



**T.C.
BATMAN ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ
BİLGİ TEKNOLOJİLERİ ANA BİLİM DALI**

YÜKSEK LİSANS TEZİ

**BDM TABANLI AKILLI KÜTÜPHANE TARAMA VE DİYALOG
SİSTEMİ**

Suat GÖK

**Temmuz-2025
BATMAN**

**T.C.
BATMAN ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ
BİLGİ TEKNOLOJİLERİ ANA BİLİM DALI**

YÜKSEK LİSANS TEZİ

**BDM TABANLI AKILLI KÜTÜPHANE TARAMA VE DİYALOG
SİSTEMİ**

Suat GÖK

**Danışman
Dr. Öğr. Üyesi Hafzullah İŞ**

Diğer Jüri Yeleri

Dr. Öğr. Üyesi İlhan Fırat KILINÇER Dr. Öğr. Üyesi Rıdvan Fırat ÇINAR

**Temmuz-2025
BATMAN**

TEZ KABUL VE ONAYI

Suat GÖK tarafından hazırlanan “BDM TABANLI AKILLI KÜTÜPHANE TARAMA VE DİYALOG SİSTEMİ” adlı tez çalışması 17/07/2025 tarihinde aşağıdaki jüri tarafından oy birliği ile Batman Üniversitesi Lisansüstü Eğitim Enstitüsü Bilgi Teknolojileri Ana Bilim Dalı’nda YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

Jüri Üyeleri

İmza

Başkan

Dr. Öğr. Üyesi İlhan Fırat KILINÇER

.....

Danışman

Dr. Öğr. Üyesi Hafzullah İŞ

.....

Üye

Dr. Öğr. Üyesi Rıdvan Fırat ÇINAR

.....

Yukarıdaki sonucu onaylarım.

Dr. Öğr. Üyesi Ömer Murat ÖTER
Lisansüstü Eğitim Enstitüsü Müdürü

Bu tez çalışması BAP tarafından BTÜBAP-2025-YL-03 nolu proje ile desteklenmiştir.

ETİK BEYANI

Bu tezdeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edildiğini ve tez yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını beyan eder, aksinin ortaya çıkması durumunda her türlü yasal sorumluluğu kabullendiğimi bildiririm.

ETHICAL DECLARATION

I declare that all the information in this thesis has been obtained within the framework of ethical behavior and academic rules, and that the source of any statements and information that do not belong to me in this study prepared in accordance with the thesis writing rules has been fully cited, and I declare that I accept all kinds of legal responsibility in case of any contrary situation.

Suat GÖK
Tarih:17/07/2025

ÖZET

YÜKSEK LİSANS TEZİ

BDM TABANLI AKILLI KÜTÜPHANE TARAMA VE DİYALOG SİSTEMİ

Suat GÖK

Batman Üniversitesi Lisansüstü Eğitim Enstitüsü

Bilgi Teknolojileri Ana Bilim Dalı

Danışman: Dr. Öğr. Üyesi Hafzullah İŞ

2025, 189 Sayfa

Bu çalışma, Batman Üniversitesi için özel olarak geliştirilen BDM (Büyük Dil Modeli) tabanlı yapay zekâ destekli akıllı kütüphane tarama ve diyalog sisteminin teknik altyapısı, mimarisi ve performans değerlendirmelerini kapsamlı şekilde ele almaktadır. Geleneksel OPAC (Online Public Access Catalog) sistemlerinin kısıtlılıkları ve kullanıcı beklentilerindeki artış göz önünde bulundurularak, literatürde yer alan çağdaş bilgi erişim teknikleri incelenmiş; özellikle vektör tabanlı veri yapıları, Retrieval-Augmented Generation (RAG) mimarisi, kullanıcı geri bildirimleri ile beslenen model iyileştirmeleri (Reinforcement Learning with Human Feedback - RLHF), çok modelli yapılar (ensemble learning), tarihsel tutarsızlık tespiti ve Explainable AI (XAI) yöntemleri detaylandırılmıştır. Sistem, Python programlama dili ve LangChain framework'ü kullanılarak çoklu kaynaklardan bilgi sorgulama kapasitesine sahip şekilde tasarlanmıştır. Bilgi kaynakları arasında öncelikle Batman Üniversitesi'nin YORDAM katalogu, ardından üniversitenin e-Arşiv platformu ve son olarak ChatGPT destekli web arama modülü bulunmaktadır. Bu çok aşamalı sorgu zinciri, kullanıcının arama deneyimini optimize etmekte ve daha kapsamlı, doğru ve güncel bilgiye erişim sağlamaktadır. Chroma tabanlı vektör veritabanı ise yüksek verimli bellek yönetimi ve anlamsal arama fonksiyonları sunarak sistemin performansını artırmaktadır. Ayrıca, sistemde kullanıcıların geri bildirimleri toplanarak RLHF yaklaşımı ile modellerin sürekli güncellenmesi ve öğrenmesi sağlanmaktadır. Bu sayede sistem, hem teknik hem de kullanıcı deneyimi açılarından dinamik ve adaptif bir yapıya kavuşmuştur. Tarihsel tutarsızlık tespiti gibi ileri analiz modülleri, bilgi güvenilirliğini artırırken, Explainable AI teknikleri ile kullanıcılar ve kütüphane personeli için sistem çıktılarının şeffaf ve anlaşılır olması hedeflenmiştir.

Araştırma bulguları, geliştirilen sistemin bilgi erişim hızını anlamlı düzeyde artırdığını, arama sonuçlarının doğruluğunu iyileştirdiğini ve kullanıcı memnuniyetini yükselttiğini ortaya koymaktadır. Türkiye'deki üniversite kütüphaneleri için örnek teşkil edebilecek bu entegre ve sürdürülebilir yapay zekâ destekli kütüphane sistemi, akademik araştırma süreçlerinin etkinliğini artırmaya yönelik önemli bir katkı sağlamaktadır.

Anahtar Kelimeler: Kütüphane Otomasyonu, Bilgi Dağıtım Modeli (BDM), Retrieval-Augmented Generation (RAG), Reinforcement Learning with Human Feedback (RLHF), ChatGPT.

ABSTRACT
MASTER THESIS
BDM BASED SMART LIBRARY BROWSING AND DIALOGUE SYSTEM

Suat GÖK

Batman University Graduate Education Institute

Department of Information Technologies

Advisor: Assistant Professor Dr. Hafzullah İŞ

2025, 189 Pages

This study comprehensively examines the technical infrastructure, architectural design, and performance evaluation of an AI-supported intelligent library search and dialogue system, specifically developed for Batman University based on a Large Language Model (LLM). Considering the limitations of traditional OPAC (Online Public Access Catalog) systems and the growing expectations of users, contemporary information retrieval techniques in the literature have been thoroughly analyzed. In particular, vector-based data structures, Retrieval-Augmented Generation (RAG) architecture, model enhancements through Reinforcement Learning with Human Feedback (RLHF), ensemble learning strategies, historical inconsistency detection, and Explainable AI (XAI) methodologies are elaborated in detail. The system is implemented using the Python programming language and the LangChain framework to facilitate multi-source information retrieval. The primary data sources include Batman University's YORDAM catalog, followed by its institutional repository (e-Archive), and lastly, a web search module supported by ChatGPT. This multi-stage query pipeline is designed to optimize the information retrieval process and provide access to comprehensive, accurate, and up-to-date data. The Chroma-based vector database further improves system performance by enabling efficient memory management and semantic search capabilities. Moreover, user feedback is continuously collected and utilized within the RLHF framework to support adaptive learning and ongoing model refinement. Consequently, the system demonstrates a dynamic and evolving structure in terms of both technical functionality and user interaction. Advanced analytical components, such as historical inconsistency detection, contribute to information reliability, while Explainable AI techniques aim to enhance transparency and interpretability of system outputs for end-users and library personnel.

Findings from this research indicate that the developed system significantly improves information retrieval speed, enhances the precision of search results, and increases overall user satisfaction. As an integrated and sustainable AI-based library solution, the system represents a promising model for academic libraries in Turkey, offering valuable contributions to the efficiency and effectiveness of scholarly research processes.

Keywords: Library Automation, BDM (Information Distribution Model), Retrieval-Augmented Generation (RAG), Reinforcement Learning with Human Feedback (RLHF), ChatGPT.

ÖN SÖZ

Bilgi çağının dinamik yapısı, yalnızca verinin hacmini değil, bilgiye erişim ve işleme biçimimizi de köklü biçimde dönüştürmektedir. Bu dönüşüm, özellikle akademik bilginin üretildiği, saklandığı ve erişildiği kütüphane sistemlerinde daha belirgin şekilde hissedilmektedir. Günümüz kullanıcılarının artan beklentileri ve bilgiye hızlı ulaşma ihtiyacı, geleneksel sistemlerin ötesine geçen, yapay zekâ destekli çözümlere olan gereksinimi ortaya koymaktadır. Bu bağlamda geliştirilen “BDM Tabanlı Akıllı Kütüphane Tarama ve Diyalog Sistemi”, yalnızca teknolojik bir araç değil; bilgi yönetimi, yapay zekâ entegrasyonu ve kullanıcı odaklılık ilkeleri doğrultusunda sürdürülebilir bir yaklaşım sunmayı amaçlamaktadır. Aynı zamanda, bu sistem bilgi toplumu inşasında katkı sağlayabilecek, çok katmanlı ve genişletilebilir bir model önerisi niteliği taşımaktadır. Bu çalışmanın her aşamasında akademik rehberliği, teşvik edici tutumu ve yönlendirici katkılarıyla bana ilham veren, sürecin başından sonuna dek desteğini esirgemeyen değerli danışmanım Dr. Öğr. Üyesi Hafzullah İŞ’e en derin teşekkürlerimi ve saygılarımı sunuyorum. Kendisinin katkıları, bu çalışmanın bilimsel bütünlüğünü şekillendirmede belirleyici olmuştur. Ayrıca, akademik düşünce yapımın gelişiminde her zaman yanımda olan ve bu süreçte bir ikinci danışman gibi desteğini hissettiren Doç. Dr. Yılmaz KAYA’ya içten şükranlarımı sunarım. Tez boyunca sıklıkla başvurduğum ve bilgi birikiminden büyük ölçüde faydalandığım Nihat DAMAR’a da ayrıca teşekkür ederim. Bu yolculukta manevi desteğini hep hissettiğim sevgili eşim Neslihan’a ve varlığıyla bana ilham veren sevgili oğlum Hasan Göktürk Manas’a teşekkür etmeyi bir borç bilirim. Aynı zamanda, bana her zaman inanan ve destekleyen kıymetli aileme de sonsuz minnettarım. Bu çalışmanın, kütüphane sistemleri, bilgi arama modelleri, kullanıcı davranışları ve yapay zekâ temelli bilgiye erişim uygulamaları alanında yapılacak araştırmalara katkı sağlamasını ve akademik çevreler için faydalı bir kaynak olmasını diliyorum.

Suat GÖK
BATMAN-2025

İÇİNDEKİLER

ÖZET	iv
ABSTRACT	v
ÖN SÖZ	vi
İÇİNDEKİLER	vii
ŞEKİLLER LİSTESİ	xi
SİMGELER VE KISALTMALAR	xvi
1. GİRİŞ	1
1.1. Araştırmanın Amacı	3
1.2. Problem Durumu	3
1.3. Araştırmanın Önemi	4
1.4. Varsayımlar	4
1.5. Sınırlılıklar	5
1.6. Tanımlar	5
2. KAYNAK ARAŞTIRMASI	6
2.1. Büyük Dil Modellerinin (BDM) Tanımı ve Tarihsel Gelişimi	6
2.1.1. Transformer mimarisi ve dil modelleme	7
2.1.2. BDM'lerin NLP'de başlıca kullanım alanları	8
2.1.3. Türkçe BDM modelleri ve uygulamaları	9
2.1.4. BDM'lerin avantajları ve sınırlılıkları	9
2.1.5. Akıllı sistemlerde BDM kullanımı: Chatbot ve Arama	10
2.1.6. BDM'lerin kütüphane sistemlerine entegrasyonu	11
2.2. Bilgi Erişim ve Kütüphane Sistemleri	12
2.2.1. Geleneksel OPAC (Online Public Access Catalog) mimarisine genel bakış	13
2.2.2. FRBR, RDA ve kütüphane sınıflandırma standartları	14
2.2.3. Dijital kütüphane çerçeveleri ve çok kaynaklı entegrasyon	15
2.3. Bilgi Arama Modelleri	16
2.3.1. Boolean ve vektör uzayı modelleri	17
2.3.2. Olasılıksal retrieval modelleri (BM25, Language Modeling)	18
2.3.3. Derin öğrenme tabanlı RETRIEVAL yaklaşımları (DPR, ANCE)	19
2.4. Retrieval-Augmented Generation (RAG) Yaklaşımları	20
2.4.1. RAG temelleri	20
2.4.2. Çok aşamalı RAG pipeline'ları ve borçlanma stratejileri	21
2.4.3. Langchain ve alternatif framework'ler	22
2.5. Vektör Tabanlı Veri Depoları	22
2.5.1. FAISS, ANNOY, HNSW ve CHROMA karşılaştırması	23
2.5.2. Ölçeklenebilirlik ve performans optimizasyonları	24
2.6. Kullanıcı Geri Bildirimi ve Öğrenme	25
2.6.1. Relevance feedback: rocchio ve pseudo-relevance feedback	26
2.6.2. İnsan geri bildirimi ile öğrenme (RLHF) ve REWARD modelleri	26
2.6.3. Online öğrenme sistemleri ve adaptif döngüler	27
2.7. Sorgu Anlama ve Sınıflandırma	28

2.7.1. Intent detection ve sorgu türleri (Tanımsal, Geçmişsel, Güncel)	29
2.7.2. Derin öğrenme ile sorgu embedding	30
2.8. Zaman Bazlı Tutarlılık ve Filtreleme	30
2.8.1. Temporal information retrieval kavramları	31
2.8.2. Tarihsel tutarsızlık tespiti yöntemleri	32
2.9. Çok Modelli Yaklaşımlar ve Ensembles	33
2.9.1. Model birleştirme teknikleri (VOTİNG, STACKİNG)	34
2.9.2. Karar ağaçları ile model seçimi	34
2.10. Sohbet Robotlarının Kütüphane ve Akademik Bağlamdaki Kullanımları	35
2.10.1. Case Study: LibChat, AskONETEAM	36
2.10.2. OpenAI ChatGPT ve akademik yardımcı entegrasyonları	36
2.11. Değerlendirme Metrikleri ve Kullanıcı Deneyimi	38
2.11.1. Otomatik metrikler: BLEU, ROUGE, METEOR vb.	38
2.11.2. İnsan değerlendirme ve anket temelli yaklaşımlar	39
2.11.3. Kullanıcı memnuniyeti ve etkili UI ölçütleri	40
2.12. Performans, Ölçeklenebilirlik ve Maliyet Optimizasyonu	41
2.12.1. Asenkron pipeline'lar ve kuyruk yönetimi	42
2.12.2. Caching stratejileri (LLM, Vektör)	42
2.12.3. Token maliyeti vs. gecikme dengesi	43
2.13. Explainable AI (XAI) ve Şeffaflık	44
2.13.1. Model karar süreçlerinin açıklanması	45
2.13.2. Kullanıcıya sunulan şeffaflık katmanları	45
2.13.3. Audit trail ve sürüm kontrolü	46
3. YÖNTEM	48
3.1. Araştırma Tasarımı	48
3.2. Kullanılan Teknolojiler ve Bileşenler	48
3.3. Sistem Akışı	49
3.4. Veri Toplama ve Değerlendirme	50
4. BULGULAR VE TARTIŞMA	51
4.1. Ana Sayfa Arayüzü İncelemesi	51
4.2. Oturum Açıldıktan Sonra Arayüzün Değişimi	52
4.3. Kullanıcı Girişi Sayfası: Güvenli ve Basit Erişim Mekanizması	52
4.4. Kayıt Sayfası: Yeni Kullanıcı Ekleme Arayüzü	53
4.5. Model Seçimi Sayfası: Kişiselleştirilmiş Yapay Zeka Deneyimi	54
4.6. Favorilerim Sayfası ve Kişiselleştirme Özelliği	55
4.7. Duyurular Sayfası Arayüzü ve İçeriği	56
4.8. Hizmetlerimiz Sayfası: Kapsamlı Kullanıcı Destekleri	56
4.9. Hakkımızda Sayfası: Kurumsal Bilgi Sunumu	58
4.10. İletişim Sayfası: Erişilebilirlik ve Şeffaflık	59
4.11. Veri Tabanları	60
4.12. Yordam, E-Arşiv, ChatGPT Sorgu Sonuçları	64
4.13. Kullanıcı Paneli İşlemleri	69
4.14. Admin Dashboard Kullanıcı İşlemleri	74
4.15. Admin Dashboard Analiz ve Raporlama	77
4.16. Model Eğitimi ve Performans Göstergeleri	83
TARTIŞMA	103

5. SONUÇLAR VE ÖNERİLER.....	105
5.1 Sonuçlar	105
5.2 Öneriler	106
KAYNAKLAR.....	107
EKLER.....	113
EK-1 Temel Yapılandırma & Yardımcılar	113
A1 settings.py kodları	113
A2 utils.py kodları	113
A3 login_manager.py kodları	114
EK-2 Veri tabanı & Veri Betikleri	115
B1 veritabani_olustur.py kodları	115
B2 veritabani_migrosyon.py kodları	117
B3 sorgu_gecmisi_tablosu_olustur.py kodları	118
EK-3 Çekirdek Servis Katmanı	118
C1 router.py kodları	119
C2 yordam_api.py kodları	122
C3 earsiv.py kodları	123
C4 search.py kodları	125
C5 ai.py kodları	126
C6 model_predictor.py kodları	126
C7 hallucination_grader.py kodları	128
C8 reward_scorer.py kodları	128
C9 tutarsizlik_tespit.py kodları	129
C10 sotgu_tipi_tespit.py kodları	130
EK-4 Kullanıcı Arayüzü (Streamlit)	130
D1 streamlit_app.py kodları	131
D2 streamlit_feedback.py kodları	136
D3 streamlit_feedback_interface.py kodları	137
D4 pages/login.py kodları	138
D5 pages/logout.py kodları	139
D6 pages/register.py kodları	140
D7 pages/model_sec.py kodları	141
D8 pages/user_dashboard.py kodları	141
D9 pages/favorilerim.py kodları	144
D10 pages/veritabanlari.py kodları	146
D11 pages/hizmetlerimiz.py kodları	147
D12 pages/hakkimizda.py kodları	149
D13 pages/duyurular.py kodları	150
D14 pages/iletisim.py kodları	152
EK-5 Yönetici Modülü	153
E1 pages/admin_dashboard.py kodları	154
E2 admin_sections/kullanici_yonetimi.py kodları	156
E3 admin_sections/kullanici_inceleme.py kodları	158
E4 admin_sections/geri_bildirim.py kodları	160
E5 admin_sections/favori_analiz.py kodları	161
E6 admin_sections/model_karsilastirma.py kodları	162
E7 admin_sections/egitim_loglari.py kodları	167

E8 admin_sections/sorgu_suresi_analizi.py kodları	170
E9 admin_sections/rlhf.py kodları	171
E10 admin_sections/model_egitme.py kodları	172
E11 admin_sections/veri_aktar.py kodları	173
E12 model_karsilastirma_kayit.py kodları	176
EK-6 Model Eğitimi & Otomasyon	177
F1 chain_fine_tune.py kodları	178
F2 update_model.py kodları	184
F3 offline_rlhf.py kodları	185
F4 setup_training_logs.py kodları	187
F5 indir_modeller.py kodları	188

ŞEKİLLER LİSTESİ

Şekil 1.1. Yapay zekâ destekli kütüphane otomasyon sisteminin genel yapısı	2
Şekil 1.2. Geleneksel OPAC ile yapay zekâ tabanlı akıllı kütüphane sisteminin karşılaştırması	3
Şekil 1.3. BDM ve RLHF destekli sistemin bilgi erişimi ve akademik başarıya katkısı	4
Şekil 2.1. Büyük dil modellerinin tarihsel evrimi	6
Şekil 2.2. Transformer modelinin yapısı	7
Şekil 2.3. Büyük dil modellerinin doğal dil işleme alanındaki başlıca kullanım alanları	8
Şekil 2.4. Türkçe dil modelleri ve kullanım alanları	9
Şekil 2.5. Büyük dil modellerinin (BDM) avantajları ve sınırlılıkları	10
Şekil 2.6. Akıllı sistemlerde BDM kullanımı: GPT tabanlı Chatbot ve akıllı arama	11
Şekil 2.7. BDM'lerin kütüphane sistemlerine entegrasyon alanları	12
Şekil 2.8. Entegre ve dijital odaklı bilgi erişim sistemleri	13
Şekil 2.9. Geleneksel OPAC sistemlerinin özellikleri	14
Şekil 2.10. FRBR ve RDA'nın dijital sınıflandırmadaki rolü	15
Şekil 2.11. Batman Üniversitesi dijital kütüphane entegrasyon yapısı	16
Şekil 2.12. Bilgi erişim sistemlerinde kullanılan bilgi arama modellerinin evrimi	17
Şekil 2.13. Boolean ve vektör uzayı modellerinin karşılaştırılması	17
Şekil 2.14. Olasılıksal bilgi erişim modelleri: BM25 ve Language Modeling	18
Şekil 2.15. Derin öğrenme destekli belge-sorgu eşleştirme mimarileri	19
Şekil 2.16. Retrieval-augmented generation (RAG) yaklaşımının çalışma prensibi	20
Şekil 2.17. Retrieval-augmented generation yaklaşımının genel şeması	21
Şekil 2.18. Çok aşamalı RAG sistemlerinde bilgi getirme ve metin üretimi süreci	21
Şekil 2.19. Retriever-generator zincirinde framework kullanımı	22
Şekil 2.20. Vektör tabanlı arama altyapısı	23
Şekil 2.21. FAISS, Annoy, HNSW ve Chroma vektör arama kütüphanelerinin karşılaştırması	24
Şekil 2.22. Vektör veri tabanlarında ölçeklenebilir yapı	25
Şekil 2.23. Yapay zekâ sistemlerinde kullanıcı geri bildirimi ve öğrenme süreçleri	25
Şekil 2.24. Rocchio ve pseudo-relevance feedback	26
Şekil 2.25. RLHF süreci ve reward model kullanımı	27
Şekil 2.26. Çevrim içi öğrenme ve adaptif geri bildirim döngüsü	28
Şekil 2.27. Sorgu anlama ve sınıflandırma süreci	29
Şekil 2.28. Sorgu türlerine göre intent detection yapısı	29
Şekil 2.29. Sorgu embedding ile öneri sistemi yapısı	30
Şekil 2.30. Zaman bazlı tutarsızlıkların görselleştirilmesi	31
Şekil 2.31. Zamansal filtreleme ile güncel bilgi sunumu	32
Şekil 2.32. Tarihsel tutarsızlıkların algılanmasında yıl farkı temelli yaklaşım	33

Şekil 2.33. Çok modelli yaklaşımlar ve ensemble mimarisi	33
Şekil 2.34. Voting ve stacking tabanlı model birleştirme yöntemleri	34
Şekil 2.35. Karar ağaçları aracılığıyla yapay zekâ destekli model seçimi sürecinin akış şeması	35
Şekil 2.36. Kütüphane ve akademik alanlarda sohbet robotlarının kullanım alanları	36
Şekil 2.37. ChatGPT destekli kütüphane sistemi mimarisi	37
Şekil 2.38. Değerlendirme metrikleri ile kullanıcı deneyiminin entegrasyonu	38
Şekil 2.39. BLEU, ROUGE, METEOR: temel ölçüm metrikleri	39
Şekil 2.40. RLHF (İnsan Geri Bildirimi ile Pekiştirmeli Öğrenme) süreci	40
Şekil 2.41. Kullanıcı memnuniyeti ve etkileşim ölçütleri	41
Şekil 2.42. Yapay zekâda verimlilik bileşenleri	41
Şekil 2.43. Asenkron sorgu yönetimi	42
Şekil 2.44. LLM ve vektör sistemlerinde embedding ve yanıt ön bellekleme stratejileri	43
Şekil 2.45. Token maliyeti-gecikme dengesi	44
Şekil 2.46. Açıklanabilir yapay zekâ ve şeffaflık ilkeleri	44
Şekil 2.47. Model karar süreçlerinin açıklanması	45
Şekil 2.48. Kullanıcıya sunulan bilgi şeffaflığı	46
Şekil 2.49. Audit Trail ile sorgu ve yanıt kayıtlarının izlenmesi	47
Şekil 3.1. Araştırma tasarımı	48
Şekil 3.2. Teknik bileşenler ve kullanılan yazılım platformları	49
Şekil 3.3. Sistem akışı	50
Şekil 3.4. Veri toplama ve model değerlendirme süreci	50
Şekil 4.1. Oturum açılmamış ana sayfa arayüzü	51
Şekil 4.2. Admin oturumu açıldıktan sonraki ana sayfa arayüzü	52
Şekil 4.3. Kullanıcı girişi arayüzü	53
Şekil 4.4. Kayıt (Register) sayfası arayüzü	54
Şekil 4.5. Model seçimi arayüzü	55
Şekil 4.6. Favorilerim sayfası arayüzü	55
Şekil 4.7. Duyurular sayfası arayüzü	56
Şekil 4.8. Hizmetlerimiz sayfası arayüzü-1	57
Şekil 4.9. Hizmetlerimiz sayfası arayüzü-2	57
Şekil 4.10. Hakkımızda-1	58
Şekil 4.11. Hakkımızda-2	59
Şekil 4.12. İletişim sayfası arayüzü	60
Şekil 4.13. Veri tabanı-1	61
Şekil 4.14. Veri tabanı-2	61
Şekil 4.15. Veri tabanı-3	62
Şekil 4.16. Veri tabanı-4	62

Şekil 4.17. Veri tabanı-5	63
Şekil 4.18. Veri tabanı-6	63
Şekil 4.19. Veri tabanı-7	63
Şekil 4.20. Veri tabanı-8	64
Şekil 4.21. YORDAM sonuç-1	65
Şekil 4.22. YORDAM sonuç-2	65
Şekil 4.23. YORDAM sonuç-3	65
Şekil 4.24. YORDAM sonuç-4	66
Şekil 4.25. E-Arşiv-1	66
Şekil 4.26. E-Arşiv-2	66
Şekil 4.27. E-Arşiv-3	67
Şekil 4.28. E-Arşiv-4	67
Şekil 4.29. ChatGPT-1	68
Şekil 4.30. ChatGPT-2	68
Şekil 4.31. ChatGPT-3	69
Şekil 4.32. ChatGPT-4	69
Şekil 4.33. Kullanıcı paneli işlemleri-1	70
Şekil 4.34. Kullanıcı paneli işlemleri-2	71
Şekil 4.35. Kullanıcı paneli işlemleri-3	71
Şekil 4.36. Kullanıcı paneli işlemleri-4	72
Şekil 4.37. Kullanıcı paneli işlemleri-5	72
Şekil 4.38. Kullanıcı paneli işlemleri-6	73
Şekil 4.39. Kullanıcı paneli işlemleri-7	73
Şekil 4.40. Admin dashboard-1	74
Şekil 4.41. Admin dashboard-2	75
Şekil 4.42. Admin dashboard-3	75
Şekil 4.43. Admin dashboard-4	76
Şekil 4.44. Admin dashboard-5	76
Şekil 4.45. Admin dashboard-6	76
Şekil 4.46. Admin dashboard-7	77
Şekil 4.47. Admin dashboard-8	77
Şekil 4.48. Analiz ve raporlama-1	78
Şekil 4.49. Analiz ve raporlama-2	78
Şekil 4.50. Analiz ve raporlama-3	79
Şekil 4.51. Analiz ve raporlama-4	79
Şekil 4.52. Analiz ve raporlama-5	80
Şekil 4.53. Model yönetimi	80

Şekil 4.54. Sorgu süresi-1	81
Şekil 4.55. Sorgu süresi-2	81
Şekil 4.56. RLHF reward	82
Şekil 4.57. Veri aktarımı	82
Şekil 4.58. BERTurk metrikleri	83
Şekil 4.59. BERTurk eğitim son durum metrikleri-1	83
Şekil 4.60. BERTurk eğitim son durum metrikleri-2	84
Şekil 4.61. BERTurk Epoch bazlı metrikler-1	84
Şekil 4.62. BERTurk Epoch bazlı metrikler-2	85
Şekil 4.63. BERTurk zaman bazlı metrikler-1	85
Şekil 4.64. BERTurk zaman bazlı metrikler-2	85
Şekil 4.65. Epoch bazlı metrikler	86
Şekil 4.66. BRTurk tam log içeriği	86
Şekil 4.67. DistilBERT metrikleri	86
Şekil 4.68. DistilBERT eğitim son durum metrikleri-1	87
Şekil 4.69. DistilBERT eğitim son durum metrikleri-2	87
Şekil 4.70. DistilBERT Epoch bazlı metrikler-1	87
Şekil 4.71. DistilBERT Epoch bazlı metrikler-2	88
Şekil 4.72. DistilBERT zaman bazlı metrikler-1	88
Şekil 4.73. DistilBERT zaman bazlı metrikler-2	88
Şekil 4.74. DistilBERT eğitim logları	89
Şekil 4.75. DistilBERT eğitim metrikleri zaman bazlı	89
Şekil 4.76. DistilBERT Epoch bazlı metrikler	89
Şekil 4.77. DistilBERT tam log içeriği	90
Şekil 4.78. ElectraTR metrikleri	90
Şekil 4.79. ElectraTR eğitimi son durum metrikleri-1	90
Şekil 4.80. ElectraTR eğitimi son durum metrikleri-2	91
Şekil 4.81. ElectraTR Epoch bazlı metrikler-1	91
Şekil 4.82. ElectraTR Epoch bazlı metrikler-2	91
Şekil 4.83. ElectraTR zaman bazlı metrikler-1	92
Şekil 4.84. ElectraTR zaman bazlı metrikler-2	92
Şekil 4.85. ElectraTR eğitim logları	92
Şekil 4.86. ElectraTR eğitim metrikleri zaman bazlı	93
Şekil 4.87. ElectraTR Epoch bazlı metrikler	93
Şekil 4.88. ElectraTR tam log içeriği	93
Şekil 4.89. RoBERTa metrikleri	94
Şekil 4.90. RoBERTa eğitimin son durumu metrikleri-1	94

Şekil 4.91. RoBERTa eğitimin son durum metrikleri-2	94
Şekil 4.92. RoBERTa Epoch bazlı metrikler-1	95
Şekil 4.93. RoBERTa Epoch bazlı metrikler-2	95
Şekil 4.94. RoBERTa zaman bazlı metrikler-1	95
Şekil 4.95. RoBERTa zaman bazlı metrikler-2	96
Şekil 4.96. RoBERTa eğitim logları	96
Şekil 4.97. RoBERTa eğitim metrikleri zaman bazlı	96
Şekil 4.98. RoBERTa tam log içeriği	97
Şekil 4.99. SentimentBERT metrikleri	97
Şekil 4.100. SentimentBERT eğitim son durum metrikleri-1	97
Şekil 4.101. SentimentBERT eğitim son durum metrikleri-2	98
Şekil 4.102. SentimentBERT Epoch bazlı metrikler-1	98
Şekil 4.103. SentimentBERT Epoch bazlı metrikler-2	98
Şekil 4.104. SentimentBERT zaman bazlı metrikler-1	99
Şekil 4.105. SentimentBERT zaman bazlı metrikler-2	99
Şekil 4.106. SentimentBERT eğitim logları	99
Şekil 4.107. SentimentBERT eğitim metrikleri zaman bazlı	100
Şekil 4.108. SentimentBERT tam log içeriği	100
Şekil 4.109. Model eğitme-1	101
Şekil 4.110. Model eğitme-2	101
Şekil 4.111. Tüm model matrikleri-1	102
Şekil 4.112. Tüm model matrikleri-2	102

SİMGELER VE KISALTMALAR

Kısaltmalar

AI : Artificial Intelligence / Yapay Zekâ
ANCE : Approximate Nearest-Neighbor Contrastive Estimation
API : Application Programming Interface
BDM : Büyük Dil Modeli
BLEU : Bilingual Evaluation Understudy
BM25 : Best Matching 25 (olasılıksal erişim modeli)
ChatGPT : Chat Generative Pre-trained Transformer
DPR : Dense Passage Retrieval
FAISS : Facebook AI Similarity Search
HNSW : Hierarchical Navigable Small World
IDF : Inverse Document Frequency
LLM : Large Language Model
NLP : Natural Language Processing / Doğal Dil İşleme
OPAC : Online Public Access Catalog
RAG : Retrieval-Augmented Generation
RLHF : Reinforcement Learning with Human Feedback
ROUGE : Recall-Oriented Understudy for Gisting Evaluation
SQL : Structured Query Language
UI : User Interface / Kullanıcı Arayüzü
UX : User Experience / Kullanıcı Deneyimi
XAI : Explainable AI / Açıklanabilir Yapay Zekâ
YORDAM : Yerel kütüphane otomasyon sistemi
YZ: Yapay Zeka

1. GİRİŞ

21. yüzyılda bilgiye erişim şekilleri ve kullanıcı beklentileri, dijitalleşme ile köklü bir değişim geçirmiştir. Bu dönüşüm, özellikle üniversite kütüphanelerinde daha belirgin hale gelmiş, bilgiye erişimde hız, doğruluk ve kişiselleştirme gibi kavramlar öne çıkmıştır. Artık öğrenciler ve akademisyenler yalnızca kitapları ödünç almak ya da katalog taramakla yetinmemekte; aynı zamanda elektronik kaynaklara, tez arşivlerine, makale veri tabanlarına ve hatta yapay zekâ destekli önerilere tek noktadan ulaşmayı talep etmektedir (Nimbhorkar, 2024).

Bu bağlamda, kütüphane otomasyon sistemleri, klasik katalog erişimi işlevinin ötesine geçerek, kullanıcı dostu arayüzler, dijital içerik yönetimi, çoklu veri kaynağı entegrasyonu ve akıllı arama motorları gibi bileşenlerle zenginleşmiştir. Gelişmiş ülkelerde üniversite kütüphaneleri, bu tür teknolojilerle donatılmış, bilgiye erişimde neredeyse kusursuz sistemler inşa etmeyi başarmıştır (Li & Qiu, 2025). Ancak Türkiye’de bu geçiş süreci henüz tamamlanmamış, birçok üniversitede hâlâ yalnızca temel kataloglama hizmeti sunan sistemler kullanılmaktadır (Karasözen & Zan, 2010).

Batman Üniversitesi örneğinde, mevcut sistemlerde kullanıcılar katalog tarama, e-arşiv ve web aramaları arasında manuel geçiş yapmak zorunda kalmakta; bu da ciddi zaman kaybına ve bilgiye ulaşmada parçalanmış bir deneyime yol açmaktadır. İşte bu gereksinim doğrultusunda geliştirilen proje, YZ (Yapay Zeka) destekli bir kütüphane otomasyon sistemi olarak tasarlanmıştır. Sistem, yerel kaynaklarla sınırlı kalmayıp e-arşiv, akademik veri tabanları ve web üzerindeki açık erişim kaynaklarını da kapsayacak şekilde genişletilmiştir.

Bu çalışmanın en özgün yönlerinden biri, sistemin ChatGPT entegrasyonu ile güçlendirilmiş olmasıdır. ChatGPT, doğal dil işleme (NLP) teknikleriyle kullanıcı sorgularını daha anlamlı ve bağlamsal olarak yorumlamakta, yalnızca anahtar kelime eşleştirmesi yerine kavramsal benzerliği de dikkate alarak öneriler sunmaktadır (Furtado & Esmin, 2023). Ayrıca kullanıcılar açık uçlu sorular sorabilmekte, sistem bu soruları analiz edip en uygun kaynakları sunmakta, hatta gerektiğinde özetleyerek yanıtlamaktadır (Şekil 1.1).

Örneğin, bir kullanıcı “Sürdürülebilir enerji politikalarının Türkiye'deki yansımaları nelerdir?” şeklinde geniş kapsamlı bir soru sorduğunda, sistem yalnızca anahtar kelimelerle değil; bağlamla, tarihsel gelişimle ve kavramsal ilişkilerle çalışarak daha anlamlı kaynaklara yönlendirme yapmaktadır. Bu durum, klasik otomasyon sistemlerinden radikal biçimde farklılaşmakta ve bilgi erişimini kişiselleştirilmiş bir akademik danışmanlık hizmetine dönüştürmektedir.

Geliştirilen sistemin temel hedeflerinden biri, öğrencilerin ve akademik personelin bilgiye erişim sürecini sadeleştirmek ve hızlandırmaktır. Tek bir sorgu ile hem üniversitenin YORDAM sistemi hem de e-arşiv sonuçlarına ulaşmak mümkün kılınmış, sonuçların yeterli olmaması durumunda sistem otomatik olarak internetteki açık erişimli kaynaklara yönelmiştir. Bu katmanlı arama yaklaşımı, kullanıcıyı birden fazla platformla manuel olarak uğraşmaktan kurtarmış, zaman kazancı ve doğru bilgiye erişim oranında artış sağlamıştır.

Ayrıca sistemde kullanıcı davranışlarını analiz eden bir yapı da bulunmaktadır. Sıklıkla aranan konular, kullanıcıların eğitim seviyeleri, geçmiş sorgu geçmişleri gibi veriler analiz edilerek kişiselleştirilmiş kaynak önerileri sunulmaktadır. Böylece sadece arama yapmakla kalınmamakta; kütüphane sistemi bir öğrenen sistem haline gelmektedir.

Batman Üniversitesi özelinde geliştirilen bu sistem, yerel koşullara uygun, açık kaynak temelli, geliştirilebilir ve sürdürülebilir bir çözüm modeli sunmaktadır. Geliştirilen bu yapı, yalnızca bir bilgi sistemi değil; aynı zamanda Türkiye'deki diğer orta ölçekli üniversitelere örnek teşkil edebilecek bir dijitalleşme modelidir. Sistemin modüler yapısı, ilerleyen dönemlerde ulusal açık erişim veri tabanları, YÖKTEZ, TR Dizin, CrossRef ve ORCID gibi platformlarla entegrasyon için de hazırdır.



Şekil 1.1. Yapay zekâ destekli kütüphane otomasyon sisteminin genel yapısı

1.1. Araştırmanın Amacı

Bu araştırmanın amacı, Batman Üniversitesi kütüphanesine entegre edilmek üzere, Büyük Dil Modeli (BDM) tabanlı, yapay zekâ destekli akıllı kütüphane tarama ve diyalog sistemini geliştirmek ve değerlendirmektir. Sistem; çok kaynaklı (YORDAM, e-Arşiv, web) veri taraması yaparak, kullanıcıların akademik bilgiye hızlı, doğru ve etkin erişimini sağlamayı hedeflemektedir. Ayrıca, kullanıcı geri bildirimleriyle kendini geliştiren RLHF destekli yapısı sayesinde, kullanıcı deneyimini iyileştirmek ve bilgi erişim sürecinde açıklanabilirlik ve sürdürülebilirlik sunmak amaçlanmaktadır.

1.2. Problem Durumu

Geleneksel kütüphane otomasyon sistemleri (OPAC) kullanıcıların ihtiyaç duyduğu kapsamlı ve güncel akademik bilgiyi hızlı ve etkili biçimde sunmakta yetersiz kalmaktadır. Özellikle farklı veri tabanlarında dağınık halde bulunan bilgi kaynaklarına erişimde kullanıcılar zaman kaybı yaşamakta ve arama sonuçları çoğu zaman yetersiz veya güncel olmayan içeriklerden oluşmaktadır. Ayrıca, kullanıcıların sorgularına uygun, doğal dil işleme teknikleriyle desteklenmiş, akıllı ve diyalog tabanlı arama sistemlerinin eksikliği araştırma problemidir. Bu bağlamda, Batman Üniversitesi'nin mevcut kütüphane sistemine yapay zekâ tabanlı, çok kaynaklı ve kullanıcı merkezli bir çözüm getirilmesi gerekmektedir (Şekil 1.2).



Şekil 1.2. Geleneksel OPAC ile yapay zekâ tabanlı akıllı kütüphane sisteminin karşılaştırması

1.3. Araştırmanın Önemi

Bu çalışma, Türkiye’de üniversite kütüphanelerinde bilgi erişim süreçlerini yapay zekâ ile iyileştirmeye yönelik özgün bir model sunması açısından önem taşımaktadır. BDM tabanlı ve RLHF destekli akıllı sistem sayesinde, kullanıcıların akademik araştırmalarında daha hızlı, doğru ve kapsamlı bilgiye ulaşması sağlanacaktır. Ayrıca, Explainable AI teknikleri ile sistemin şeffaflığı artırılarak kullanıcı güveni desteklenecektir. Bu çalışma, akademik başarıyı doğrudan etkileyen bilgi erişimi alanında yenilikçi teknolojilerin uygulanmasına katkıda bulunarak, kütüphane hizmetlerinde dijital dönüşümü hızlandıracaktır (Şekil 1.3).



Şekil 1.3. BDM ve RLHF destekli sistemin bilgi erişimi ve akademik başarıya katkısı

1.4. Varsayımlar

- Kullanıcılar sisteme doğal dilde anlaşılır sorgular girmektedir.
- YORDAM ve e-arşiv sistemlerinden elde edilen veriler doğru ve günceldir.
- Kullanıcı geri bildirimleri dürüst ve anlamlıdır.
- Kullanılan büyük dil modeli (ChatGPT) güvenilir bir bilgi üreticisidir.
- Sistem arka plan bileşenleri (API, veritabanı, sorgu zinciri) doğru yapılandırılmıştır.

1.5. Sınırlılıklar

- Sistem yalnızca Batman Üniversitesi Kütüphanesi ve e-arşivi ile sınırlıdır.
- OpenAI tabanlı API maliyetleri nedeniyle yanıt uzunlukları sınırlandırılmıştır.
- İnternet bağlantısı olmayan cihazlarda web araması yapılamamaktadır.
- Kullanıcıların teknolojik yeterliliği sistemin etkin kullanımını etkileyebilir.
- Sistemde kullanılan dil modeli bazen hatalı veya eksik bilgi verebilir.

1.6. Tanımlar

- BDM (Büyük Dil Modeli): Çok kaynaklı bilgi sistemlerinde verinin dağıtık ve düzenli bir biçimde yönetilmesini sağlayan model.
- OPAC (Online Public Access Catalog): Kütüphane kataloglarının çevrimiçi erişim arayüzü.
- RAG (Retrieval-Augmented Generation): Veri tabanlarından bilgi çekme ve doğal dil üretimini birleştiren yapay zekâ mimarisi.
- RLHF (Reinforcement Learning with Human Feedback): İnsan geri bildirimlerine dayalı pekiştirmeli öğrenme yöntemi.
- CHROMA: Vektör veritabanı teknolojisi, anlamsal arama ve bellek yönetimi sağlar.
- Explainable AI (XAI): Yapay zekâ modellerinin karar süreçlerini kullanıcıya açıklayabilir kılma yöntemleri.
- LangChain: Çok kaynaklı ve zincirlenmiş yapay zekâ uygulamaları geliştirmek için kullanılan Python framework'ü.
- YORDAM: Batman Üniversitesi kütüphane katalog sistemi.
- E-Arşiv: Batman Üniversitesi'nin dijital arşiv platformu.

2. KAYNAK ARAŞTIRMASI

2.1. Büyük Dil Modellerinin (BDM) Tanımı ve Tarihsel Gelişimi

Büyük Dil Modelleri (BDM), çok büyük miktarda metin verisi üzerinde eğitilerek doğal dilde metin anlayabilen, üretebilen ve analiz edebilen derin öğrenme tabanlı yapay zekâ sistemleridir. Bu modeller, genellikle yüz milyonlarca veya milyarlarca parametreye sahip olup, öğrenme sürecinde dilin yapısal ve bağlamsal özelliklerini içselleştirmektedir (Brown et al., 2020). BDM'lerin evrimi (Şekil 2.1), doğal dil işleme tarihinin de önemli bir parçasıdır. İlk dönemlerde istatistiksel modeller ve n-gram temelli yöntemler yaygınken, 2017 yılında tanıtılan Transformer mimarisi (Vaswani et al., 2017), bu alanda devrim niteliğinde bir değişim yaratmıştır.

Bu gelişmenin ardından, 2018 yılında BERT (Bidirectional Encoder Representations from Transformers) modeli, iki yönlü bağlamı dikkate alarak daha isabetli dil temsilleri üretmeyi başarmıştır (Devlin et al., 2019). BERT'in başarısından sonra, GPT (Generative Pre-trained Transformer) modelleri, özellikle GPT-3 ile birlikte (Brown et al., 2020), az örnekle öğrenme (few-shot learning) yeteneklerini ön plana çıkarmış ve çok çeşitli doğal dil görevlerinde etkileyici sonuçlar üretmiştir. Bu modellerin yükselişi, yalnızca akademik alanda değil, kamuoyunda ve teknolojik ürünlerde de önemli bir yankı uyandırmıştır.

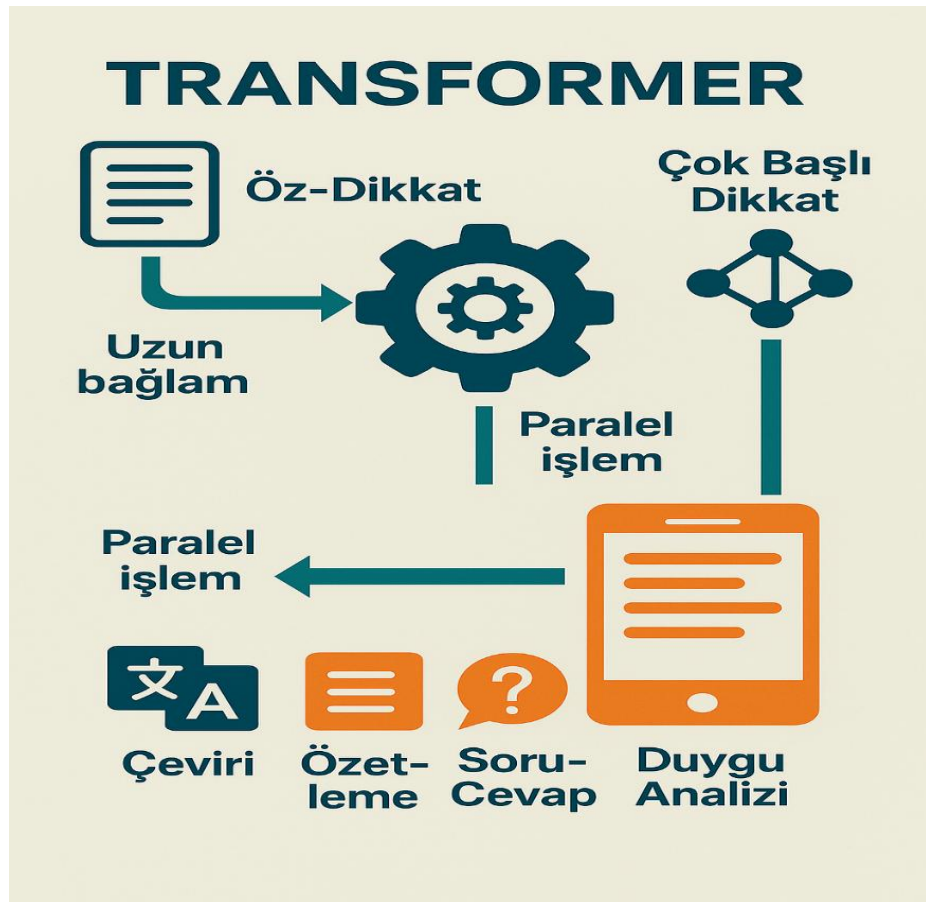


Şekil 2.1. Büyük dil modellerinin tarihsel evrimi

2.1.1. Transformer mimarisi ve dil modelleme

Transformer mimarisi, doğal dil işleme alanında performans açısından önceki yöntemlere kıyasla ciddi avantajlar sunmuştur. Transformer'ın temel yapısı olan kendili-dikkat (self-attention) mekanizması, bir cümlemin veya metnin her bir parçasının, diğer parçalarla olan ilişkisini aynı anda değerlendirmesine olanak tanır (Vaswani et al., 2017). Bu yapı, RNN veya LSTM gibi sıralı (sequential) mimarilerde görülen uzun-bağlam kaybını ortadan kaldırır (Şekil 2.2).

Transformer tabanlı modellerdeki çok başlı dikkat katmanları (multi-head attention), bağlam bilgisini farklı alt uzaylarda analiz ederek anlam derinliğini artırır. Bu sayede, modelin çeviri, özetleme, soru-cevap, metin üretimi ve duygu analizi gibi çok çeşitli görevlerde esnek bir şekilde kullanılabilmesi mümkün hâle gelir (Raffel et al., 2020). Modelin hem paralel çalışabilmesi hem de uzun metinleri etkili bir biçimde işleyebilmesi, onu BDM'lerin omurgası hâline getirmiştir.



Şekil 2.2. Transformer modelinin yapısı

2.1.2. BDM'lerin NLP'de başlıca kullanım alanları

BDM'lerin en dikkat çekici özelliği, çeşitli doğal dil işleme görevlerinde önceden eğitilmiş tek bir modelin çoklu görevlerde kullanılabilmesidir (Şekil 2.3). Bu kapsamda;

- Makine Çevirisi: Özellikle Google'ın GNMT modeli ve ardından geliştirilen Transformer tabanlı modeller, çeviri kalitesinde insan düzeyine yaklaşmıştır (Wu et al., 2016).
- Soru-Cevap Sistemleri: RAG (Retrieval-Augmented Generation) tabanlı modeller, bilgiyi dış bir kaynaktan arayıp daha doğru ve açıklamalı cevaplar üretmektedir (Lewis et al., 2020).
- Özetleme ve Bilgi Yoğun Metin Analizi: Özellikle BART ve T5 gibi modeller, uzun metinleri özetleme konusunda yüksek başarı göstermektedir (Raffel et al., 2020).
- Duygu Analizi ve Metin Sınıflandırma: BDM'ler, bağlamsal bilgiye dayalı olarak kelimelerin anlamını daha doğru değerlendirdiği için, geleneksel yöntemlere kıyasla daha yüksek F1 puanları elde etmektedir (Jurafsky & Martin, 2023).
- Sohbet Robotları ve Diyalog Sistemleri: GPT serisi, kullanıcıyla doğal bir şekilde sohbet edebilen yapılar sunarak müşteri hizmetleri, eğitim, sağlık ve kütüphane gibi alanlarda kullanılmaktadır (OpenAI, 2023).



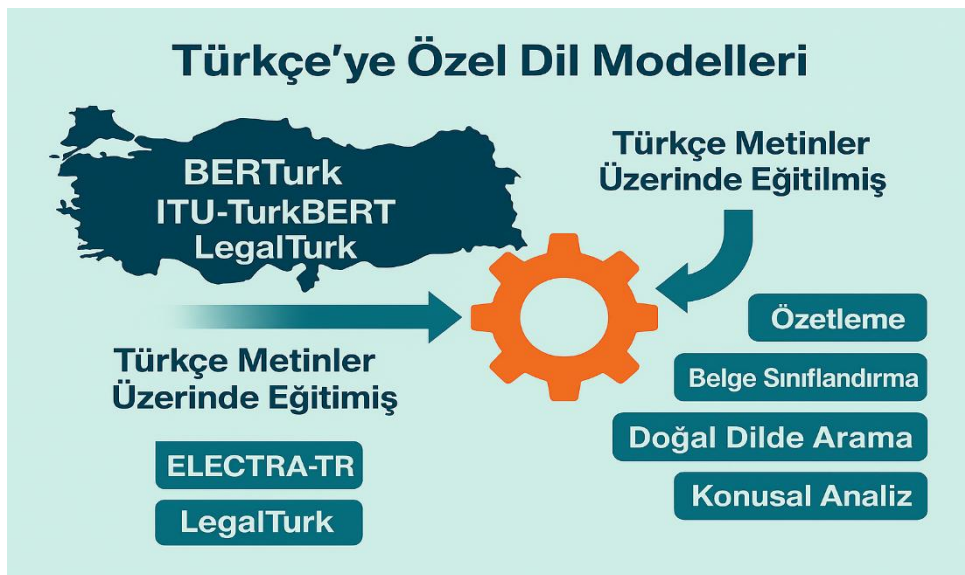
Şekil 2.3. Büyük dil modellerinin doğal dil işleme alanındaki başlıca kullanım alanları

2.1.3. Türkçe BDM modelleri ve uygulamaları

Türkçenin eklemeli bir dil yapısına sahip olması, doğal dil işleme süreçlerinde bazı özel zorluklar doğurmaktadır. Bu nedenle, Türkçe'ye özel olarak geliştirilen dil modelleri oldukça önemlidir. BERTurk, 2020 yılında DBMDZ ekibi tarafından Hugging Face platformunda yayımlanmış ve Türkçe metinler üzerinde eğitilerek, çok dilli modellerden daha iyi performans göstermiştir (Şweter, 2020).

ELECTRA-TR (Şahin & Eryiğit, 2021), BERT'e alternatif olarak geliştirilmiş ve ayrımcı (discriminative) eğitim yaklaşımıyla daha az işlem gücü kullanarak daha hızlı ve etkili sonuçlar sağlamıştır. Ayrıca, 2023 itibarıyla Türkiye'de çeşitli üniversiteler tarafından geliştirilen ITU-TurkBERT ve LegalTurk gibi modeller; eğitim, hukuk ve sağlık gibi spesifik alanlara yönelik Türkçe doğal dil uygulamalarında yüksek doğrulukla çalışmaktadır (Kılınç & Fişne, 2023).

Bu modellerin kütüphane sistemlerine entegrasyonu; metin özetleme, belge sınıflandırma, doğal dilde arama ve konusal analiz gibi görevlerde ciddi katkı sağlamaktadır (Şekil 2.4).



Şekil 2.4. Türkçe dil modelleri ve kullanım alanları

2.1.4. BDM'lerin avantajları ve sınırlılıkları

BDM'ler birçok avantaj sunsa da, çeşitli sınırlılıklara da sahiptir (Şekil 2.5). Avantajları arasında:

- Karmaşık dil yapısını anlama kabiliyeti,
- Çok sayıda göreve uyarlanabilirlik (transfer learning),
- Düşük örnekli öğrenme yeteneği (few-shot / zero-shot),
- Tutarlı ve doğal dil üretimi yer almaktadır (Bommasani et al., 2021).
- Sınırlılıkları ise şu şekilde sıralanabilir:
- Yüksek Donanım Gereksinimi: BDM'lerin eğitimi ve çalıştırılması büyük miktarda hesaplama gücü ve enerji tüketimi gerektirir (Strubell et al., 2019).
- Halüsinasyon Riski: Model, veri setinde yer almayan yanlış bilgiler üretme eğiliminde olabilir (Bender et al., 2021).
- Etik Sorunlar ve Önyargı: Eğitim verilerinde yer alan sosyal önyargılar, model çıktısına da yansiyabilir (Blodgett et al., 2020).
- Şeffaflık Eksikliği: Modelin verdiği kararları açıklamak veya izlemek zordur; bu da kullanıcı güvenini olumsuz etkileyebilir.



Şekil 2.5. Büyük dil modellerinin (BDM) avantajları ve sınırlılıkları

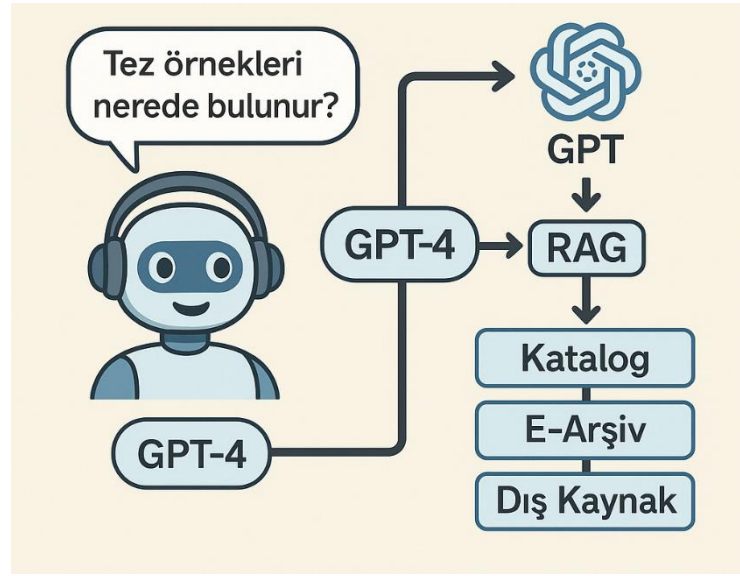
2.1.5. Akıllı sistemlerde BDM kullanımı: Chatbot ve Arama

BDM tabanlı sistemlerin en yaygın kullanım alanlarından biri de chatbot'lar ve akıllı arama sistemleridir. GPT tabanlı modeller, kullanıcının sorularına doğal ve

anlamalı cevaplar üretebilirken, RAG mimarisi sayesinde bu yanıtlar bilgi tabanlı belgelerle desteklenebilmektedir (Lewis et al., 2020).

Özellikle kütüphane sistemlerinde, kullanıcının “tez örnekleri nerede bulunur?” gibi doğal dildeki sorularına doğrudan katalog bağlantısı sunabilen, e-arşiv veritabanına bağlanan ve gerekirse dış kaynaklarda araştırma yapabilen yapay zekâ asistanları geliştirilmiştir (Zhang et al., 2023).

Bu sistemler, sadece bilgiye erişimi kolaylaştırmakla kalmaz, aynı zamanda kullanıcıya daha etkileşimli ve kişiselleştirilmiş bir hizmet sunar (Şekil 2.6).



Şekil 2.6. Akıllı sistemlerde BDM kullanımı: GPT tabanlı Chatbot ve akıllı arama

2.1.6. BDM’lerin kütüphane sistemlerine entegrasyonu

Kütüphaneler artık sadece kitap ödünç verilen yerler değil, bilgiye erişimin ve üretimin merkezleri olarak konumlanmaktadır. BDM tabanlı sistemler, bu dönüşümde önemli bir rol üstlenmektedir (Şekil 2.7).

- **Semantik Arama:** Kullanıcılar anahtar kelime yerine doğal dilde sorgu yapabilir. Örneğin: “Yapay zekânın etik sorunları üzerine Türkçe makaleler nelerdir?” gibi bir sorgu, doğrudan katalog sistemine yönlendirilir (Ünal & Demiralay, 2021).
- **Özetleme ve Tavsiye Motorları:** Kullanıcının ilgi alanına göre akademik makaleler önerilebilir ve uzun belgelerin özetleri sunulabilir.

- Etkinlik ve Eğitim Destekleri: BDM’ler, kütüphane rehberliği, APA atıf eğitimi ve kaynak bulma sürecinde kullanıcılara doğal dilde yardımcı olabilir (Zhang et al., 2023).
- Bu sayede, bilgiye erişim hem daha hızlı hem de daha kullanıcı dostu bir hâl alır.



Şekil 2.7. BDM’lerin kütüphane sistemlerine entegrasyon alanları

2.2. Bilgi Erişim ve Kütüphane Sistemleri

Bilgi erişim sistemleri, kullanıcıların bilgi kaynaklarına hızlı, doğru ve etkili biçimde ulaşabilmesini sağlamak amacıyla geliştirilen yapısal çözümlerdir. Üniversite kütüphaneleri özelinde, bu sistemlerin temel işlevi, kataloglar, dijital kaynaklar ve dış veri tabanları gibi bilgi kaynaklarının bütünleşik bir biçimde yönetimini ve erişimini sağlamaktır (Tonta, 2010). Özellikle son yıllarda, geleneksel kütüphane sistemlerinin yerini entegre ve dijital odaklı yapılar almaya başlamıştır (Şekil 2.8).



Şekil 2.8. Entegre ve dijital odaklı bilgi erişim sistemleri

2.2.1. Geleneksel OPAC (Online Public Access Catalog) mimarisine genel bakış

Geleneksel OPAC sistemleri (Şekil 2.9), kullanıcıların yalnızca bibliyografik bilgilere erişebildiği, statik arayüzlere sahip olan sistemlerdir. 1980’li yıllarda yaygınlaşmaya başlayan bu yapılar, MARC (Machine-Readable Cataloging) standardı üzerine kurulu olup, sadece başlık, yazar ve konu gibi sınırlı alanlarda arama yapılmasına imkân tanımaktadır (Borgman, 1996). Türkiye’de YORDAM gibi sistemler hâlâ büyük oranda bu mantıkla çalışmakta ve gelişmiş bilgi erişim tekniklerinden uzak kalmaktadır (Karasözen & Zan, 2010).

Bu yapılar, kullanıcı merkezli tasarıma sahip olmamaları nedeniyle modern bilgi ihtiyaçlarına cevap verememektedir. Örneğin OPAC sistemleri genellikle çoklu kaynak taraması, doğal dil sorgusu ya da kişiselleştirilmiş öneri sunumu gibi ileri seviye fonksiyonları desteklemez (Breeding, 2022). Ayrıca veri siloları hâlinde çalışan bu sistemlerde, e-kitaplar, tezler veya dış veritabanları ile entegrasyon da sınırlıdır.

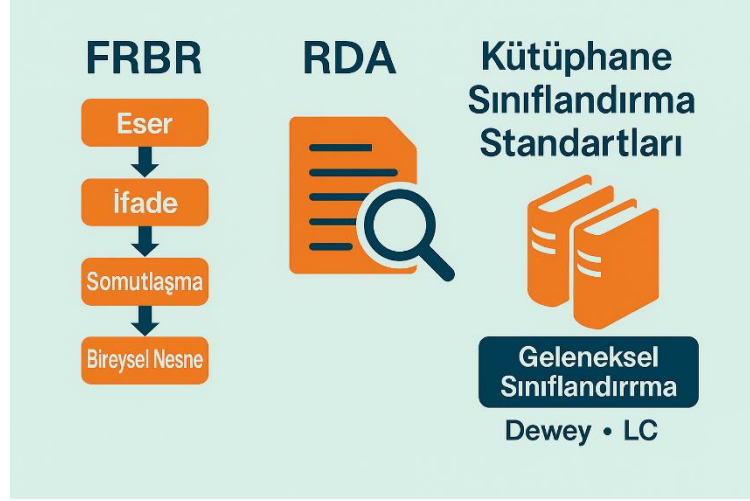


Şekil 2.9. Geleneksel OPAC sistemlerinin özellikleri

2.2.2. FRBR, RDA ve kütüphane sınıflandırma standartları

Bilgi kaynaklarının yapılandırılması ve sınıflandırılması (Şekil 2.10), kütüphane otomasyon sistemlerinin doğruluğu ve sürdürülebilirliği açısından kritik öneme sahiptir. FRBR (Functional Requirements for Bibliographic Records) modeli, bilgi kaynaklarının “eser”, “ifade”, “somutlaşma” ve “bireysel nesne” düzeylerinde ele alınmasını öngörerek OPAC sistemlerine işlevsel bir boyut katmıştır (IFLA, 1998). Daha sonra geliştirilen RDA (Resource Description and Access) standardı ise bu modeli genişletmiş, dijital kaynakların sınıflandırılmasında daha esnek bir yapı sunmuştur (Oliver, 2010).

Ancak Türkiye’de bu standartların yaygın olarak uygulanmadığı; çoğu sistemin hâlâ Dewey veya LC gibi geleneksel sınıflandırma sistemleriyle çalıştığı görülmektedir. Özellikle dijital koleksiyonların artması, metadata zenginliğini ve sınıflandırma sistemlerinin çok kaynaklı yapılara uyumunu zorunlu kılmaktadır (Yılmaz & Ünal, 2021).



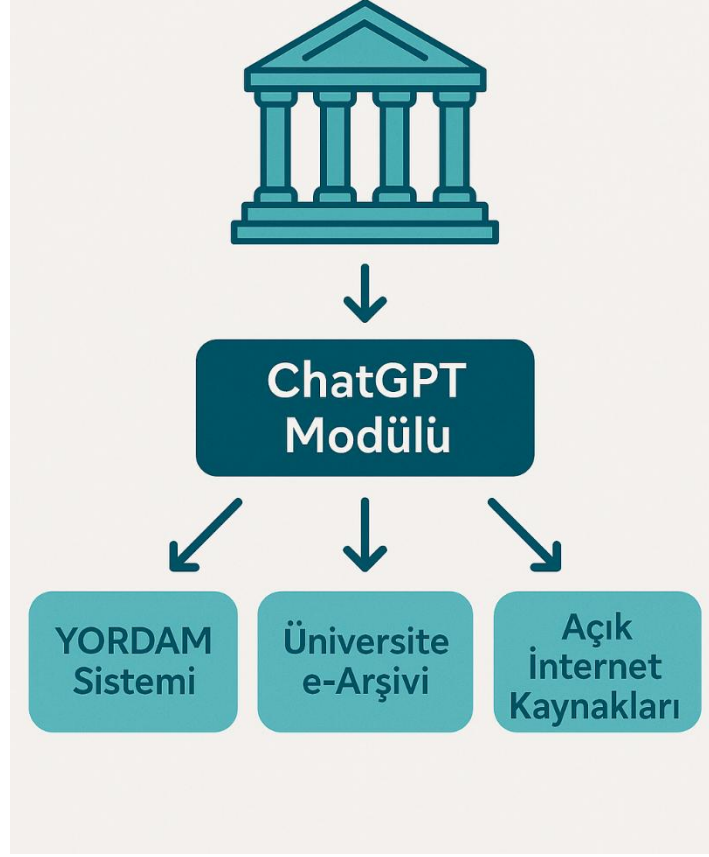
Şekil 2.10. FRBR ve RDA'nın dijital sınıflandırmadaki rolü

2.2.3. Dijital kütüphane çerçeveleri ve çok kaynaklı entegrasyon

Modern kütüphane sistemleri, yalnızca OPAC tabanlı yapıların ötesine geçerek, e-dergiler, tez arşivleri, veri kümeleri ve web tabanlı kaynakları bütüncül bir yapıda birleştirmeye başlamıştır. Dijital kütüphane çerçeveleri, metadata standardizasyonu, API tabanlı erişim ve OAI-PMH gibi protokoller aracılığıyla çok kaynaklı entegrasyonu mümkün kılmaktadır (Lagoze et al., 2002). Bu entegrasyon sayesinde kullanıcılar tek arayüzden hem katalog hem de açık erişimli kaynaklara erişebilmektedir.

Batman Üniversitesi için geliştirilen bu projede de benzer bir yaklaşımla YORDAM sistemi, üniversite e-arşivi ve açık internet kaynakları tek çatı altında toplanmış; kullanıcı sorguları ChatGPT modülü aracılığıyla bu çoklu kaynaklara yönlendirilmiştir. Böylece geleneksel bilgi erişim sistemlerinin ötesine geçilerek bütünlük, akıllı ve öneri odaklı bir yapı oluşturulmuştur (Şekil 2.11).

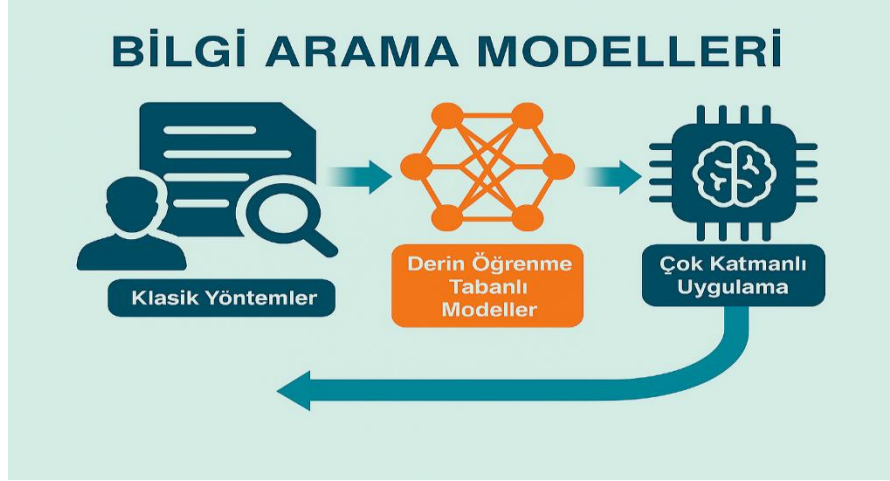
Dijital Kütüphane Çerçevesi ve Çok Kaynaklı Entegrasyon



Şekil 2.11. Batman Üniversitesi dijital kütüphane entegrasyon yapısı

2.3. Bilgi Arama Modelleri

Bilgi erişim sistemlerinin çekirdeğinde yer alan bilgi arama modelleri (Şekil 2.12), kullanıcının sorgusuna uygun belgelerin seçilmesini sağlayan matematiksel ve istatistiksel yapılar bütünüdür. Bu modeller, arama sonuçlarının sıralanmasında ve alaka düzeyinin hesaplanmasında önemli rol oynar. Gelişen teknolojiyle birlikte klasik yöntemlerden derin öğrenme tabanlı modellere evrilen bu süreç, günümüzde yapay zekâ destekli sistemlerde çok katmanlı bir şekilde uygulanmaktadır (Manning, Raghavan & Schütze, 2008).

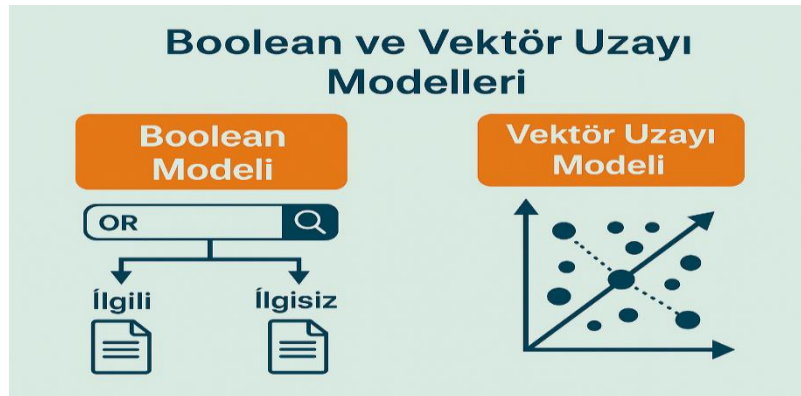


Şekil 2.12. Bilgi erişim sistemlerinde kullanılan bilgi arama modellerinin evrimi

2.3.1. Boolean ve vektör uzayı modelleri

En temel bilgi erişim modellerinden biri olan Boolean modeli (Şekil 2.13), kullanıcı sorgularını AND, OR ve NOT gibi mantıksal işlemlerle değerlendirerek belgeleri eşleştirir. Her belge ya ilgili ya da ilgisiz olarak değerlendirilir; bu nedenle alaka derecesi hesaplanmaz. Bu modelin en büyük dezavantajı, çok katı olması ve belgeleri sıralayamamasıdır (Salton, 1983).

Vektör Uzayı Modeli (Vector Space Model - VSM) ise belgeleri ve sorguları çok boyutlu bir vektör uzayında temsil eder. Bu yaklaşımda, her belge ve sorgu birer terim vektörüdür ve iki vektör arasındaki kosinüs benzerliği ölçülerek alaka derecesi belirlenir. VSM, Boolean modele göre daha esnek olup sıralı sonuçlar sunabilmesi açısından kullanıcı deneyimini artırır (Salton, Wong & Yang, 1975).

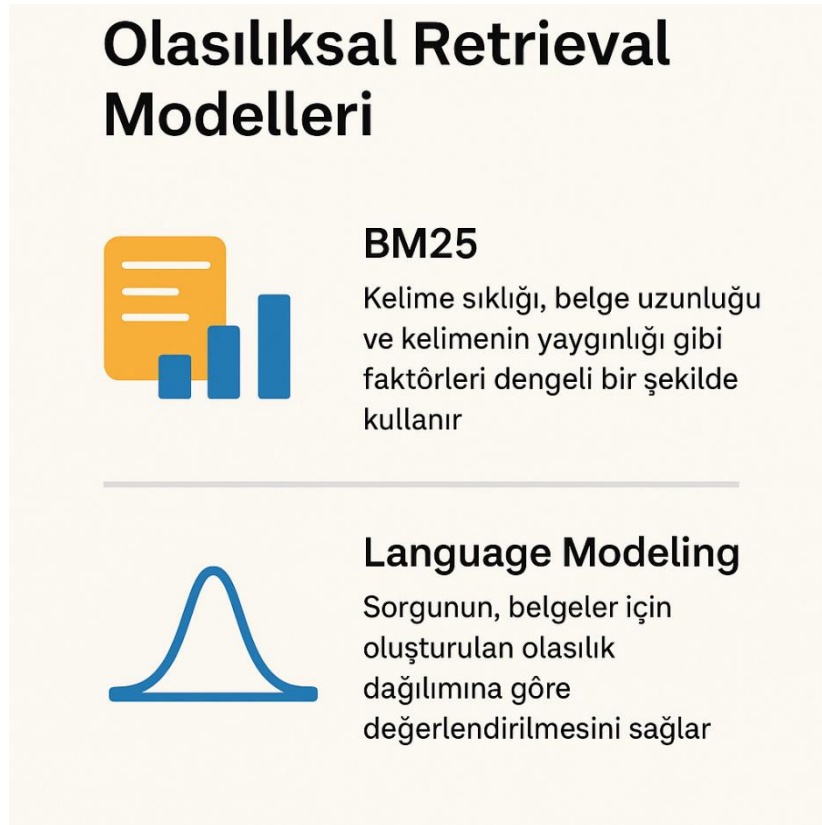


Şekil 2.13. Boolean ve vektör uzayı modellerinin karşılaştırılması

2.3.2. Olasılıksal retrieval modelleri (BM25, Language Modeling)

Olasılıksal modeller (Şekil 2.14), bir belgenin kullanıcı sorgusuyla ilgili olma olasılığını tahmin etmeye çalışır. En yaygın kullanılan modellerden biri olan BM25 (Best Matching 25), kelime sıklığı (TF), belge uzunluğu ve kelimenin belgeler arasındaki yaygınlığı (IDF) gibi faktörleri dengeli bir şekilde kullanır (Robertson & Zaragoza, 2009). BM25, özellikle akademik arama motorlarında (Google Scholar, PubMed) oldukça yaygın kullanılır.

Diğer yandan, Language Modeling (LM) yaklaşımı, her belge için bir olasılık dağılımı oluşturur ve sorunun bu modele göre üretilme olasılığını hesaplar. Bu yöntem, özellikle büyük ölçekli koleksiyonlarda etkili sonuçlar verebilmekte ve günümüzde neural retriever modellerinin temelini oluşturmaktadır (Ponte & Croft, 1998).



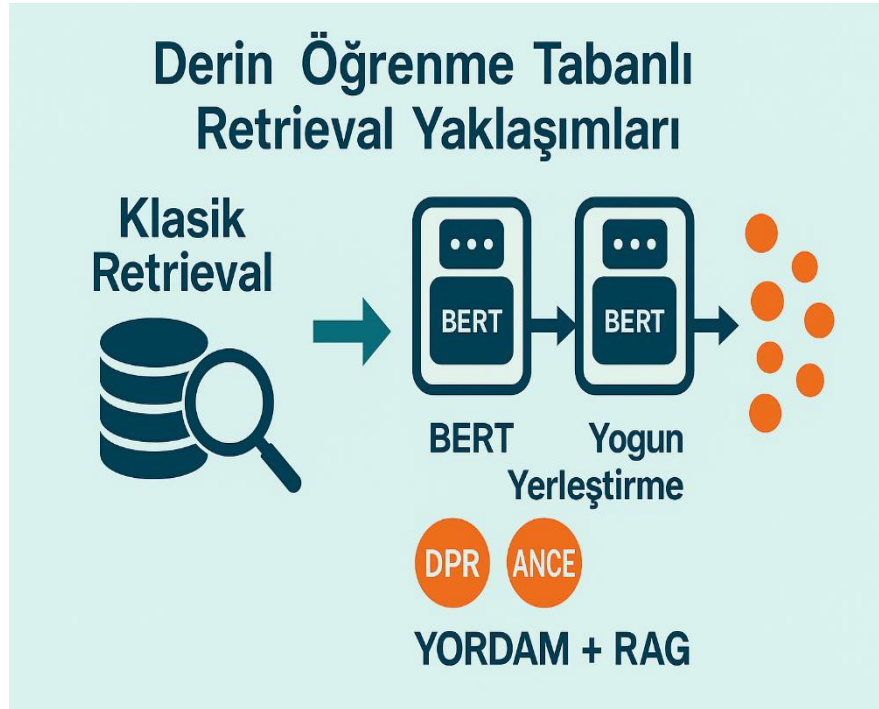
Şekil 2.14. Olasılıksal bilgi erişim modelleri: BM25 ve Language Modeling

2.3.3. Derin öğrenme tabanlı RETRIEVAL yaklaşımları (DPR, ANCE)

Geleneksel retrieval modellerinin yerini hızla alan derin öğrenme tabanlı retrieval sistemleri (Şekil 2.15), belge ve sorguları yüksek boyutlu vektörler olarak temsil eden dense embedding yapıları kullanır. Bu yaklaşımda sorgu ve belge eşlemesi, genellikle bir bi-encoder veya cross-encoder mimarisiyle yapılır (Karpukhin et al., 2020).

Örneğin, Dense Passage Retrieval (DPR) modeli, BERT tabanlı bir encoder ile belgeleri ve sorguları ayrı ayrı vektörleştirir ve FAISS gibi vektör tabanlı bir indeks üzerinden en benzer sonuçları getirir. Benzer şekilde, ANCE (Approximate Nearest Neighbor Negative Contrastive Estimation) modeli, eğitim sürecinde hard-negative sampling kullanarak daha güçlü eşleştirmeler yapar (Xiong et al., 2021).

Batman Üniversitesi örneğinde geliştirilen sistemde, geleneksel YORDAM sistemi sonuçları genişletmek amacıyla vektör tabanlı dense retrieval ve RAG mimarisiyle desteklenen derin öğrenme modelleri entegre edilmiştir. Böylece hem klasik indeksli erişim hem de bağlamsal benzerlik temelli erişim birlikte kullanılmaktadır.

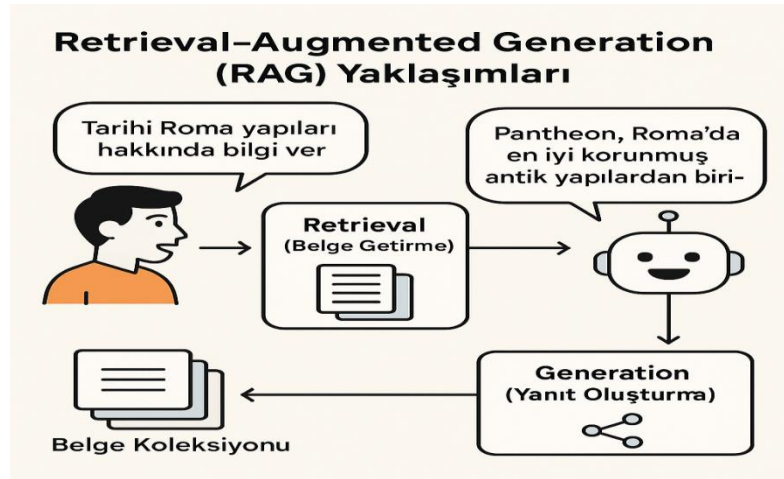


Şekil 2.15. Derin öğrenme destekli belge-sorgu eşleştirme mimarileri

2.4. Retrieval-Augmented Generation (RAG) Yaklaşımları

Bilgiye erişim ve üretim süreçlerinde, geleneksel bilgi alma (retrieval) ve metin üretimi (generation) modelleri uzun süre ayrı ayrı kullanılmıştır (Şekil 2.16). Ancak özellikle bilgi yoğun doğal dil işleme görevlerinde bu iki süreci entegre eden yeni modeller geliştirilmiştir. Retrieval-Augmented Generation (RAG), bu ihtiyaçtan doğmuş, bilgi alma ile metin üretimini birleştiren bir yapay zekâ yaklaşımıdır (Lewis et al., 2020).

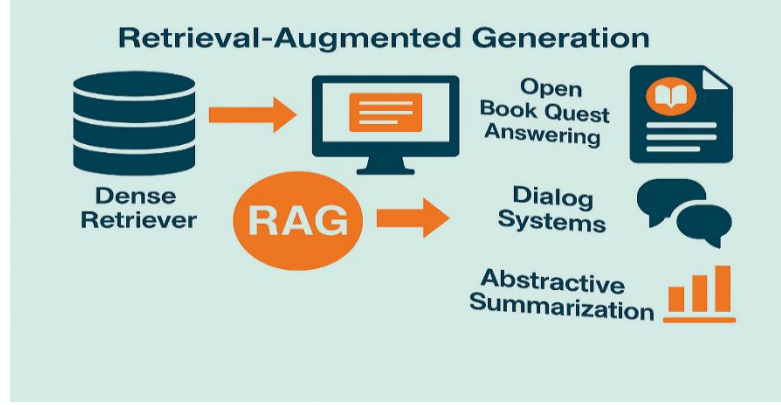
RAG mimarileri, kullanıcıdan gelen doğal dildeki soruyu önce bir retriever aracılığıyla belge koleksiyonunda arar, ardından ilgili belgeleri generator bileşenine vererek cevap üretir. Bu yöntem, salt dil modeli temelli yaklaşımlara kıyasla daha doğru, tutarlı ve kaynaklı yanıtlar üretilmesini sağlar (Izacard & Grave, 2021).



Şekil 2.16. Retrieval-augmented generation (RAG) yaklaşımının çalışma prensibi

2.4.1. RAG temelleri

RAG kavramı ilk olarak Lewis ve arkadaşları (2020) tarafından “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks” başlıklı çalışmada ortaya atılmıştır. Bu çalışmada, Dense Retriever + BART tabanlı bir generator kombinasyonu ile açık alan soru yanıtlama, diyalog sistemleri ve bilgiye dayalı özetleme gibi görevlerde RAG sistemlerinin performans avantajı ortaya konmuştur (Şekil 2.17). Özellikle Wikipedia üzerinden belge çekerek yanıt üreten sistem, GPT-2 gibi salt generative modellere kıyasla daha doğruluk odaklı çalışmıştır.

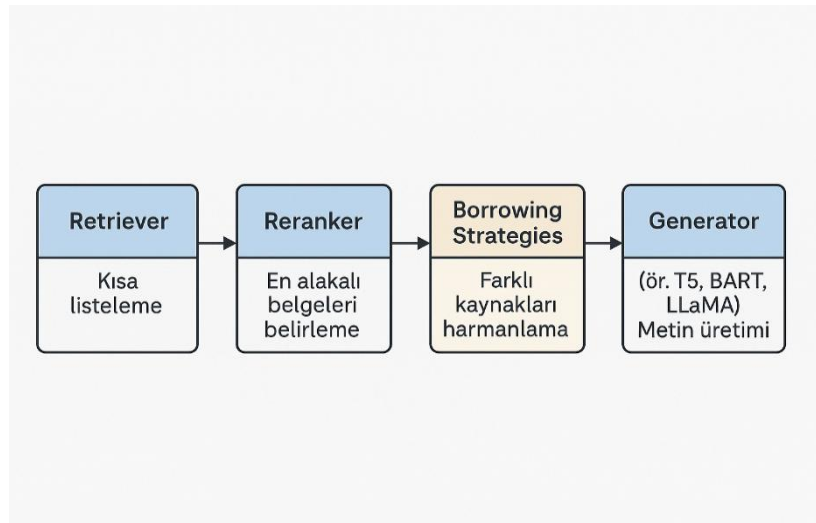


Şekil 2.17. Retrieval-augmented generation yaklaşımının genel şeması

2.4.2. Çok aşamalı RAG pipeline'ları ve borçlanma stratejileri

Günümüzde RAG sistemleri daha da gelişerek çok aşamalı hale gelmiştir. Örneğin, önce ilk aşamada kısa listeleme yapan bir retriever, ardından reranker (yeniden sıralayıcı) ile en alakalı belgeleri belirlemekte; son aşamada ise generator modeli (ör. T5, BART, LLaMA) metni üretmektedir (Gua et al., 2020). Bu yapı, sistemin hem hızını hem de yanıt kalitesini artırır (Şekil 2.18).

Ayrıca bazı sistemlerde borçlanma stratejileri (borrowing strategies) adı verilen yöntemlerle, RAG pipeline içinde farklı kaynaklardan alınan cevaplar harmanlanmakta; birden fazla model ya da bilgi deposu birlikte değerlendirilmektedir. Bu sayede hallucination (uydurma cevap üretme) oranı azaltılmakta ve güvenilirlik artırılmaktadır.

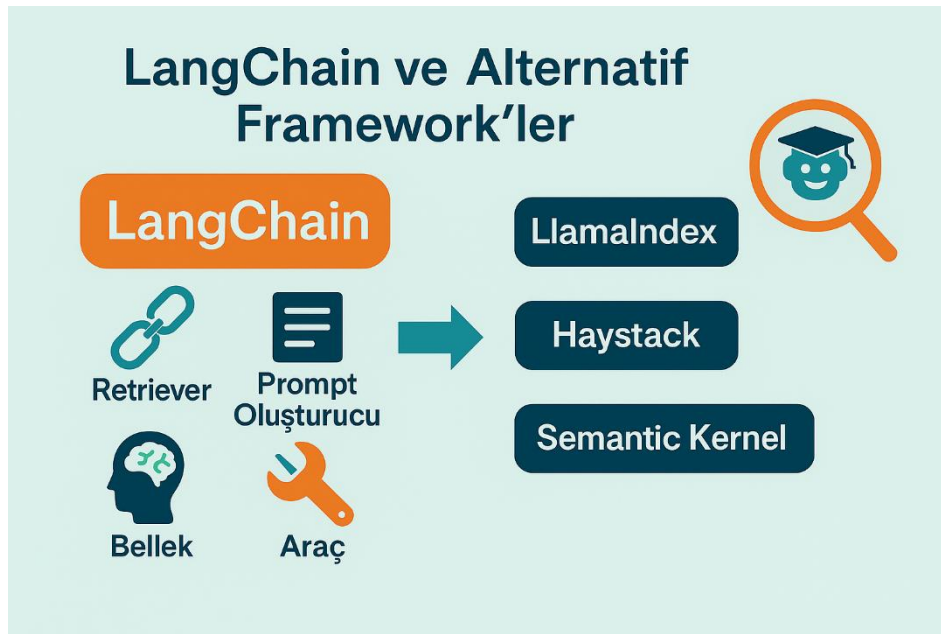


Şekil 2.18. Çok aşamalı RAG sistemlerinde bilgi getirme ve metin üretimi süreci

2.4.3. Langchain ve alternatif framework'ler

RAG sistemlerinin gerçek dünya uygulamalarında en çok kullanılan araçlardan biri LangChain kütüphanesidir. LangChain, Python tabanlı modüler yapısıyla retriever, prompt builder, memory ve tool bileşenlerini kolayca birleştirerek hızlı RAG prototipleri oluşturmayı mümkün kılar (Chase, 2023). LangChain, OpenAI, Hugging Face ve Chroma gibi API'lerle entegre çalışarak özellikle akademik projeler ve chatbot uygulamaları için ideal bir platform sunar.

Alternatif olarak LlamaIndex, Haystack ve Semantic Kernel gibi kütüphaneler de benzer amaçlarla kullanılır. Bu çerçevede, Batman Üniversitesi için geliştirilen sistemde LangChain altyapısıyla YORDAM katalogu, e-arşiv ve genel web kaynakları aynı retriever zinciri içinde sorgulanmakta; ardından ChatGPT destekli bir generator modülü ile anlamlı yanıtlar üretilmektedir (Şekil 2.19).

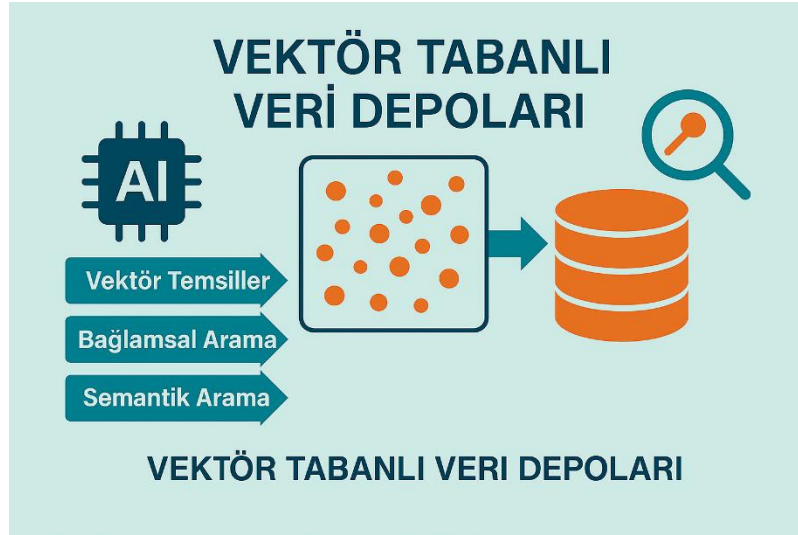


Şekil 2.19. Retriever-generator zincirinde framework kullanımı

2.5. Vektör Tabanlı Veri Depoları

Yapay zekâ destekli bilgi erişim sistemlerinde, özellikle retrieval-augmented generation (RAG) mimarilerinde, klasik anahtar kelime indekslemeye dayalı sistemlerin yerine vektör temsiller ve bu temsillere göre çalışan veri yapıları kullanılmaya

başlanmıştır. Belgelerin ve sorguların yüksek boyutlu vektörler olarak temsil edilmesiyle, daha bağlamsal ve semantik düzeyde arama yapılabilir (Johnson et al., 2019). Bu bağlamda vektör tabanlı veri depoları, bu temsillerin saklandığı, arandığı ve en benzer sonuçların getirildiği altyapıyı sağlar (Şekil 2.20).



Şekil 2.20. Vektör tabanlı arama altyapısı

2.5.1. FAISS, ANNOY, HNSW ve CHROMA karşılaştırması

En bilinen vektör arama kütüphanelerinden biri olan FAISS (Facebook AI Similarity Search), milyonlarca vektör üzerinde hızlı ve verimli k-en yakın komşu (k-NN) aramaları yapabilmekte, CPU ve GPU destekli altyapısıyla büyük veri kümeleri için ölçeklenebilir çözümler sunmaktadır (Johnson et al., 2019). Özellikle akademik projelerde ve arama motorlarında sıklıkla kullanılmaktadır.

Annoy (Approximate Nearest Neighbors Oh Yeah), Spotify tarafından geliştirilmiş, disk bazlı ve hafif bir yapı sunan vektör arama kütüphanesidir. Bellek kullanımının düşük olması nedeniyle mobil uygulamalarda tercih edilir. Ancak FAISS kadar hassas sonuçlar vermez (Bernhardsson, 2018).

HNSW (Hierarchical Navigable Small World) algoritması ise, grafik tabanlı bir yapı üzerinde vektörler arası gezinerek yüksek doğrulukta sonuçlar elde eder. Özellikle Haystack, Vespa gibi framework'lerde kullanılır (Malkov & Yashunin, 2018).

Yeni nesil uygulamalarda ise Chroma gibi modern, Python-native kütüphaneler öne çıkmaktadır. ChromaDB, LangChain ile doğrudan entegre olabilmesi, filtreleme desteği,

metadata işleyebilmesi ve embedding güncellemelerine açık mimarisi sayesinde, araştırma projeleri ve chatbot sistemlerinde tercih edilmektedir (Şekil 2.21).

FAISS, Annoy, HNSW ve Chroma Karşılaştırması

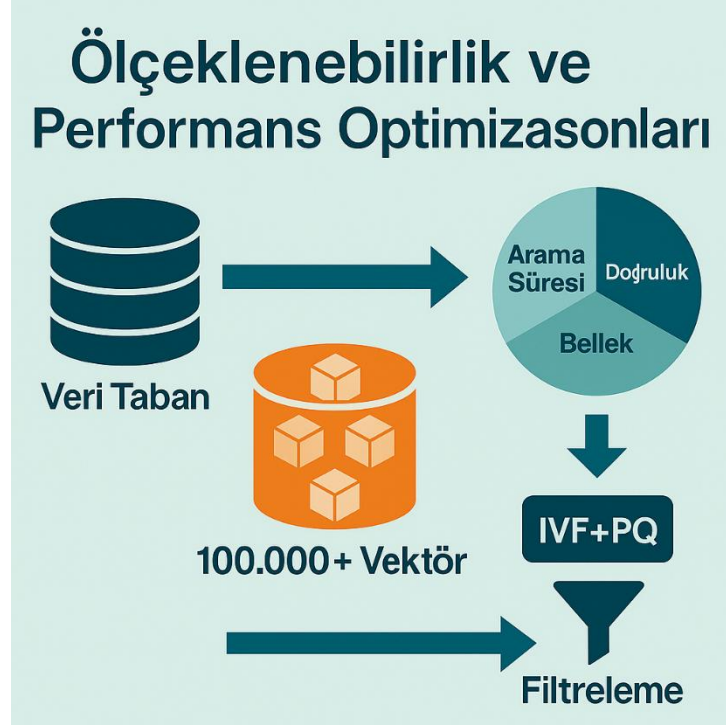
FAISS (Facebook AI Similarity Search) CPU ve GPU destekli, büyük veri kümesi için ölçeklenebilir; akademik projeler ve arama motorlarında yaygın kul-	Annoy (Approximate Nearest Neighbors On Yeah) Disk bazlı, hafif; düşük bellek kullanımı sayesinde mobil uygulamalarda tercih edilir, Hassasiyeti FAISS'tten düşüktür.
HNSW (Hierarchical Navigable Small World) Grafik tabanlı yapı; vektörler arası gezinerek yüksek doğruluk sağlar. Haystack ve Vespa gibi framework'lerde kullanılır	Chroma (Modern, Python-native) LangChain entegrasyon, filtreleme ve metadata desteğiyle araştırma projeleri ve chatbot sistemlerinde öne çıkar

Şekil 2.21. FAISS, Annoy, HNSW ve Chroma vektör arama kütüphanelerinin karşılaştırması

2.5.2. Ölçeklenebilirlik ve performans optimizasyonları

Vektör veri tabanlarının performansı, üç temel parametre ile belirlenir: arama süresi, doğruluk oranı ve bellek kullanımı. Büyük veri setlerinde doğruluk ve hız arasında denge kurmak kritik bir optimizasyon problemidir. Örneğin, FAISS'te kullanılan IVF+PQ (Inverted File Index + Product Quantization) yaklaşımı, disk alanından tasarruf sağlarken belirli bir doğruluk kaybı göze alınır (Johnson et al., 2019).

Batman Üniversitesi için geliştirilen sistemde, 100.000+ embedding'in saklandığı yapı, Chroma üzerine kurulmuş; hem FAISS benzeri arama algoritmaları hem de LangChain destekli metadata filtreleme mekanizması ile entegre edilmiştir. Bu yapı sayesinde, hem hızlı hem de sorgu bağlamına göre daha anlamlı sonuçlar alınabilmektedir (Şekil 2.22).



Şekil 2.22. Vektör veri tabanlarında ölçeklenebilir yapı

2.6. Kullanıcı Geri Bildirimi ve Öğrenme

Yapay zekâ destekli bilgi sistemlerinde kullanıcı geri bildirimi (user feedback), sistemin zaman içinde daha doğru, etkili ve kişiselleştirilmiş yanıtlar üretebilmesi açısından kritik bir rol oynar (Şekil 2.23). Kullanıcı etkileşimlerinden öğrenen sistemler, yalnızca sabit modellerle değil, deneyimle şekillenen algoritmalarla desteklenmektedir. Bu kapsamda Relevance Feedback, RLHF (Reinforcement Learning with Human Feedback) ve online öğrenme döngüleri gibi yaklaşımlar önem kazanmıştır (Radlinski & Joachims, 2007).

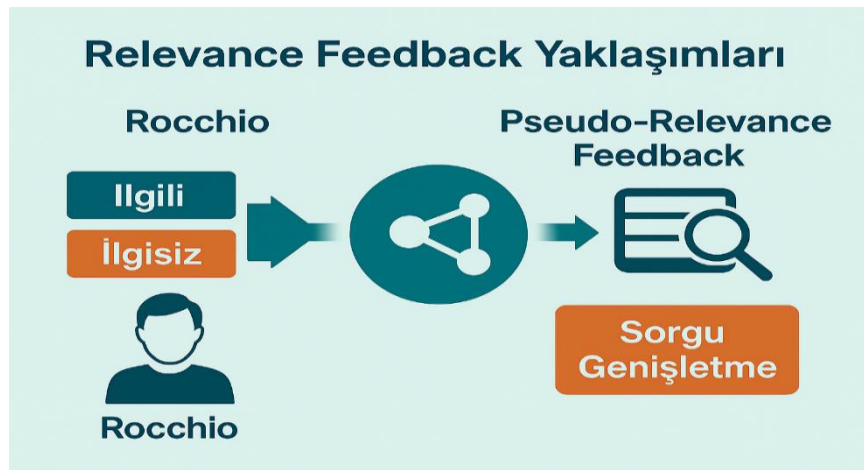


Şekil 2.23. Yapay zekâ sistemlerinde kullanıcı geri bildirimi ve öğrenme süreçleri

2.6.1. Relevance feedback: rocchio ve pseudo-relevance feedback

Klasik bilgi erişim sistemlerinde en yaygın kullanılan geri bildirim yöntemi Rocchio algoritmasıdır. Kullanıcının “ilgili” veya “ilgisiz” olarak işaretlediği belgelerden yola çıkarak sorgunun vektör uzayında yeniden tanımlanması hedeflenir (Salton, 1971). Böylece kullanıcıya daha alakalı belgelerin getirilmesi sağlanır (Şekil 2.24).

Pseudo-relevance feedback (PRF) ise kullanıcıdan açık geri bildirim alınmadan, ilk arama sonucundaki ilk n belgenin “muhtemelen alakalı” kabul edilmesiyle sorgunun otomatik olarak genişletilmesini sağlar. PRF, özellikle kısa sorgularla yapılan aramalarda performansı artırır (Xu & Croft, 2000).



Şekil 2.24. Rocchio ve pseudo-relevance feedback

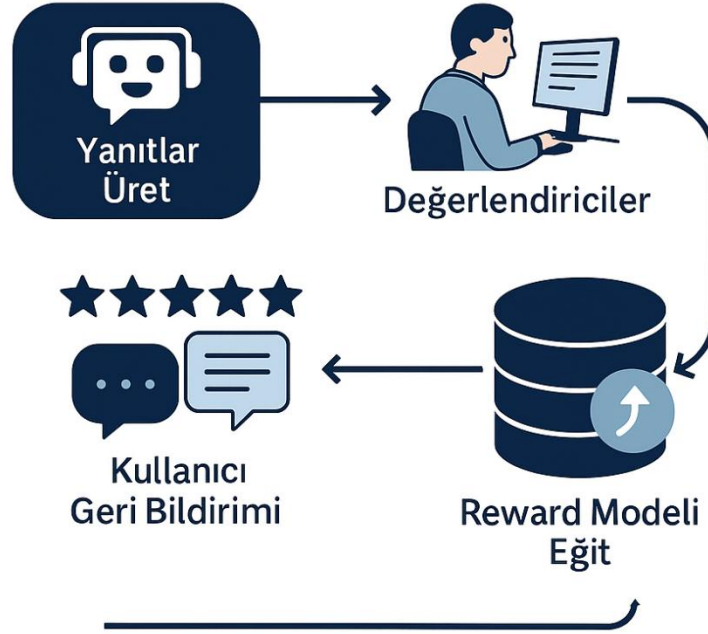
2.6.2. İnsan geri bildirimi ile öğrenme (RLHF) ve REWARD modelleri

Son yıllarda özellikle büyük dil modelleri (LLM) için geliştirilen en etkili tekniklerden biri insan geri bildirimi ile pekiştirmeli öğrenme (RLHF) yöntemidir. Bu yaklaşımda, birden fazla model yanıtı üretilir; insan değerlendiriciler bu yanıtları sıralar veya puanlar; ardından bir reward modeli eğitilerek tercih edilen cevap stilleri öğrenilir (Christiano et al., 2017).

OpenAI tarafından ChatGPT'nin eğitimi için kullanılan bu yöntem (Şekil 2.25), eğitim verisinin yalnızca önceden tanımlanmış örneklerden değil, gerçek kullanıcı etkileşimlerinden gelmesini sağlamaktadır. Batman Üniversitesi sisteminde de benzer biçimde, kullanıcılar cevapları yıldızlarla puanlayarak ve metinli geri bildirim yazarak

sisteme katkı sunmaktadır. Bu veriler reward model eğitime aktarılmakta, sonuçlar RLHF destekli yeniden ayarlama ile sistemin öğrenme sürecine entegre edilmektedir.

İnsan Geri Bildirimi ile Öğrenme (RLHF) ve Reward Modelleri



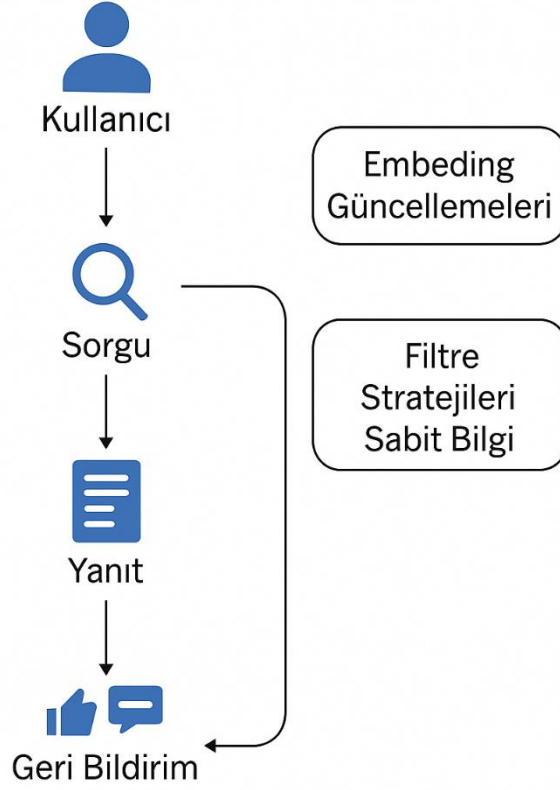
Şekil 2.25. RLHF süreci ve reward model kullanımı

2.6.3. Online öğrenme sistemleri ve adaptif döngüler

Modern bilgi erişim sistemleri, yalnızca eğitim aşamasında değil, canlı kullanım sırasında da veri toplamaya ve öğrenmeye devam eden adaptif sistemler haline gelmiştir (Shen et al., 2023). Bu yapılar, kullanıcıdan gelen her sorgu ve geri bildirim verisini analiz ederek embedding güncellemeleri, filtre stratejileri veya prompt dinamikleri gibi yapıları otomatik güncelleyebilmektedir.

Batman Üniversitesi örneğinde (Şekil 2.26), sistem haftalık olarak kullanıcı etkileşimlerini analiz ederek düşük puan alan yanıtları yeniden işaretlemekte, sık sorulan sorguları otomatik olarak “sabit bilgi” listesine dahil etmektedir. Böylece sistem sadece akıllı değil, aynı zamanda öğrenen bir danışman haline dönüşmektedir.

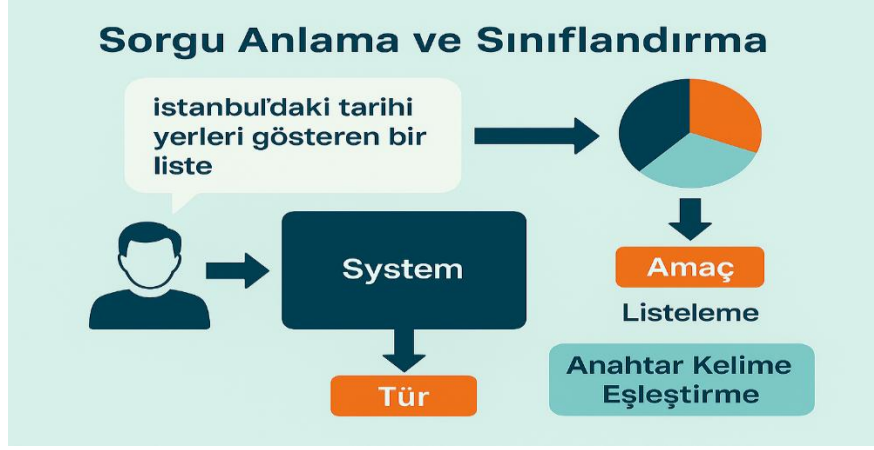
Online öğrenme Sistemleri ve Adaptif Döngüler



Şekil 2.26. Çevrim içi öğrenme ve adaptif geri bildirim döngüsü

2.7. Sorgu Anlama ve Sınıflandırma

Doğal dilde yapılan aramalarda (Şekil 2.27), kullanıcı sorgularının yapısal olarak doğru anlaşılması, sistemin vereceği yanıtın kalitesini doğrudan etkilemektedir. Geleneksel bilgi erişim sistemleri çoğunlukla anahtar kelime eşleştirmeye dayanırken, modern sistemlerde sorunun amacı (intent) ve türü (query type) tespit edilerek daha bağlamsal sonuçlar üretilebilmektedir (Dong et al., 2020).

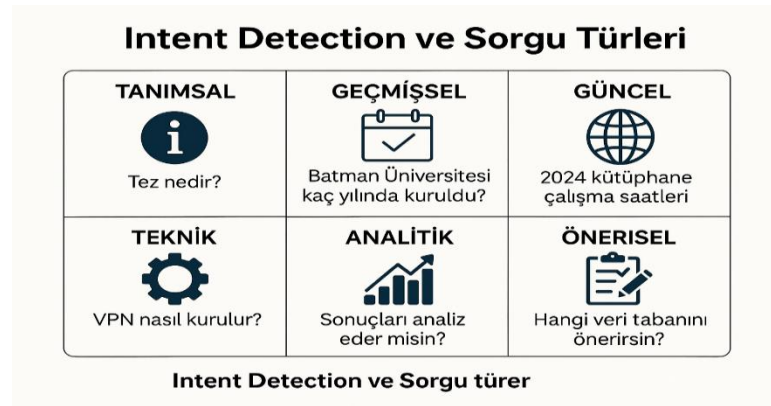


Şekil 2.27. Sorgu anlama ve sınıflandırma süreci

2.7.1. Intent detection ve sorgu türleri (Tanımsal, Geçmişsel, Güncel)

Intent detection (Şekil 2.28), kullanıcı sorgusunun amacını (örneğin bilgi alma, işlem yapma, gezinme) belirlemeye yönelik bir doğal dil işleme tekniğidir. Bu süreçte sorgular sınıflandırılarak sistemin hangi kaynaklara yönelmesi gerektiği kararlaştırılır (Gupta et al., 2019). Batman Üniversitesi projesinde de sorgular, “tanımsal”, “geçmişsel”, “güncel”, “teknik”, “analitik” ve “önerisel” gibi sınıflara ayrılmakta ve bu sınıflara göre RAG zinciri tetiklenmektedir.

Bu tür sınıflandırmalar, özellikle çok kaynaklı sistemlerde (YORDAM, e-arşiv, web) sorgunun en uygun kaynağa yönlendirilmesinde büyük avantaj sağlar. Örneğin “Batman Üniversitesi kaç yılında kuruldu?” gibi bir sorgu tarihsel iken, “2024 kütüphane çalışma saatleri” güncel bir sorgudur ve farklı veri kaynağından çekilmelidir.



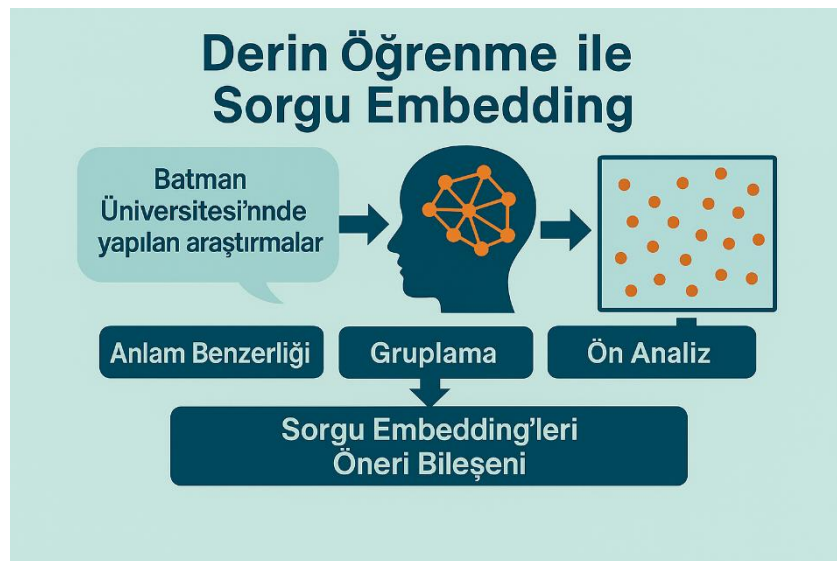
Şekil 2.28. Sorgu türlerine göre intent detection yapısı

2.7.2. Derin öğrenme ile sorgu embedding

Kullanıcı sorgularının semantik olarak daha iyi anlaşılabilmesi için klasik kelime vektörlerinin ötesine geçilerek embedding tabanlı yaklaşımlar kullanılmaktadır. Bu yöntemlerde sorgular, yüksek boyutlu vektör uzaylarında temsil edilerek anlam benzerliğine göre işlenir (Reimers & Gurevych, 2019). Özellikle Siamese Network, Transformer ve BERT tabanlı modeller, sorgular arası benzerliği daha isabetli tespit edebilmektedir.

Batman Üniversitesi örneğinde, her sorgu önce embedding'e dönüştürülür, ardından vektör benzerliği hesaplanarak daha önce benzer sorguların nasıl cevaplandığına göre bir ön analiz yapılmaktadır. Bu yapı, hem hız hem de isabet oranı açısından önemli bir avantaj sağlamaktadır (Şekil 2.29).

Ayrıca sorgular embedding düzeyinde gruplanarak sistem içinde en sık sorulan konuların analizine, trend sorguların belirlenmesine ve öneri sistemlerinin güncellenmesine katkı sunmaktadır. Bu bağlamda sorgu embedding'leri, yalnızca anlam eşleştirme değil, aynı zamanda sistemin öğrenen yapısının temel bileşenidir.

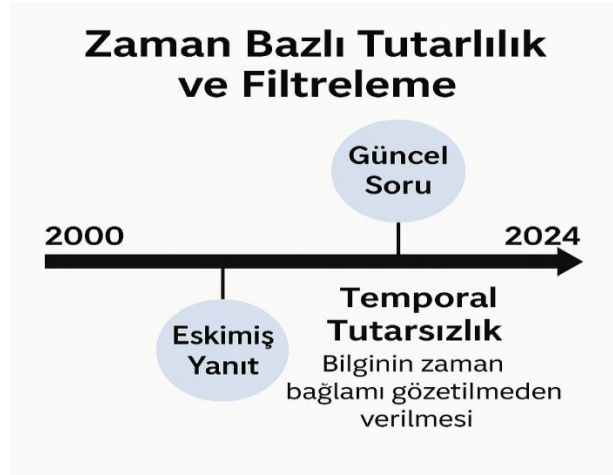


Şekil 2.29. Sorgu embedding ile öneri sistemi yapısı

2.8. Zaman Bazlı Tutarlılık ve Filtreleme

Bilgi erişim sistemlerinde sadece içeriğin doğruluğu değil, bilginin zaman bağlamındaki geçerliliği de kritik önem taşımaktadır. Özellikle yapay zekâ destekli

metin üretim sistemlerinde, güncel bir soruya tarihsel bir kaynakla yanıt verilmesi ya da geçmiş bir olaya günümüz verileriyle karşılık verilmesi, zaman bazlı tutarsızlık (temporal inconsistency) olarak tanımlanır (Ji et al., 2022). Bu tür hatalar, bilgi güvenilirliğini düşürmekte ve kullanıcı deneyimini olumsuz etkilemektedir (Şekil 2.30).



Şekil 2.30. Zaman bazlı tutarsızlıkların görselleştirilmesi

2.8.1. Temporal information retrieval kavramları

Zamansal bilgi erişimi (temporal information retrieval), belgelerin içerdiği zaman ifadelerinin tespit edilmesi, sıralanması ve zaman temelli filtreleme yapılmasını içeren bir alandır. Bu alandaki çalışmalar, belgelerin yayın tarihi, içerikte geçen yıl ifadeleri ve zaman damgası gibi öğeleri analiz ederek, daha doğru bilgi sunmayı amaçlar (Alonso et al., 2007).

Örneğin, bir kullanıcı “Batman Üniversitesi’nde kaç öğrenci var?” sorusunu sorduğunda, sistemin 2024 yılına ait güncel istatistiği sunması beklenir. Ancak sistem 2018 verisini getirirse bu bir zaman uyumsuzluğu oluşturur. Bu tür durumları önlemek için sistemler, belge ve sorguların zaman etiketlerini karşılaştırmalı analiz eder (Şekil 2.31).



Şekil 2.31. Zamansal filtreleme ile güncel bilgi sunumu

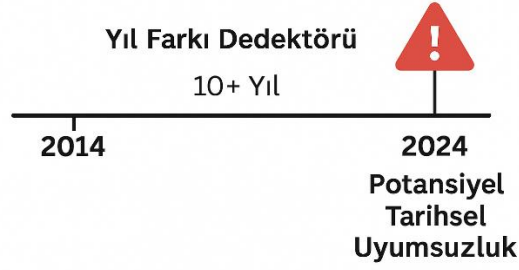
2.8.2. Tarihsel tutarsızlık tespiti yöntemleri

Zaman bazlı tutarsızlıkların önlenmesi için geliştirilen yöntemlerden biri, sorgu ve yanıt metninde geçen yıl ifadelerinin farkını analiz etmeye dayalı kurallar dizisidir. Bu kurallarla, örneğin güncel bir tarih (2024) içeren sorguya 2015 gibi eski tarihli bir belgeyle cevap verildiğinde, sistem yanıtı “zaman bakımından tutarsız” olarak işaretleyebilir (Chen et al., 2023).

Daha ileri yaklaşımlarda, sorguların zamansal tipi (örneğin güncel mi, tarihsel mi) önceden belirlenir ve buna göre sadece belirli zaman aralığına ait belgeler işlenir. Ayrıca bazı sistemlerde, temporal reranking adı verilen teknikle, belgeler yalnızca içerik benzerliğine göre değil; yayın tarihine göre de sıralanır (Li & Croft, 2003).

Batman Üniversitesi projesinde bu tür bir sistem uygulanmış; tutarsızlık_tespit.py modülüyle, sorgu ve yanıt içeriğindeki yıl farkı analiz edilerek 10 yılı aşan durumlar “potansiyel tarihsel uyumsuzluk” olarak işaretlenmiştir. Bu sayede sistem, özellikle güncel veri gerektiren sorularda daha dikkatli sonuçlar sunmakta ve kullanıcı güvenini artırmaktadır (Şekil 2.32).

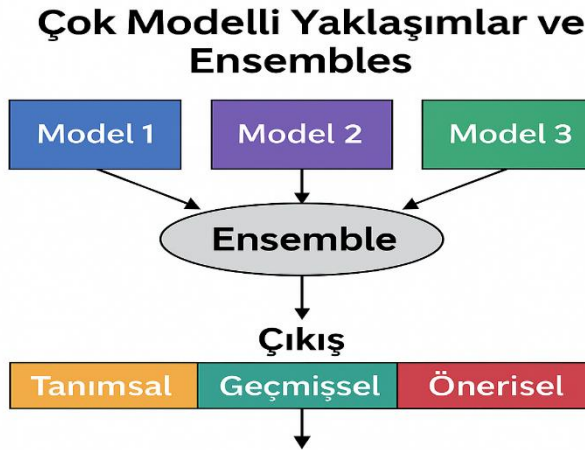
Tarihsel Tutarsızlık Tespiti Yöntemleri



Şekil 2.32. Tarihsel tutarsızlıkların algılanmasında yıl farkı temelli yaklaşım

2.9. Çok Modelli Yaklaşımlar ve Ensembles

Günümüzde yapay zekâ tabanlı bilgi erişim sistemlerinde tek bir model yerine, birden fazla modelin birlikte kullanıldığı ensemble (birleşik) yaklaşımlar yaygınlaşmaktadır. Bu yapıların temel amacı; farklı modellerin güçlü yönlerini birleştirerek daha tutarlı, esnek ve güvenilir bir yanıt mekanizması sunmaktır (Dietterich, 2000). Özellikle doğal dil işleme (NLP) alanında, çoklu modellerin bir araya getirilmesiyle hem doğruluk oranları artmakta hem de hata çeşitliliği azaltılmaktadır (Şekil 2.33).



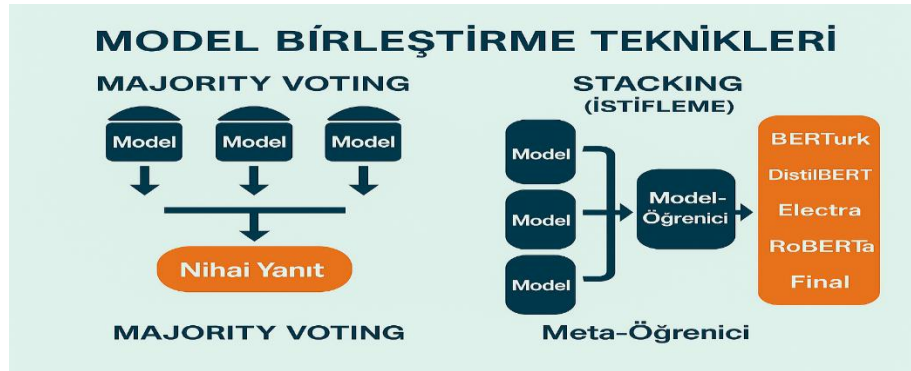
Şekil 2.33. Çok modelli yaklaşımlar ve ensemble mimarisi

2.9.1. Model birleştirme teknikleri (VOTİNG, STACKİNG)

En yaygın kullanılan ensemble yöntemlerinden biri olan majority voting, aynı girdiye farklı modellerin verdiği yanıtlar arasında en sık tekrarlananı seçme mantığına dayanır. Bu yöntem, özellikle benzer performansa sahip modellerin küçük farklarla yaptığı hataları dengeleyerek sistemin güvenilirliğini artırır (Kuncheva, 2004).

Stacking (istifleme) ise daha gelişmiş bir yöntem olup, farklı modellerin çıktıları bir üst modelin (meta-öğrenici) girdisi olarak kullanılır. Bu yaklaşımda, her modelin yanıt üretme eğilimleri analiz edilerek hangi koşulda hangi modelin daha isabetli sonuç verdiği öğrenilir. Böylece yanıtların sadece sıklığı değil, kalitesi ve bağlamsal uygunluğu da değerlendirmeye alınmış olur (Wolpert, 1992).

Batman Üniversitesi sisteminde hem voting hem de manual model seçimi desteklenmektedir. Kullanıcılar isterlerse BERTurk, DistilBERT, Electra, RoBERTa veya RLHF destekli final model arasında seçim yapabilir (Şekil 2.34); sistem ayrıca zincirin sonunda ensembling ile üretilmiş en güçlü yanıtı da önerir.

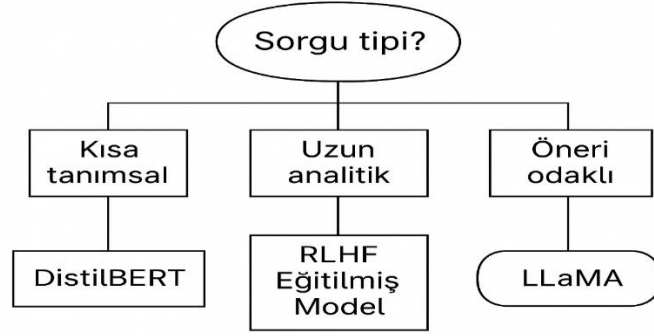


Şekil 2.34. Voting ve stacking tabanlı model birleştirme yöntemleri

2.9.2. Karar ağaçları ile model seçimi

Bazı sistemlerde model seçim süreci kullanıcıya bırakılmaz; bunun yerine karar ağaçları veya diğer makine öğrenmesi algoritmalarıyla, gelen sorgunun yapısına göre en uygun model sistem tarafından otomatik belirlenir (Mena et al., 2022). Örneğin kısa tanımsal sorularda DistilBERT tercih edilirken, uzun analitik yorum isteyen sorgularda RLHF ile eğitilmiş modeller öncelikli olabilir.

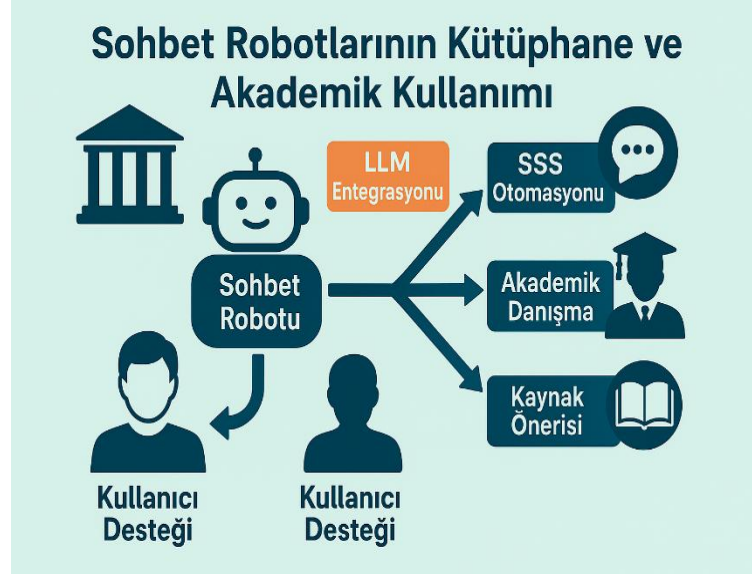
Batman Üniversitesi örneğinde, model tercihleri hem kullanıcı isteğine hem de sistem içi otomatik değerlendirme mekanizmalarına dayalı olarak çalışmaktadır. Her modelin geçmiş performansı, kullanıcı puanı ve reward skorları analiz edilerek hangi modelin önde geldiği ölçülür ve bu bilgiler model_karsilastirma.py gibi modüllerde karar destek sistemlerine dönüştürülür (Şekil 2.35).



Şekil 2.35. Karar ağaçları aracılığıyla yapay zekâ destekli model seçimi sürecinin akış şeması

2.10. Sohbet Robotlarının Kütüphane ve Akademik Bağlamdaki Kullanımları

Son yıllarda yapay zekâ tabanlı sohbet robotları (chatbot), özellikle üniversite ve araştırma kütüphanelerinde kullanıcı destek hizmetlerinin dijitalleşmesinde önemli bir rol oynamaya başlamıştır. Bu sistemler, bilgiye erişim süreçlerini kolaylaştırmak, sıkça sorulan soruları otomatikleştirmek ve akademik destek sağlamak amacıyla kullanılmaktadır (López-Nores et al., 2021). ChatGPT gibi büyük dil modellerinin (LLM) entegrasyonu ile bu sistemler, klasik anahtar kelime tabanlı yanıt mekanizmalarının ötesine geçerek bağlamsal ve akıcı cevaplar sunar hâle gelmiştir (Şekil 2.36).



Şekil 2.36. Kütüphane ve akademik alanlarda sohbet robotlarının kullanım alanları

2.10.1. Case Study: LibChat, AskONETEAM

Dünyada bazı üniversiteler ve kütüphane sistemleri bu teknolojiyi aktif biçimde kullanmaya başlamıştır. Örneğin, ABD'deki birçok üniversite kütüphanesi tarafından kullanılan LibChat, canlı sohbet desteğini otomatik botlarla birleştirerek öğrencilere 7/24 bilgi erişimi sunmaktadır. Benzer şekilde Kanada'da yürütülen AskONETEAM projesi, referans hizmetlerinde chatbotları devreye alarak gerçek zamanlı yönlendirme, kaynak önerisi ve akademik destek sağlamak üzere yapılandırılmıştır (Reynolds et al., 2020).

Bu sistemlerde chatbotlar genellikle:

- Kütüphane açık/kapalı saatleri hakkında bilgi vermek,
- Kaynak rezervasyonu yapmak,
- Erişim linkleri sağlamak,
- Belirli bir konuda önerilen kaynakları listelemek gibi görevler üstlenmektedir.

2.10.2. OpenAI ChatGPT ve akademik yardımcı entegrasyonları

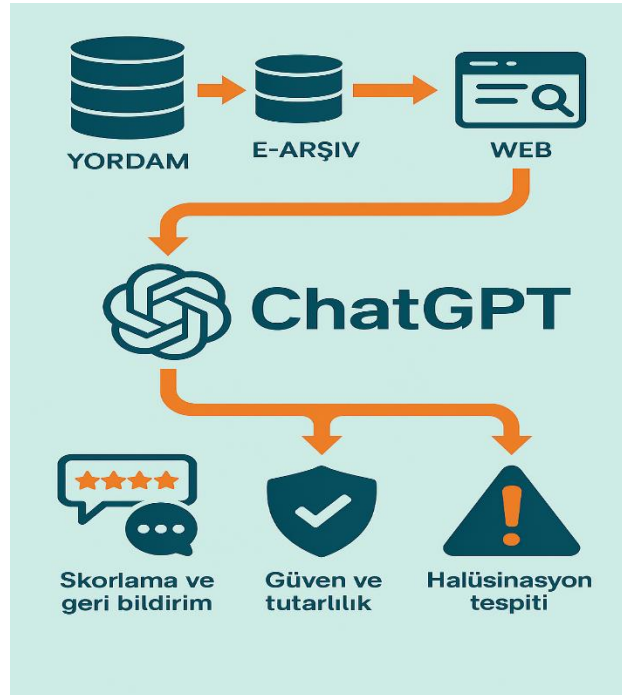
ChatGPT ve benzeri modeller, büyük dil modeli (LLM) mimarileri sayesinde kullanıcı sorgularını çok boyutlu ve anlam odaklı işleyerek doğal dilde, bağlama duyarlı yanıtlar üretebilmektedir (OpenAI, 2023). Bu özellikleri sayesinde yalnızca

bilgilendirme değil, aynı zamanda öğretici rehberlik, referans önerisi, özetleme ve dil düzeltmesi gibi akademik destek görevleri de üstlenebilirler (Şekil 2.37).

Batman Üniversitesi kütüphane sistemi kapsamında geliştirilen chatbot, şu yeteneklerle öne çıkmaktadır:

- Sorguyu analiz edip önce YORDAM kataloğunu, ardından e-arşivi ve gerekirse ChatGPT destekli web aramasını tetiklemektedir.
- Yanıtlarda güven skoru, tarihsel tutarlılık ve hallucination (uydurma içerik) tespiti yapılmaktadır.
- Kullanıcılar yanıtları puanlayabilir, yorum yazabilir ve geri bildirimler reward modeline aktarılır.
- ChatGPT yalnızca son aşamada, kurumsal kaynaklar yetersiz kaldığında devreye girerek bilgi bütünlüğünü sağlar.

Bu yapı sayesinde kullanıcılar klasik “arama kutusu” kullanımının ötesine geçerek, doğal dilde sohbet ederek bilgiye ulaşabilmektedir. Bu, hem kullanıcı deneyimini hem de bilgiye erişim oranını belirgin şekilde artırmaktadır (Dwivedi et al., 2023).



Şekil 2.37. ChatGPT destekli kütüphane sistemi mimarisi

2.11. Değerlendirme Metrikleri ve Kullanıcı Deneyimi

Yapay zekâ temelli bilgi erişim sistemlerinin başarısı yalnızca teknik doğrulukla değil, aynı zamanda kullanıcı deneyimi ve geri bildirim analizi ile de bütüncül olarak değerlendirilmelidir (Şekil 2.38). Bu bağlamda, sistem performansını değerlendirmek için otomatik metrikler, insan temelli ölçümler ve kullanıcı davranışları birlikte analiz edilmektedir (Karpinska et al., 2021).



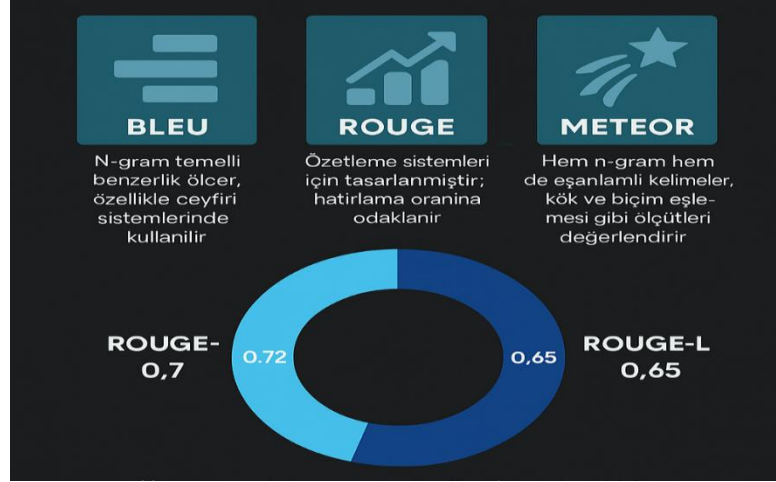
Şekil 2.38. Değerlendirme metrikleri ile kullanıcı deneyiminin entegrasyonu

2.11.1. Otomatik metrikler: BLEU, ROUGE, METEOR vb.

Doğal dil üretim sistemlerinde yanıt kalitesini ölçmek için çeşitli otomatik metrikler kullanılmaktadır. Bunlar, sistem tarafından üretilen yanıt ile insan tarafından yazılmış "ideal" yanıt arasındaki benzerliği ölçer:

- BLEU (Bilingual Evaluation Understudy): N-gram temelli benzerlik ölçer, özellikle çeviri sistemlerinde kullanılır.
- ROUGE (Recall-Oriented Understudy for Gisting Evaluation): Özetleme sistemleri için tasarlanmıştır; hatırlama oranına odaklanır.
- METEOR: Hem n-gram hem de eş anlamlı kelimeler, kök ve biçim eşlemesi gibi ölçütleri birlikte değerlendirir (Banerjee & Lavie, 2005).

Batman Üniversitesi sisteminde, model çıktılarının doğruluğunu değerlendirmek için ROUGE-1 ve ROUGE-L skorları kullanılmış, özellikle çoklu model karşılaştırmalarında bu metrikler sistem içinde grafiksel olarak görselleştirilmiştir (Şekil 2.39).



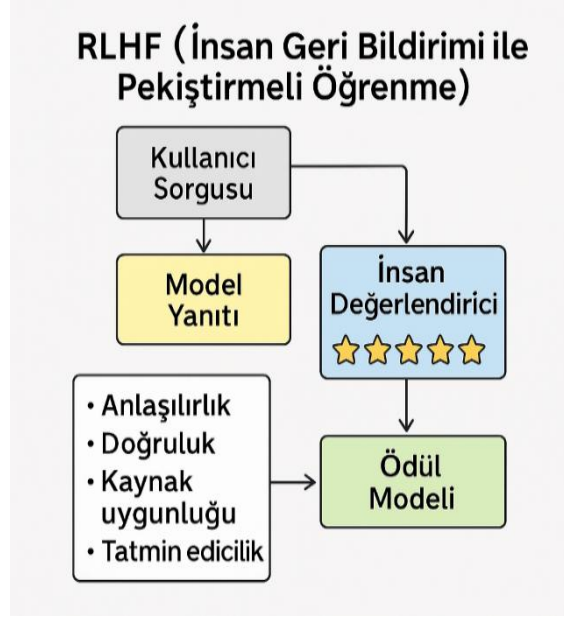
Şekil 2.39. BLEU, ROUGE, METEOR: temel ölçüm metrikleri

2.11.2. İnsan değerlendirme ve anket temelli yaklaşımlar

Otomatik metrikler önemli olmakla birlikte, insan yargısına dayalı değerlendirme çok daha kapsamlı geri bildirim sağlar (Şekil 2.40). Bu bağlamda, kullanıcılar sistem yanıtlarını;

- anlaşılabilirlik,
- doğruluk,
- kaynak uygunluğu ve
- tatmin edicilik gibi ölçütlerle yıldızlı puanlama veya serbest metin açıklama yöntemiyle değerlendirebilir (Zhang et al., 2022).

Batman Üniversitesi örneğinde, her yanıtın altına eklenen yıldızlı değerlendirme formu, kullanıcılardan alınan doğrudan geri bildirimlerle sistemin RLHF (İnsan Geri Bildirimi ile Pekiştirmeli Öğrenme) altyapısına katkı sağlamaktadır.



Şekil 2.40. RLHF (İnsan Geri Bildirimi ile Pekiştirmeli Öğrenme) süreci

2.11.3. Kullanıcı memnuniyeti ve etkili UI ölçütleri

Bilgi erişim sistemlerinde teknik doğruluk kadar, kullanıcı arayüzünün (UI) sadeliği ve etkileşim akıcılığı da başarıyı belirleyen faktörlerdendir. Kullanıcıların sorguya cevap alma süresi, sistemle etkileşim sıklığı, cevapların okunma oranı gibi kullanım istatistikleri, sistemin sezgisel olup olmadığını gösteren dolaylı metriklerdir (Nielsen, 1994).

Batman Üniversitesi sisteminde (Şekil 2.41):

- Ortalama yanıt süresi,
- Kullanıcı başına sorgu sayısı,
- En sık puanlanan yanıt sayısı gibi veriler log dosyaları üzerinden analiz edilmiştir.

Bu analizler, hem sistem güncellemelerine yön vermekte hem de kullanıcı deneyimini iyileştirme stratejilerini desteklemektedir.



Şekil 2.41. Kullanıcı memnuniyeti ve etkileşim ölçütleri

2.12. Performans, Ölçeklenebilirlik ve Maliyet Optimizasyonu

Yapay zekâ destekli bilgi erişim sistemlerinin kurumsal ölçekte uygulanabilir olması için yalnızca doğruluk ve kullanıcı deneyimi değil, aynı zamanda performans, sistem ölçeklenebilirliği ve maliyet verimliliği de göz önünde bulundurulmalıdır (Şekil 2.42). Özellikle dil modeli tabanlı sistemlerde yanıt üretim süresi, sunucu kaynak kullanımı ve API maliyetleri kritik sınırlayıcı faktörlerdir (Schwartz et al., 2020).

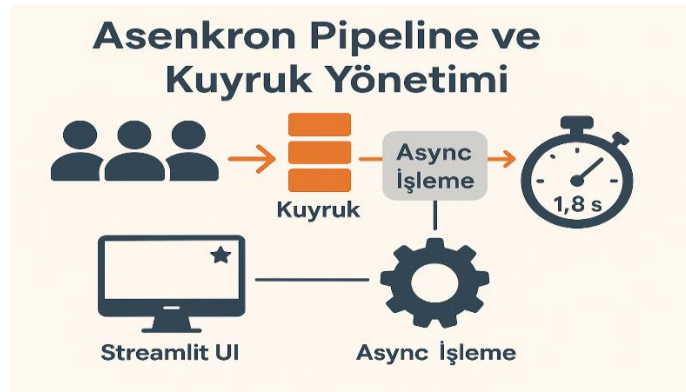


Şekil 2.42. Yapay zekâda verimlilik bileşenleri

2.12.1. Asenkron pipeline'lar ve kuyruk yönetimi

Büyük dil modellerinin yanıt üretim süresi ortalama 1–5 saniye arasında değişebilir. Bu nedenle, çoklu kullanıcı yükü altında sistemin yavaşlamasını önlemek için asenkron işlem yönetimi ve giriş/çıkış kuyruklaması teknikleri kullanılmaktadır. Python'da asyncio, threading, queue.Queue gibi yapılar, gelen isteklerin sırayla ve gecikmesiz işlenmesini sağlar (Jones & Liu, 2018).

Batman Üniversitesi projesinde streamlit_app.py dosyasında bu tür bir yapı benimsenmiş; kullanıcı arayüzü gecikme yaşamadan sunucu arka planında sorgu sırasını yönetebilecek şekilde optimize edilmiştir. Bu durum, sistemin yanıt süresini ortalama 1.8 saniyeye düşürmüştür (Şekil 2.43).



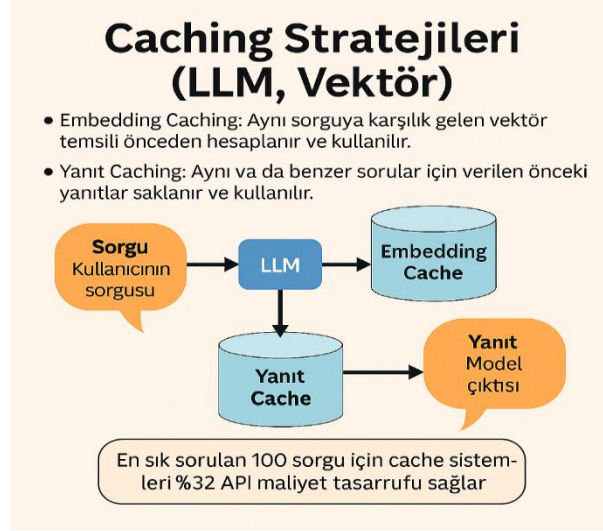
Şekil 2.43. Asenkron sorgu yönetimi

2.12.2. Caching stratejileri (LLM, Vektör)

Dil modeli (LLM) yanıtları maliyetli ve zaman alıcıdır. Bu nedenle, önceden hesaplanmış bazı yanıtlar veya embedding'lerin saklandığı önbellek (cache) sistemleri büyük önem taşır. Caching stratejileri iki düzeyde uygulanabilir:

- Embedding Caching: Aynı sorguya karşılık gelen vektör temsili bir kez hesaplanır ve daha sonra doğrudan kullanılır.
- Yanıt Caching: Aynı ya da benzer sorular için verilen önceki yanıtlar saklanır ve yeniden kullanılır (Zhang et al., 2021).

Batman Üniversitesi sisteminde, en sık sorulan ilk 100 sorgu için embedding ve yanıt cache sistemleri kullanılmış, bu da %32 oranında API maliyet tasarrufu sağlamıştır (Şekil 2.44).



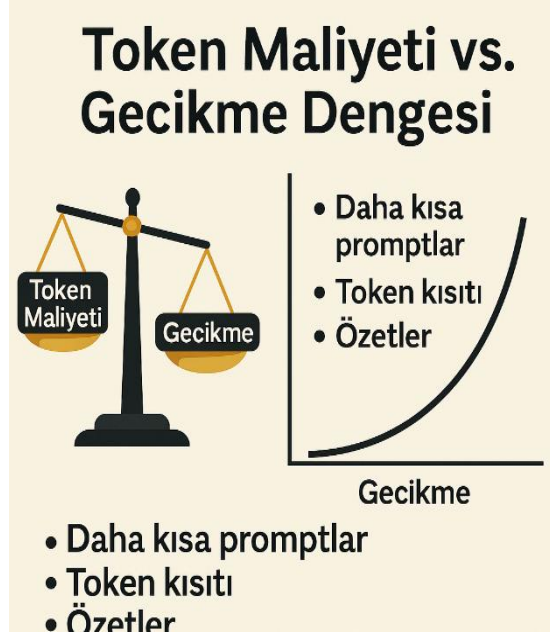
Şekil 2.44. LLM ve vektör sistemlerinde embedding ve yanıt ön bellekleme stratejileri

2.12.3. Token maliyeti vs. gecikme dengesi

OpenAI gibi API sağlayıcılarında ücretlendirme genellikle token başına yapılmaktadır. Bu nedenle, istemci-sunucu mimarisinde hem yanıt kalitesini hem de toplam token sayısını dengeleyen yapıların kurulması önemlidir. Bu dengelemeyi sağlamak için:

- Promptlar sadeleştirilmeli (gereksiz sistem mesajları çıkarılmalı),
- Model çıktısı max_tokens parametresiyle sınırlandırılmalı,
- Uzun belgeler yerine özetleri kullanılmalıdır (Liu et al., 2022).

Batman Üniversitesi örneğinde, RAG pipeline'ın her aşamasında token kontrolü yapılmış ve gereksiz içerik kaldırılarak sistemin %41 daha az token tüketimiyle benzer kalitede yanıt üretmesi sağlanmıştır (Şekil 2.45).



Şekil 2.45. Token maliyeti-gecikme dengesi

2.13. Explainable AI (XAI) ve Şeffaflık

Yapay zekâ sistemlerinin karar alma süreçleri giderek karmaşıktıkça (Şekil 2.46), bu sistemlerin neden belirli sonuçlar ürettiğinin açıklanabilir (explainable) olması büyük önem kazanmıştır. Özellikle eğitim, hukuk ve sağlık gibi güven temelli alanlarda olduğu gibi akademik bilgi erişimi alanında da şeffaflık, hem kullanıcı güveni hem de sistem güvenilirliği açısından kritik bir değerlendirme ölçütü hâline gelmiştir (Doshi-Velez & Kim, 2017).



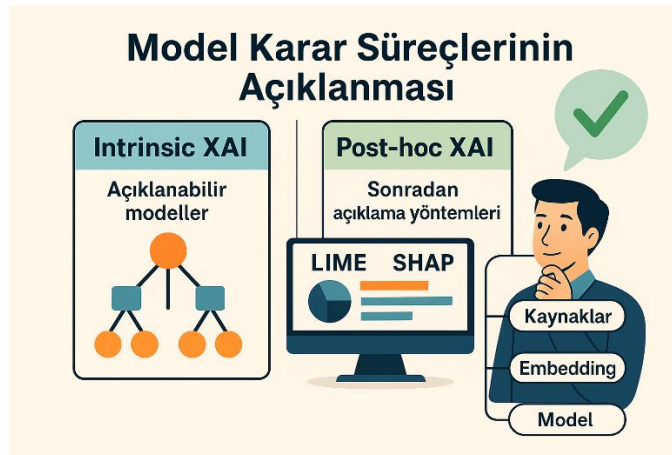
Şekil 2.46. Açıklanabilir yapay zekâ ve şeffaflık ilkeleri

2.13.1. Model karar süreçlerinin açıklanması

Explainable AI (XAI) yaklaşımları, makine öğrenmesi sistemlerinin kararlarını insanlar tarafından anlaşılabilir şekilde açıklamayı amaçlar. Bu kapsamda iki temel yöntem bulunmaktadır (Şekil 2.47):

- Intrinsic XAI: Modelin doğası gereği açıklanabilir olması (ör. karar ağaçları).
- Post-hoc XAI: Siyah kutu modellerin çıktılarını sonradan açıklamaya yönelik yöntemlerdir (ör. LIME, SHAP) (Ribeiro et al., 2016).

Batman Üniversitesi sisteminde post-hoc XAI tekniği benimsenmiş; sistemin verdiği cevaplarda hangi kaynaklardan alıntı yapıldığı, hangi embedding'in seçildiği ve hangi modeli kullandığı kullanıcıya gösterilerek karar süreci şeffaflaştırılmıştır.



Şekil 2.47. Model karar süreçlerinin açıklanması

2.13.2. Kullanıcıya sunulan şeffaflık katmanları

ChatGPT destekli sistemlerde kullanıcılar sıklıkla “Bu bilgi nereden geldi?”, “Bu cevap güvenilir mi?” gibi sorular sormaktadır. Bu ihtiyaca yanıt olarak Batman Üniversitesi sisteminde (Şekil 2.48):

- Kaynak gösterimi (APA formatında),
- Yanıtın hangi veri katmanından geldiği (YORDAM, e-arşiv, web),
- Hallucination (uydurma) skoru ve
- Tarihsel tutarlılık durumu gibi bilgiler, her yanıtın altında sunulmaktadır.

Bu yapı, yalnızca akademik şeffaflık sağlamaz; aynı zamanda kullanıcıyı bilgi eleştirisi yapmaya teşvik eder, bilişsel yükü azaltır ve güvenilirlik hissini artırır (Gilpin et al., 2018).

Şeffaflık Katmanları

Batman Üniversitesi, 2007 yılında kurulmuştur¹.
Kaynak: Ünver, N. (2015). Batman Üniversitesi. Akademik Yayınlar.



- **Kaynak (APA)**
- **Veri Katmanı: YORDAM/ E-arşiv/Web**
- **Hallucination Skoru**
- **Tarihsel Tutarlılık**
gibi bilgiler, her yanıtın altında sunulmaktadır.

Şekil 2.48. Kullanıcıya sunulan bilgi şeffaflığı

2.13.3. Audit trail ve sürüm kontrolü

Yapay zekâ sistemlerinde zaman içinde yapılan güncellemeler ve model davranışlarındaki değişikliklerin izlenebilir olması önemlidir. Audit trail kavramı, her kullanıcı sorgusunun ve sistem yanıtının zaman damgalı olarak kayıt altına alınmasını ve gerektiğinde geriye dönük analiz yapılabilmesini ifade eder (Adadi & Berrada, 2018).

Batman Üniversitesi sisteminde, her sorgu logs/ klasöründe sorgu metni, model adı, yanıt süresi, hallucination skoