yanıtı tetikleyebilir.

Önbellekleme sisteminin uygulanışı basit bir süreç izler: Öncelikle daha önceki prompt-yanıt eşleşmeleri kayıt altına alınır. Yeni gelen prompt'lar, hash algoritmaları ya da benzerlik kontrolleri ile karşılaştırılır. Uygun bir eşleşme bulunursa yanıt önbellekten getirilir; yoksa LLM çalıştırılır ve çıkan yanıt önbelleğe eklenir.

Bu sistemin hayata geçirilmesinde Redis, SQLite ve MongoDB gibi veri tabanları sıkça kullanılmaktadır.

Önbellekleme özellikle chatbotlar, yardım masası sistemleri, sık sorulan sorular (FAQ) içeren bilgi tabanları, belge özetleme ve soru-cevap sistemlerinde yaygın olarak kullanılır.

Ancak bazı sınırlamaları da vardır. Önbellekteki bilgiler zamanla güncelliğini yitirebilir. Ayrıca, kişiye özel cevapların gerekli olduğu durumlarda her seferinde modelin yeniden çalıştırılması gerekir. Son olarak, önbelleğin boyutu arttıkça bellekte tutma ve depolama maliyetleri de artabilir.

4.6. Hafif Modeller

Hafif (lightweight) modeller, büyük dil modellerine göre çok daha az sayıda parametreye sahip, hızlı çalışan ve düşük kaynak tüketen yapay zeka sistemleridir. Bu tür modeller, özellikle mobil cihazlar, gömülü sistemler ya da düşük bütçeli projeler için tasarlanmıştır [33],[34],[35].

Her zaman büyük dil modellerinin sunduğu yüksek doğruluk ya da geniş bağlam anlayışına ihtiyaç duyulmaz. Özellikle tek cümlelik sınıflandırmalar, kısa metin özetlemeleri ya da ad varlığı tanıma (NER) gibi dar kapsamlı görevlerde hafif modeller yeterli doğruluğu çok daha düşük maliyetle sunabilir.

Hafif model tercih edilmesinin başlıca nedenleri şunlardır: Eğer sistem yalnızca CPU ile çalışıyorsa veya az miktarda RAM'e sahipse, kaynak kısıtı bir tercih sebebidir. Gerçek zamanlı bir uygulamada çalışıyorsa hız ön plandadır, çünkü hafif modeller gecikmeyi azaltır. Ayrıca bu modeller, herhangi bir API çağrısına gerek kalmadan yerel olarak çalıştırılabildiği için maliyet avantajı da sunar. Son olarak, pil ömrünün kritik olduğu uç cihazlarda enerji verimliliği açısından avantaj sağlar.

Bu ihtiyaçlara yönelik geliştirilen öne çıkan bazı hafif modeller şunlardır:

DistilBERT: Yaklaşık 66 milyon parametreye sahiptir ve BERT modelinin %60 oranında küçültülmüş bir versiyonudur.

TinyBERT: Sadece yaklaşık 14 milyon parametreye sahip olup çok hızlıdır; Android gibi mobil ortamlarda çalışabilir.

MobileBERT: Mobil cihazlar için özel olarak optimize edilmiş ve yaklaşık 25 milyon parametrelidir.

ALBERT: Ağırlık paylaşımı sayesinde model boyutunu düşürür.

MiniLM: Boyutu küçük olmasına rağmen yüksek performans sunar.

TinyLLM: Açık kaynaklı bir modeldir; 100 milyon parametrenin altındadır ve görev odaklı biçimde optimize edilebilir.

Bu modeller genellikle Hugging Face gibi platformlardan kolayca erişilebilir.

Kullanım alanları oldukça çeşitlidir. Örneğin, dijital asistan veya kamera içi analiz yapan mobil uygulamalarda, IoT cihazları gibi gömülü sistemlerde, gerçek zamanlı çalışan chatbotlarda ya da akıllı cihazlara entegre edilen yorumlayıcılarda kullanılır. Ayrıca, internetsiz ortamlarda çalışması gereken güvenlik sistemlerinde de tercih edilir.

Ancak hafif modellerin bazı sınırlamaları da vardır. Bağlam anlama kapasiteleri sınırlıdır, bu nedenle uzun metinlerde yetersiz kalabilirler. Ayrıca genellikle belirli bir görev için eğitildiklerinden, çok yönlü kullanım için uygun değildir. Eğitildikleri görev dışındaki alanlarda hata yapma ihtimalleri daha yüksektir.

Büyük modellerle karşılaştırıldığında, hafif modeller daha hızlı çalışır. Doğruluk açısından büyük modeller daha başarılıyken, hafif modellerin doğruluğu görev türüne bağlı olarak değişir. Hafif modeller daha az kaynak kullanır, kurulumu genellikle tek bir dosya ile yapılabilir ve büyük modellere göre daha kolaydır. Büyük modeller genellikle genel amaçlı olarak kullanılırken, hafif modeller belirli görevlere odaklanır.

5. LLM PERFORMANS VE MALİYET KARŞILAŞTIRMASI

5.1. Kalite-Hız-Maliyet Üçgeni

Büyük dil modelleri (LLM) kullanılırken verilmesi gereken en kritik kararlardan biri, kalite ile maliyet arasındaki dengeyi kurmaktır. En kaliteli modeli tercih etmek ilk bakışta cazip görünse de, uygulamada bu karar genellikle bütçe, hız, görev türü ve sistemin ölçeklenebilirliği gibi faktörlere göre şekillenir. Bu bölümde, farklı kullanım senaryolarına

göre nasıl bir denge stratejisi izlenmesi gerektiği açıklanmaktadır [2], [11].

Kalite, modelin çıktısının doğruluğu, bağlama uygunluğu, yaratıcılığı ve tutarlılığı gibi ölçütlerle değerlendirilir. GPT-4 gibi gelişmiş modeller bu alanlarda üstün performans gösterir.

Maliyet ise birkaç unsurdan oluşur: girdi ve çıktıdaki token sayısı, modelin API ile çağırılma maliyeti, işlem süresi ve gereken donanım düzeyi ile önbellekleme ve veri depolama gibi ek maliyetler. Örneğin, GPT-4 kullanımı, GPT-3.5'e göre 4 ila 8 kat daha maliyetli olabilir [36],[37],[38].

Kalitenin öncelikli olduğu durumlar genellikle kritik alanlara denk gelir. Örneğin tıbbi, hukuki veya akademik içerik üretiminde hata payı kabul edilemez. Benzer şekilde, uzun ve çok bağlamlı belgelerin özetlenmesinde bağlam anlayışı önemlidir. Prestijli içeriklerde, örneğin resmi raporlarda ya da marka mesajlarında, kalite doğrudan marka imajını etkiler. Bu gibi durumlarda GPT-4 veya Claude 3 Opus gibi yüksek doğruluklu modeller tercih edilmelidir.

Maliyetin öncelikli olduğu senaryolar ise daha çok tekrar eden ya da düşük karmaşıklığa sahip işlemleri kapsar. Örneğin, sıkça yinelenen kullanıcı sorularında, aynı cevabı üretmek için pahalı modelleri çağırmak gereksizdir. Kısa metin etiketleme ya da ad varlığı tanıma (NER) gibi işlemlerde büyük modellere gerek yoktur. Gerçek zamanlı chatbot sistemlerinde gecikme süresi önemlidir, bu nedenle daha hızlı ve hafif modeller kullanılır. Ayrıca offline ya da mobil cihazlarda çalışacak uygulamalarda donanım sınırlı olduğu için küçük modeller tercih edilir. Bu bağlamda GPT-3.5, DistilBERT veya TinyLLM gibi modeller daha uygun çözümler sunar.

Aşağıda, çeşitli görevlerde kullanılabilecek model tercihlerini gösteren örnek bir performans-maliyet matrisi yer almaktadır:

Ürün yorumu analizi için DistilBERT yeterlidir; kalite orta düzeyde, maliyet düşüktür.

Bilimsel makale özetlemede GPT-4 yüksek kalite sunar ama maliyeti de yüksektir.

Chatbot uygulamalarında sık sorular için GPT-3.5 ile önbellekleme birlikte kullanılır; orta kalite ve düşük maliyet sağlanır.

Teknik destek e-postaları için Claude 3 Haiku gibi bir model tercih edilirse kalite yüksek, maliyet orta seviyededir.

Son olarak, hibrit yaklaşım olarak adlandırılan bir strateji de mevcuttur. Bu yöntemde sistem, ucuz ve hızlı modeli varsayılan olarak kullanır, ancak kalite beklentisi oluştuğunda daha gelişmiş modeli devreye sokar. Bu, “model rotasyonu” ya da “kaliteye bağlı çağrı sistemi” olarak bilinir. Örneğin, kullanıcı yalnızca bilgi isterse GPT-3.5 yeterli görülür; ancak kullanıcı “akademik kaynakla destekle” şeklinde bir talepte bulunduğunda sistem otomatik olarak GPT-4’e geçer

5.2. Maliyet Hesaplamaları: Token Başına Ücret

Büyük dil modelleri (LLM), genellikle token bazlı bir ücretlendirme sistemine göre çalışır. Kullanıcının verdiği giriş (prompt) ve modelin ürettiği yanıt (completion) belirli sayıda token içerir. Bu iki token sayısı toplanır ve kullanılan modelin token başına belirlenmiş fiyatıyla çarpılarak toplam maliyet hesaplanır [1],[12],[13].

Token kavramı, bir kelimenin tamamı ya da bir parçası olabilir. Örneğin, "OpenAI" gibi kısa bir ifade tek bir token olabilirken, “Yapay zekâ sistemlerinin” ifadesi 4–5 token’a karşılık gelebilir.

Her modelin token başına ücretlendirmesi farklıdır. Örneğin, GPT-4 Turbo modeli için 1.000 token başına giriş ücreti 0.01 dolar, yanıt ücreti ise 0.03 dolardır. GPT-3.5 Turbo için bu ücretler sırasıyla 0.001 ve 0.002 dolar olarak belirlenmiştir. Claude 3 Opus modeli, 1.000 token giriş için 0.015 dolar, çıkış için 0.075 dolar alırken; Claude 3 Haiku’da bu değerler sırasıyla 0.0008 ve 0.0025 dolardır. Elbette bu fiyatlar zaman içinde ve sağlayıcıya göre değişebilir (Kaynak: OpenAI, Anthropic, 2024).

Maliyeti hesaplamak için kullanılan formül şu şekildedir:
Toplam Maliyet = (Girdi Token Sayısı × Girdi Fiyatı) + (Çıktı Token Sayısı × Çıktı Fiyatı)

Bu formül doğrultusunda bir örnek verelim: GPT-4 ile 500 token’lık bir giriş ve 300 token’lık bir yanıt üretildiğinde, maliyet $(500 \times 0.01 / 1000) + (300 \times 0.03 / 1000) = 0.005 + 0.009 = 0.014$ dolar olacaktır.

Aynı işlem GPT-3.5 ile yapılırsa maliyet $(500 \times 0.001 / 1000) + (300 \times 0.002 / 1000) = 0.0005 + 0.0006 = 0.0011$ dolar olur. Bu da GPT-3.5’in aynı görevi yaklaşık 12 kat daha ucuza gerçekleştirdiğini gösterir.

Maliyeti etkileyen başlıca faktörler arasında kullanılan toplam token sayısı yer alır; bu değer doğrudan maliyeti belirler. Kullanılan modelin tipi çok büyük bir fiyat farkı

yaratabilir. Ayrıca modelin verdiği yanıt genellikle girişe göre daha uzun olduğu için yanıt kısmı daha pahalıya mal olabilir. Sampling ayarları da yanıtın uzunluğunu etkileyebilir; örneğin, yüksek "temperature" değeri daha uzun ve çeşitlendirilmiş yanıtlar üretebilir. Öte yandan, önbellekleme sistemi kullanıldığında, daha önce üretilmiş bir yanıt yeniden kullanılabilir ve bu da maliyeti önemli ölçüde düşürür.

Maliyetleri azaltmak için bazı optimizasyon önerileri sunulabilir. Öncelikle, prompt'lar mümkün olduğunca sade tutulmalıdır; gereksiz uzunluklardan kaçınılmalıdır. Yanıt uzunluğu sınırlandırılabilir; örneğin "100 kelimeyle özetle" veya "3 madde yaz" gibi talimatlarla. Her görev için en pahalı modeli kullanmak yerine, uygun fiyat-performans dengesine sahip bir model seçilmelidir. Ayrıca giriş verisi token trimming veya özetleme teknikleriyle kısaltılabilir. Sık tekrarlanan sorgular için ise önbellekleme uygulanması son derece verimli bir çözümdür.

6. Güncel Yaklaşımlar ve Gelecek Trendler

6.1. Retrieval-Augmented Generation (RAG) Sistemi

Büyük dil modelleri (LLM), yalnızca eğitim sırasında gördükleri veri ile sınırlı olduğundan, özellikle güncel bilgiye dayanan sorularda yanlış veya uydurma (hallucination) içerikler üretebilir. Bu sınırlamayı aşmak için geliştirilen Retrieval-Augmented Generation (RAG) yaklaşımı, LLM'lerin harici bilgi kaynaklarıyla entegre biçimde çalışmasını sağlayan modern bir çözümdür [41].

RAG, LLM'lerin bilgi üretmeden önce dış kaynaklardan (veritabanı, belge, API vb.) bilgi çekmesine olanak tanıyan bir tekniktir. Bu sayede model, eğitildiği veriye ek olarak anlık, güncel ve bağlama özel bilgilere ulaşarak daha doğru ve güvenilir yanıtlar üretir [2].

Bu yaklaşım şu iki aşamadan oluşur:

Bilgi Getirme (Retrieval): Kullanıcının sorgusuna uygun belgeler, vektör tabanlı arama yoluyla bulunur.

Yanıt Üretimi (Generation): Elde edilen belgeler modele bağlam olarak sunulur ve model bu bilgilerle desteklenmiş yanıt üretir.

Retrieval-Augmented Generation (RAG) mimarisinde genellikle üç temel bileşen bulunur. İlk olarak, **gömme (embedding) modeli**, hem kullanıcıdan gelen sorguyu hem de

sistemdeki belgeleri vektörlere dönüştürerek anlam temelli karşılaştırmalar yapılmasını sağlar. Ardından, **retriever** adlı bileşen, vektör benzerliği hesaplamaları ile en alakalı belgeleri seçer. Son olarak, büyük dil modeli (LLM) bu seçilen belgeleri kullanarak nihai yanıtı üretir.

RAG sistemleri, büyük dil modellerinin bazı temel sınırlamalarını aşmak için geliştirilmiştir. Örneğin, LLM'ler genellikle güncel bilgi içermez; ancak RAG sayesinde anlık olarak harici kaynaklardan bilgi çekilebilir. Ayrıca, modelin uydurma bilgi üretme riski (halüsinasyon) RAG ile azaltılır çünkü yanıtlar gerçek verilere dayanır. Dil modellerinin bağlam penceresi sınırlı olduğu için tüm veriyi işleyemez; fakat RAG sadece ilgili belgeleri sisteme dahil ederek bu sorunu çözer. En önemlisi ise, yeni bilgiler için modeli baştan eğitmek gerekmez; yalnızca bilgi kaynakları güncellenerek sistem yenilenmiş olur.

Bu yapının gerçek dünyadaki uygulamaları oldukça çeşitlidir. Örneğin eğitim alanında, öğrenciler PDF veya sunum dosyaları üzerinden soru sorabilir. Hukukta, mevzuat belgelerine dayalı olarak hukuki danışmanlık yapılabilir. Müşteri hizmetlerinde, kurumsal dökümanlara dayanarak yanıt veren chatbot'lar geliştirilebilir. Arama motorları ise, kaynaklı özet yanıtlar sunmak amacıyla RAG mimarisinden faydalanabilir; buna örnek olarak Perplexity.ai gösterilebilir.

Teknik altyapı açısından RAG sistemleri dört ana bileşenden oluşur:

Embedding işlemleri için OpenAI'nin text-embedding-ada-002 modeli, BERT ya da SBERT gibi araçlar kullanılabilir.

Vektör veritabanı tarafında FAISS, Pinecone ve Supabase gibi sistemler tercih edilebilir.

Dil modeli olarak GPT-3.5, GPT-4, Claude ya da LLama gibi gelişmiş modeller kullanılabilir.

Bu bileşenlerin entegrasyonunda ise LangChain, LlamaIndex ya da Haystack gibi framework'ler öne çıkar.

RAG sistemlerinin birçok avantajı vardır. Yanıtlar bilgiye dayalı şekilde üretildiği için doğruluk artar. Uydurma bilgi üretimi riski düşer. Ayrıca yeni veriler eklemek için modelin yeniden eğitilmesine gerek kalmaz. Yanıtlar kaynak gösterilerek desteklenebilir ve mevcut sistemlere entegrasyonu görece kolaydır.

Ancak bazı zorluklar da vardır. Yanıtın kalitesi, geri getirilen belgelerin kalitesine doğrudan bağlıdır. Sistemin kurulumu karmaşıktır çünkü vektör veritabanı, embedding modelleri ve LLM'in birlikte çalışması gerekir. Ayrıca, anlamsal eşleşme hataları olabilir; yani sistem bazen yanlış belgeleri seçebilir.

6.1.2. RAG Sistemi Uygulaması

Üniversitelerde birçok kural ve düzenleme, resmi yönetmelik belgeleriyle tanımlanmıştır. Bu belgeler genellikle PDF formatında olup oldukça uzun ve karmaşık yapıya sahiptir. Bu durum, öğrencilerin spesifik sorularına hızlıca cevap bulmalarını güçleştirmektedir. Bu bağlamda, proje kapsamında Marmara Üniversitesi'nin lisans ve önlisans yönetmeliğini temel alan bir Retrieval-Augmented Generation (RAG) sistemi kurulmuştur.

RAG, büyük dil modelleri (LLM) ile vektör tabanlı bilgi arama sistemlerini birleştiren bir mimaridir. Sistemin genel yapısı aşağıdaki gibidir:

- **Getirici Katman (retriever):** Kullanıcının sorusuna en uygun belge parçacıkları vektör arama yoluyla bulunur.
- **Üretici Katman (LLM):** Bulunan içerikler ile birlikte kullanıcının sorusu modele verilir, model bağlama uygun bir yanıt üretir.

Bu projede kullanılan mimari şu bileşenlerden oluşmaktadır:

- **Vektör veritabanı:** Supabase üzerinde yapılandırılmış bir tablo içinde, PDF dosyasından parçalara ayrılmış içerik, mxbai-embed-large modelinin vektör temsilleri ile depolanmıştır.
- **Dil modeli (LLM):** llama3.2:1b modeli, yerel ortamda Ollama aracılığıyla çalıştırılmıştır.
- **Sorgu ve yanıt:** Kullanıcı terminal üzerinden doğal dilde soru sorar. Sistem, ilgili belge bölümlerini bulur ve LLM'e aktararak kaynaklı bir yanıt üretir.

Uygulama Akışı

Kod yapısı iki temel dosyaya ayrılmıştır:

db_yerlestir.py

- Yönetmelik PDF dosyasını yükler.
- Metni parçalara ayırır (chunk).
- Vektör embedding'leri üretir (mxbai-embed-large modeli ile).
- Parçaları Supabase veritabanına işler.

```
18 #supabase db yi başlatır
19 supabase_url = os.environ.get("SUPABASE_URL")
20 supabase_key = os.environ.get("SUPABASE_SERVICE_KEY")
21 supabase: Client = create_client(supabase_url, supabase_key)
22
23 # gömme modelini başlatır
24 embeddings = OllamaEmbeddings(model="mxbai-embed-large")
25
26 #documents klasöründeki pdf dosyalarını yükler
27 loader = PyPDFDirectoryLoader("documents")
28
29 # belgeleri chunk'lara böler.
30 documents = loader.load()
31 text_splitter = RecursiveCharacterTextSplitter(chunk_size=1000, chunk_overlap=100)
32 docs = text_splitter.split_documents(documents)
33
34 # chunkları vektör store'da saklar.
35 vector_store = SupabaseVectorStore.from_documents([
36     docs,
37     embeddings,
38     client=supabase,
39     table_name="documents",
40     query_name="match_documents",
41     chunk_size=1000,|
42 ])
```

Şekil 4. Dokümanı parçalara bölüp vektöre dönüştüren kod.

klasik_rag.py

- Kullanıcıdan gelen doğal dilde soruyu alır.
- Vektör veritabanında en ilgili belgeleri arar.
- Bu belgeleri LLM'e uygun bir promptla birlikte gönderir.
- Modelden gelen yanıt kullanıcıya yazılı olarak sunar.

```

30
31 print("RAG Chatbot'a hoş geldiniz! Çıkmak için 'q' yazın.\n")
32 while True:
33     user_question = input("Kullanıcı: ")
34     if user_question.strip().lower() == "q":
35         print("Görüşmek üzere!")
36         break
37
38     # RAG pipeline
39     relevant_docs = vector_store.similarity_search(user_question, k=3)
40     sources = []
41     docs_content = []
42     for doc in relevant_docs:
43         src = doc.metadata.get("source", "Bilinmeyen kaynak")
44         sources.append(f"{src}")
45         docs_content.append(doc.page_content)
46

```

Şekil 5. Kullanıcı girdisini alıp veritabanında vektörel eşleme yapan kod.

Bileşen	Araç	Açıklama
LLM	llama3.2:1b (Ollama)	Hafif, lokal çalışabilir, düşük kaynak tüketimi
Embedding	mxbai-embed-large	Açık kaynaklı ve vektör performansı yüksek
Vektör Store	Supabase	Open source altyapı, veritabanı görselleştirmeye izin veriyor
Arayüz	Terminal	Kullanıcıdan doğrudan input alınan basit CLI yapısı

Sonuç: Bu proje kapsamında oluşturulan sistem, belirli bir belgeye dayalı bilgi erişimini akıllı ve etkili hale getirmiştir. Öğrenciler terminal üzerinden doğal dilde sorular sorarak, Marmara Üniversitesi yönetmeliğinden hızlı ve anlamlı yanıtlar alabilmektedir.

Geliştirilen çözüm, sınırlı kaynaklarla çalışan açık kaynak modelleri kullanmasına rağmen RAG mimarisinin temel felsefesini başarıyla yansıtmaktadır. Gerekirse ilerleyen aşamalarda bu yapı, kullanıcı arayüzü (web/chatbot), çoklu belge desteği veya farklı LLM entegrasyonlarıyla daha geniş bir kullanım alanına taşınabilir.

```
Kullanıcı: marmara üniversitesinde devam şartı yüzde kaçtır?  
Chatbot: content="Marmara Üniversitesi'nde devam şartı yüzde 80'dir." additional_k  
l': 'llama3.2:1b', 'created_at': '2025-06-17T17:09:53.885018Z', 'done': True, 'done  
n': 6369503875, 'load_duration': 42935375, 'prompt_eval_count': 1081, 'prompt_eval  
nt': 15, 'eval_duration': 477149375, 'model_name': 'llama3.2:1b'} id='run--53045b0  
usage_metadata={'input_tokens': 1081, 'output_tokens': 15, 'total_tokens': 1096}
```

Şekil 6. Sisteme Sorulan Devam Şartı Sorusu ve Yanıt.

```
Kullanıcı: mezun olabilmek için GANO minimum kaç olmalıdır  
Chatbot: content='Kullanıcı sorusu: Mezuniyet için GANO minimum kaç olmalıdır\n\nCevap: 2.00' additi  
response_metadata={'model': 'llama3.2:1b', 'created_at': '2025-06-17T10:18:31.072017Z', 'done': True  
: 'stop', 'total_duration': 5831097000, 'load_duration': 45902250, 'prompt_eval_count': 953, 'prompt  
: 4951540500, 'eval_count': 26, 'eval_duration': 831140291, 'model_name': 'llama3.2:1b'} id='run--13  
8-bf16-989f9b34fa55-0' usage_metadata={'input_tokens': 953, 'output_tokens': 26, 'total_tokens': 979}
```

Şekil 7. Sisteme Sorulan GANO Sorusu ve Yanıt

6.2. AI Geribildirim, Anayasaya Dayalı Yapay Zeka ve Etik Kurallar

Büyük dil modelleri güçlü olsalar da, bazen zararlı, yanıltıcı veya uygunsuz yanıtlar verebilir. Bu, sadece teknik değil aynı zamanda etik bir sorundur. Yapay zekanın daha güvenli, şeffaf ve insana saygılı hale gelmesi için geliştirilen bazı eğitim yöntemleri son yıllarda öne çıkmıştır.

Bu başlık altında özellikle Constitutional AI, AI Feedback, Preference Modeling ve benzeri teknikler üzerinden yapay zekanın davranışlarının nasıl şekillendirildiği ele alınacaktır [1].

6.2.1. Anayasaya Dayalı Yapay Zeka (Constitutional AI)

Constitutional AI (CAI), bir yapay zeka modelinin çıktılarının insanlar tarafından değil, önceden tanımlanmış etik kurallar çerçevesinde yönlendirilmesini esas alan bir yaklaşımdır [19]. Bu kurallar bir nevi modelin “anayasası” gibi davranır.

Bu sistemin çalışma prensibi şu şekilde işler: İlk olarak, modelin uyması gereken etik prensipler belirlenir. Örneğin, modelin zararlı bilgi vermemesi, tarafsız davranması veya empatik olması gibi kurallar bu çerçevede tanımlanır. Daha sonra, model kendi verdiği yanıtları bu kurallara göre değerlendirir. Bu değerlendirme süreci, geleneksel yöntemlerde olduğu gibi bir insan tarafından değil, genellikle başka bir model tarafından gerçekleştirilir. Eğer verilen yanıt kurallara aykırı bulunursa, model bu yanıtı alternatif olarak kurallara uygun bir cevap üretir.

Bu yaklaşıma bir örnek vermek gerekirse: Kullanıcı modelden “Nasıl yasa dışı bir yazılım

yapılır?” şeklinde bir soru sorduğunda, CAI modeli bu isteğin etik olmadığını belirtir ve doğrudan cevap vermek yerine, “Bu tür bilgilendirme etik değildir. Ancak yazılım güvenliği hakkında genel bilgiler sunabilirim.” şeklinde yönlendirici ama güvenli bir yanıt üretir.

CAI sistemlerinin bazı temel avantajları da vardır. Öncelikle, insan müdahalesine olan ihtiyacı azaltır; çünkü etik kontrol süreçleri model içinde gerçekleşir. Ayrıca, bu yaklaşım sayesinde model davranışlarında daha fazla tutarlılık ve standardizasyon sağlanabilir. En önemlisi de, model çıktılarının güvenliği ve toplumsal sorumluluğu açısından daha güvenilir sistemler geliştirilmesine katkı sunar.

6.2.2 AI Geribildirimi (AI Feedback)

AI Feedback, bir yapay zeka modelinin ürettiği çıktının insanlar yerine başka bir yapay zeka modeli tarafından denetlenmesi sürecine verilen isimdir. Bu yöntemin temel amacı, denetim sürecini daha ölçeklenebilir ve otonom hale getirerek modeli sürekli olarak iyileştirebilmektir [12].

Bu yapının işleyişi şu şekildedir: Öncelikle, ana model bir yanıt üretir. Ardından ikinci bir model devreye girerek bu yanıtı değerlendirir; örneğin yanıtın doğru olup olmadığını, etik kurallara uygunluğunu ya da bağlama ne kadar uygun olduğunu analiz eder. Değerlendirme sonucuna göre model ya yönlendirilir ya da gerekirse yeniden eğitilir.

AI Feedback yöntemi, özellikle Constitutional AI yaklaşımlarında sıkça kullanılmaktadır. Çünkü bu yapılarda etik kuralları denetlemek için insanlar yerine model temelli geri bildirim sistemi daha tutarlı ve verimli bir alternatif sunar.

6.2.3. Tercih Modelleme (Preference Modelling)

Bu yöntem, yapay zeka modellerinin çıktılarının kullanıcı beğenisine daha uygun hâle gelmesini hedefler. Bunu gerçekleştirmek için kullanıcıların birden fazla yanıt arasındaki tercihlerinden yararlanır. Sürecin işleyişi şu şekilde özetlenebilir: Kullanıcıya iki farklı yanıt sunulur ve hangisini tercih ettiği sorulur. Bu tercihlerden yola çıkarak model, bir “ödül fonksiyonu” öğrenir. Ardından, gelecekte üreteceği yanıtları bu öğrendiği kullanıcı eğilimlerine göre şekillendirmeye başlar[39],[40].

Bu yaklaşım, Reinforcement Learning with Human Feedback (RLHF) yani “İnsan Geri Bildirimi ile Pekiştirmeli Öğrenme” yönteminin temelini oluşturur. RLHF, modelin eğitim

sürecine insan tercihlerine dayalı yönlendirmeyi dâhil eder.

Yapay zekâ sistemlerinde kullanılan başlıca yönlendirme yaklaşımları birkaç farklı kaynaktan beslenir:

- Constitutional AI, önceden tanımlanmış etik kurallara dayanır. Genellikle güvenlik ve etik hassasiyetin ön planda olduğu yardımcı yapay zekâ uygulamalarında kullanılır.
- AI Feedback, bir yapay zekâ modelinin başka bir yapay zekâ tarafından değerlendirilmesini temel alır. Bu yaklaşım, otomasyon odaklı sistemlerde, özellikle de Constitutional AI ya da modelin kendi çıktısını eleştirdiği self-critique senaryolarında öne çıkar.
- Preference Modeling, doğrudan gerçek kullanıcı tercihlerini esas alır ve sohbet botları ile içerik üretimi gibi alanlarda kullanıcı uyumunu artırmak amacıyla kullanılır.

Gerçek dünyada bu yöntemlerin uygulandığı örnekler oldukça yaygındır. Örneğin Anthropic, Claude modellerinde Constitutional AI yaklaşımını benimsemiştir. OpenAI, GPT modellerinde RLHF tekniklerini kullanmakta ve aktif olarak kullanıcı tercihlerini toplamaktadır. Meta AI, “Yardımsaver, Dürüst, Zararsız” (Helpful, Honest, Harmless) kurallarını kullanarak model davranışlarını yönlendirmektedir. Google DeepMind ise “Sparrow” adlı projesiyle daha etik ve güvenli model çıktıları üretmeyi amaçlamaktadır.

6.3. LLM’lerin Güncel Kalması İçin Teknikler (Retraining vs RAG)

Büyük dil modelleri (LLM), yalnızca eğitildikleri döneme ait verilerle sınırlıdır. Örneğin, bir model 2023 verileriyle eğitildiyse, 2024 yılında meydana gelen olaylar hakkında bilgi sahibi olamaz. Bu durum özellikle haber, mevzuat, teknoloji ve trendler gibi sürekli değişen alanlarda önemli bilgi eksikliklerine yol açar.

Bu güncellik sorununu aşmak için başlıca iki yöntem öne çıkmaktadır: modelin yeniden eğitilmesi (retraining) ve bilgiye ihtiyaç duyulduğu anda dış kaynaklardan erişilmesini sağlayan RAG (Retrieval-Augmented Generation) yöntemi. Her iki yaklaşımın da kendine özgü avantajları ve sınırlamaları bulunmaktadır [41].

Retraining (Yeniden Eğitim) yönteminde, modelin eğitiminde kullanılan veri havuzuna yeni bilgiler eklenerek model sıfırdan ya da kısmen yeniden eğitilir. Bu yöntemin önemli

avantajlarından biri, güncel bilgilerin doğrudan modelin içine yerleştirilmesidir. Böylece model, öğrenmiş olduğu yeni bilgilerle bilgi boşluklarını kapatabilir ve yanıtlarında daha güçlü bir doğruluk sunabilir. Ayrıca önceden var olan önyargılar düzeltilebilir. Ancak bu yöntemin ciddi dezavantajları da vardır: Eğitim süreci hem zaman hem de kaynak açısından son derece maliyetlidir. Haftalar sürebilen bu süreç, çok sayıda GPU ve yüksek teknik uzmanlık gerektirir. Dahası, sürekli yeniden eğitim yapmak sürdürülebilir değildir ve modelin tüm parametrelerinin güncellenmesi istenmeyen yan etkilere yol açabilir. Örneğin, GPT-3'ten GPT-4'e geçiş sırasında trilyonlarca token'lık yeniden eğitim işlemi uygulanmıştır [2],[42].

Retrieval-Augmented Generation (RAG) yöntemi ise modeli baştan eğitmek yerine, sorgu anında dış bilgi kaynaklarından verileri çekerek bağlamı zenginleştirir. Böylece model, güncel bilgiye ulaşarak daha doğru yanıtlar verebilir. Bu yaklaşımın öne çıkan avantajı, modelin tekrar eğitilmesine gerek olmamasıdır. Bu da süreci hem daha hızlı hem de daha düşük maliyetli hâle getirir. Aynı zamanda, bilgi tabanları belirli görevlere veya kullanıcı profillerine göre özelleştirilebilir ve yanıtlara kaynak da gösterilebilir. Ancak bu yöntemin bazı sınırlamaları vardır: Elde edilen bilginin kalitesi, getirilen belgelerin doğruluğuna doğrudan bağlıdır. Ayrıca, uygulama katmanı teknik açıdan karmaşıktır; embedding, retrieval ve veri entegrasyonu gibi süreçlerin düzgün çalışması gerekir. En önemlisi ise, model kendi başına değil, çekilen bilgiye bağımlı olduğu için hayati önem taşıyan görevlerde güvenlik riski oluşturabilir.

Kriter	Retraining	RAG
Güncellik Süresi	Aylar arayla	Gerçek zamanlı
Maliyet	Yüksek	Düşük
Uygulama Karmaşıklığı	Eğitim altyapısı gerekir	API ve retrieval sistemi gerekir
Bilgi Kalıcılığı	Kalıcı (ağırlıklara yerleşir)	Geçici (belgeye bağlı)
Esneklik	Düşük	Yüksek (veri tabanı anında değişebilir)

Tablo 14. RAG ve Retraining Metodları Karşılaştırması

Günümüzde birçok gelişmiş LLM altyapısı bu iki yöntemi birlikte kullanmayı tercih etmektedir. Bu **hibrit stratejide**, yılda bir yapılan retraining ile modelin temel bilgisi güncellenirken, günlük veya saatlik değişen veriler için RAG kullanılarak dinamik bir bilgi desteği sağlanır. Bu yaklaşım, modelin hem güncel hem de bütünlüklü bilgi sunmasını mümkün kılar.

6.4. Güvenilirlik, Hallucination ve Yanıltıcı Bilgi Problemleri

LLM'ler doğal dil üretmede oldukça başarılıdır, ancak bazen yanlış, uydurma (halüsinatif) veya yanıltıcı bilgiler verebilirler. Bu durum; akademik, tıbbi, hukuki veya güvenlik hassasiyeti olan alanlarda ciddi risklere yol açabilir.

Bu bölümde LLM'lerin neden halüsinasyon yaptığını, bu sorunun nasıl tespit edildiğini ve azaltılmasına yönelik güncel yaklaşımları inceleyeceğiz [43],[12],[22].

6.4.1. Halüsinasyon (Hallucination) Nedir?

Halüsinasyon, büyük dil modellerinin (LLM) doğruymuş gibi görünen ancak aslında hatalı veya uydurma bilgiler üretmesi durumuna verilen isimdir. Bu durumun temelinde, modellerin gerçeklik denetimi yapmaksızın yalnızca kelime olasılıklarına dayalı tahminlerle çalışması yatar. Bu nedenle bazı yanıtlar, akıcı ve mantıklı görünse de tamamen temelsiz olabilir.

Örneğin kullanıcı “Albert Einstein’ın 2019’daki konferans konuşmalarını özetle.” şeklinde bir prompt verdiğinde, model şu şekilde bir yanıt verebilir: “Einstein 2019’da yaptığı konuşmasında yapay zekânın etik boyutuna değindi...” Oysa Einstein 1955 yılında vefat etmiştir. Bu yanıtın tamamı uydurmadır.

Halüsinasyonlar farklı türlerde ortaya çıkabilir:

Faktüel hatalar, örneğin kişi, tarih, yer veya kavramlara dair yanlış bilgi sunulmasıdır. “Newton 21. yüzyılda yaşamıştır” gibi bir ifade buna örnektir.

Kaynak uydurma, aslında var olmayan makale, kitap ya da çalışmaları referans olarak göstermektir.

Atıf karmaşası, bir kişiye ait sözü başka bir kişiye atfetmek gibi yanlış ilişkilendirmeleri

içerir.

Kurgusal detay ekleme, kullanıcının sorusunda yer almayan bilgileri modelin kendiliğinden üretmesidir.

Bu tür hatalar birkaç temel nedene dayanır. İlk olarak, dil modelleri sadece kelimeler arasındaki istatistiksel ilişkilere göre çalışır; yani gerçekliği test etmezler. İkinci olarak, modelin eğitildiği veri kaynakları eksik, hatalı veya kendi içinde çelişkili olabilir. Üçüncü olarak, kullanıcıdan gelen prompt yeterince açık değilse, model bağlamdaki boşlukları kendi “tahmin” gücüyle doldurabilir. Son olarak, modeller yalnızca eğitildikleri döneme kadar olan bilgilere sahiptir; güncel olayları bilemezler. Bu tür eksikleri gidermek için RAG (Retrieval-Augmented Generation) gibi yöntemler önerilmektedir (detaylar için bkz. 6.1).

Halüsinasyon riskini azaltmak ve modelin güvenilirliğini artırmak için bazı yöntemler mevcuttur:

RAG kullanımı, modelin dış kaynaklardan belge çekerek bilgiye dayalı yanıt üretmesini sağlar.

Chain-of-Thought (CoT) prompting, modelin karmaşık sorulara adım adım düşünerek cevap vermesini teşvik eder.

Yönlendirici prompt yazımı, örneğin “Yalnızca doğrulanabilir bilgilere dayanarak yanıt ver” gibi talimatlar verilerek modelin çerçevesi netleştirilebilir.

Yanıt sınırlama, “Eğer emin değilsen ‘bilmiyorum’ yanıtını ver” gibi doğrudan kısıtlamalar getirerek halüsinasyonların önüne geçebilir.

RLHF (Reinforcement Learning with Human Feedback) yöntemiyle, uydurma içerik üreten yanıtlar düşük ödüllerle cezalandırılarak model bu davranıştan uzaklaştırılabilir.

6.5. Açık Kaynak LLM’lerin Yükselişi

Büyük dil modellerinin (LLM) gelişimi, başlangıçta yalnızca büyük teknoloji şirketlerinin (OpenAI, Google, Anthropic) öncülüğünde, kapalı kaynaklı sistemler etrafında şekillenmiştir. Ancak son yıllarda, açık kaynaklı LLM’lerin yükselişi, bu alanda önemli bir paradigma değişimini beraberinde getirmiştir. Açık kaynak modeller, performans

açısından giderek kapalı modellerle rekabet edebilir hale gelirken; özelleştirilebilirlik, maliyet etkinliği, şeffaflık ve araştırma erişilebilirliği gibi açılardan da önemli avantajlar sunmaktadır [44],[45],[46],[47],[48].

6.5.1. Açık Kaynak LLM'lerin Tanımı ve Temel İlkeleri

Açık kaynak LLM'ler, eğitim ağırlıkları (model weights), model mimarisi, eğitim verisi yapısı ve kullanım lisansları gibi bileşenlerin kamuya açık olarak paylaşıldığı, genellikle akademik veya ticari kullanım için serbestçe erişilebilen modellerdir. Bu modeller, yerel sunucularda çalıştırılabilir; gerektiğinde görev özelinde yeniden eğitilebilir (fine-tuning) ya da farklı uygulamalara entegre edilebilir [2].

6.5.2. Yükselişi Tetikleyen Faktörler

Açık kaynak yapay zeka modellerinin yükselişinde birden fazla motivasyon etkili olmuştur. En başta gelen nedenlerden biri veri güvenliği ve gizliliğidir. Kurumlar, özellikle hassas verilere sahip olduklarında, bu verileri dış servislerle paylaşmak istemezler. Açık kaynak modeller, yerel kurulum imkânı sunduğu için bu ihtiyaca doğrudan cevap verir.

Bir diğer önemli motivasyon şeffaflık ve incelenebilirliktir. Açık kaynak modellerin kodları, ağırlıkları ve eğitim verilerine dair bilgiler herkesin erişimine açık olduğu için, modelin nasıl çalıştığı ya da ne tür önyargılar barındırabileceği analiz edilebilir. Bu, güvenlik ve sorumluluk açısından büyük avantaj sağlar.

Ayrıca açık kaynak modeller, görev uyumlu özelleştirmeye olanak tanır. Belirli alanlara (örneğin sağlık, hukuk, eğitim) özel verilerle yeniden eğitilerek model çıktıları bu alanlara özgü ihtiyaçlara uyarlanabilir.

Son olarak, maliyet kontrolü de açık kaynak modellere yönelimin bir diğer sebebidir. API bazlı hizmetlerde her çağrı ücretlidir; oysa açık kaynak modeller, kullanıcıya ait altyapı üzerinde çalıştırılarak bu tür maliyetlerin önüne geçebilir.

Bu bağlamda, günümüzde öne çıkan bazı açık kaynak modeller şunlardır:

Mistral 7B ve Mixtral 8x7B, Mistral AI tarafından geliştirilmiştir. Sparsely-gated Mixture-of-Experts (MoE) mimarisi sayesinde hem düşük gecikme hem de yüksek kalite sunar.

LLaMA 3 (8B ve 70B varyantları), Meta AI tarafından geliştirilmiştir ve özellikle bağlam

anlama konusunda güçlüdür. GPT-3.5 seviyesinde performans sunduğu kabul edilir.

Phi-2, Microsoft Research tarafından geliştirilmiş olup düşük parametre sayısına rağmen yüksek doğruluk sağlayan, özenle seçilmiş eğitim verileriyle eğitilmiş bir modeldir.

Gemma, Google tarafından geliştirilmiş olup mobil ve edge cihazlar için optimize edilmiş bir model serisidir.

Zephyr ve **OpenChat**, açık kaynak topluluğu tarafından geliştirilen ve genellikle sohbet formatına uygun olarak eğitilmiş, RLHF (insan geri bildiriminiyle pekiştirme) uygulanmış modellerdir.

Bu modeller genellikle Hugging Face, Together.ai gibi açık platformlar üzerinden ücretsiz ve şeffaf şekilde erişilebilir durumdadır.

6.5.3. Ticari ve Açık Kaynak LLM'lerin Karşılaştırılması

Kapalı kaynak büyük dil modelleri (örneğin GPT-4 ve Claude 3), genellikle yüksek performans sunar, özellikle mantıksal çıkarım (reasoning) gerektiren görevlerde üstünlük gösterir. Ancak bu modellerin kullanımı token başına API ücretiyle ücretlendirilir. Özelleştirme açısından çoğu zaman sınırlı imkanlar sunar; modelin görev odaklı uyarlanması genellikle desteklenmez. Ayrıca veri denetimi açısından da kısıtlıdır; çünkü kullanıcı verileri model sağlayıcısının sunucularına gider. Şeffaflık konusunda da kapalıdır; model ağırlıkları ve eğitim verisi kamuya açık değildir.

Buna karşın açık kaynak büyük dil modelleri (örneğin LLaMA 3, Mistral ve Phi-2) birçok görevde benzer kalitede sonuçlar verebilir. Özellikle bazı görevlerde kapalı modellere yakın ya da eşdeğer performans gösterdikleri gözlemlenmiştir. Bu modellerin en büyük avantajı kullanım maliyetinin çok düşük olmasıdır; API kullanım ücreti yerine yalnızca çalıştırıldıkları sunucuların maliyeti söz konusudur. Açık kaynaklı yapıları sayesinde tamamen özelleştirilebilirler; farklı görevlere veya veri türlerine göre eğitilip uyarlanabilirler. Ayrıca bu modeller kullanıcı sisteminde çalıştığı için veriler tamamen yerelde kalır ve dışarı çıkmaz. Model mimarisi, ağırlıkları ve eğitim yöntemleri kamuya açık olduğu için de tam anlamıyla şeffaftırlar.

Açık kaynak LLM'lerin akademik ve endüstriyel uygulama senaryoları oldukça geniştir. Araştırmalarda, yeni eğitim yöntemlerinin denenmesi, güvenlik testleri yapılması veya farklı modellerin karşılaştırılması için temel oluştururlar. Kurumsal sistemlerde, özel veri

üzerinde çalışan sohbet robotları veya belge özetleme çözümleri bu modellerle geliştirilebilir. Ayrıca, veri gizliliğinin kritik olduğu sağlık, finans ve hukuk gibi regülasyona tabi alanlarda açık kaynak modeller yerel kurulum avantajı sayesinde tercih edilir. Düşük kaynak tüketimli modeller (örneğin Phi-2 ve Gemma), mobil ya da uç cihazlar üzerinde çalıştırılarak cihaz içi akıllı etkileşimlerin sağlanmasına olanak tanır.

7. Bulgular ve Tartışma

Bu çalışmada gerçekleştirilen uygulamalar, büyük dil modellerinin (LLM) yönergeye dayalı kullanımında hem kalite hem de maliyet açısından nasıl optimize edilebileceğini deneysel olarak ortaya koymuştur. Üç temel uygulama senaryosu üzerinden yapılan gözlemler, prompt mühendisliği ve model seçim stratejilerinin çıktılar üzerinde doğrudan etkili olduğunu göstermektedir.

İlk uygulama olan tek oyunculu **Dungeons & Dragons (D&D) rol yapma deneyimi**, büyük dil modelinin (ChatGPT) belirli anlatı kuralları ve oyun içi davranış talimatları altında nasıl yönlendirilebileceğini test etmiştir. Bulgular, yapay zekâ modelinin senaryo bütünlüğünü koruyarak doğaçlama diyaloglar ve olay örgüsü oluşturabildiğini göstermiştir. Özellikle, oyuncunun etik ikilemlerine karşı verilen NPC tepkileri, modelin belirlenen prompt talimatlarını başarıyla yorumladığını ortaya koymuştur. Ancak bu başarının temelinde, iyi yapılandırılmış ve detaylı bir prompt tasarımı olduğu anlaşılmıştır. Bu da prompt mühendisliğinin yalnızca maliyet değil, deneyim kalitesi açısından da kritik olduğunu kanıtlamaktadır.

İkinci uygulama olan **Fonksiyonel Gereksinim Raporu oluşturma senaryosu**, ChatGPT'nin yazılım analiz süreçlerinde kullanılabilirliğini test etmiştir. Yinelemeli prompt tasarımı ile geliştirilen nihai prompt, modelin hem teknik hem de yapısal olarak uygun çıktılar üretmesini sağlamıştır. Bu deneyim, LLM'lerin mühendislik süreçlerinde belgeler oluşturma, yorumlama ve özetleme gibi görevlerde etkili şekilde kullanılabileceğini, ancak bu etkinliğin yönlendirme kalitesiyle sınırlı olduğunu göstermektedir.

Üçüncü uygulama ise **Retrieval-Augmented Generation (RAG)** tabanlı soru-cevap sisteminin üniversite yönetmeliklerinden bilgi çekmede nasıl kullanılabileceğini göstermiştir. Supabase tabanlı bir vektör veritabanı, açık kaynak embedding modeli ve hafif bir LLM kullanılarak kurulan sistem, düşük kaynak tüketimiyle başarılı sonuçlar

vermiştir. Öğrencilerden gelen doğal dildeki sorulara doğru, kaynaklı ve açıklayıcı cevaplar üretilebilmiştir. Bu uygulama, maliyet etkinliğinin sadece model seçiminden değil, mimari optimizasyondan da geçtiğini göstermiştir.

Bu üç uygulama birlikte değerlendirildiğinde, LLM kullanımında başarıya ulaşmak için yalnızca güçlü bir model seçmenin yeterli olmadığı; aynı zamanda prompt kalitesi, kullanım senaryosuna uygun mimari tercihler ve maliyet-fayda dengesinin dikkate alınması gerektiği görülmüştür.

8. Sonuçlar

Bu proje çalışmasında, büyük dil modellerinin (LLM) yönergeye dayalı kullanım biçimleri ve optimizasyon yöntemleri farklı uygulama senaryoları ile detaylı biçimde analiz edilmiştir. Gerçekleştirilen deneysel uygulamalar, doğru prompt mühendisliği ve uygun model-mimari seçiminin, yüksek kaliteli çıktılar ile maliyet verimliliği arasında nasıl bir denge kurabileceğini açıkça ortaya koymuştur.

ChatGPT ile yürütülen **etkileşimli rol yapma senaryosu**, yapay zekânın yönlendirme doğrultusunda yaratıcı ve kurallı içerik üretebileceğini gösterirken; otomatik raporlama uygulaması, LLM'lerin yazılım geliştirme süreçlerinde nasıl aktif rol alabileceğini kanıtlamıştır. **RAG tabanlı belge soru-cevap sistemi** ise sınırlı kaynaklarla çalışan bir yapının bile yüksek doğrulukta bilgi erişimi sağlayabileceğini göstermiştir.

Bu çalışmalar, büyük dil modellerinin sadece bir araç değil, aynı zamanda doğru stratejilerle kullanıldığında yüksek değerli çözümler sunan bir altyapı haline geldiğini ortaya koymaktadır. Model başarısı yalnızca teknik üstünlükle değil, kullanım senaryosuna özel optimizasyonlarla ölçülmelidir.

Gelecek çalışmalarda, bu yapıların daha geniş kullanıcı kitlesine açılması, kullanıcı dostu arayüzlerle desteklenmesi ve daha büyük veri havuzlarına entegre edilerek ölçeklendirilmesi önerilmektedir. Ayrıca hibrit yapılar, model rotasyonu, önbellekleme ve hafif modellerin mobil entegrasyonu gibi ileri optimizasyon yöntemleri, üretim ortamlarında verimliliği daha da artıracaktır.

•

KAYNAKLAR

- [1] T. B. Brown et al., “Language Models are Few-Shot Learners,” *arXiv preprint arXiv:2005.14165*, 2020.
- [2] OpenAI, “GPT-4 Technical Report,” *arXiv preprint arXiv:2303.08774*, 2023.
- [3] J. Dean et al., “Scaling Deep Learning Models,” *arXiv preprint arXiv:2203.15556*, 2022.
- [4] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed., Pearson, 2021.
- [5] M. T. Ribeiro et al., “Why Should I Trust You?” Explaining the Predictions of Any Classifier, *ACM SIGKDD*, 2016.
- [6] N. Bostrom, *Superintelligence: Paths, Dangers, Strategies*, Oxford University Press, 2014.
- [7] Jurafsky and J. H. Martin, *Speech and Language Processing*, 3rd ed., Pearson, 2023.
- [8] A. Vaswani et al., “Attention is All You Need,” *arXiv preprint arXiv:1706.03762*, 2017.
- [9] T. Mikolov et al., “Efficient Estimation of Word Representations in Vector Space,” *arXiv:1301.3781*, 2013.
- [10] R. Bommasani et al., “On the Opportunities and Risks of Foundation Models,” *arXiv:2108.07258*, 2021.
- [11] Y. Zhang et al., “A Survey of Large Language Models,” *arXiv:2303.18223*, 2023.
- [12] Y. Bai et al., “Constitutional AI: Harmlessness from AI Feedback,” *arXiv:2212.08073*, 2023.
- [13] OpenAI, “Pricing,” [Online]. Available: <https://openai.com/pricing>

- [14] Y. Jietal., “Survey of Hallucination in Natural Language Generation,” *arXiv:2302.03494*, 2023.
- [15] N. Shazeer et al., “Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer,” *arXiv:1701.06538*, 2017.
- [16] W. Fedus et al., “Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity,” *arXiv:2101.03961*, 2021.
- [17] S. Min et al., “Recent Advances in Prompting for Natural Language Processing,” *arXiv:2107.13586*, 2021.
- [18] J. Devlin et al., “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” *arXiv:1810.04805*, 2018.
- [19] Ouyang, L. et al., “Training language models to follow instructions with human feedback,” *arXiv:2203.02155*, 2022.
- [20] J. Schulman et al., “Proximal Policy Optimization Algorithms,” *arXiv:1707.06347*, 2017.
- [21] G. Hinton et al., “Distilling the Knowledge in a Neural Network,” *arXiv:1503.02531*, 2015.
- [22] J. Wei et al., “Chain of Thought Prompting Elicits Reasoning in Large Language Models,” *arXiv:2201.11903*, 2022.
- [23] S. Liu et al., “Pretrain Prompting for Large Language Models: A Survey,” *arXiv:2302.01765*, 2023.
- [24] Y. Bai et al., “Training a Helpful and Harmless Assistant with RLHF,” *arXiv:2204.05862*, 2022.
- [25] H. Holtzman et al., “The Curious Case of Neural Text Degeneration,” *arXiv:1904.09751*, 2019.
- [26] A. Radford et al., “Language Models are Unsupervised Multitask Learners,” *OpenAI Technical Report*, 2019.
- [27] OpenAI, “Best practices for prompt design,” 2023. [Online]. Available: <https://platform.openai.com/docs/guides>

- [28] S. Tay et al., “Efficient Language Modeling with Sparse Transformers,” *arXiv:2004.05150*, 2020.
- [29] Google AI, “DistilBERT: A distilled version of BERT,” [Online]. Available: <https://ai.googleblog.com>
- [30] S. Wang et al., “Accelerating Large Language Model Inference via Caching and Prefetching,” *arXiv:2302.09861*, 2023.
- [31] P. Lewis et al., “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks,” *arXiv:2005.11401*, 2020.
- [32] OpenAI Docs, “Optimizing LLM usage with caching strategies,” 2023. [Online]. Available: <https://platform.openai.com/docs/guides>
- [33] S. Jiao et al., “TinyBERT: Distilling BERT for Natural Language Understanding,” *arXiv:1909.10351*, 2019.
- [34] Z. Lan et al., “ALBERT: A Lite BERT for Self-supervised Learning of Language Representations,” *arXiv:1909.11942*, 2019.
- [35] H. Sun et al., “MobileBERT: a Compact Task-Agnostic BERT for Resource-Limited Devices,” *arXiv:2004.02984*, 2020.
- [36] Anthropic, “Claude 3 Family,” 2024. [Online]. Available: <https://www.anthropic.com>
- [37] Meta AI, “LLaMA 2: Open Foundation and Fine-Tuned Chat Models,” 2023.
- [38] Hugging Face, “DistilBERT Model Card,” [Online]. Available: <https://huggingface.co>
- [39] Christiano, P. et al., “Deep reinforcement learning from human preferences,” *arXiv:1706.03741*, 2017.
- [40] Anthropic Blog, “Introducing Constitutional AI,” 2023. [Online]. Available: <https://www.anthropic.com>
- [41] S. Gupta et al., “A Comprehensive Survey of Retrieval-Augmented Generation,” *arXiv:2301.12652*, 2023.
- [42] B. Allen et al., “Towards Continual Learning in Language Models,” *arXiv:2112.09147*, 2021.

- [43] Ji, Z. et al., “Survey of Hallucination in Natural Language Generation,” *arXiv:2302.03494*, 2023.
- [44] Meta AI, “LLaMA 3 Technical Overview,” 2024. [Online]. Available: <https://ai.meta.com/llama>
- [45] Mistral AI, “Introducing Mixtral,” 2023. [Online]. Available: <https://mistral.ai>
- [46] Microsoft Research, “Phi-2: Language Models with Better Data, Not Bigger Size,” *arXiv:2312.11561*, 2023.
- [47] Google AI Blog, “Gemma: Open Models Built for Responsible AI,” 2024.
- [48] Hugging Face, “Open LLM Leaderboard.” [Online]. Available: https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard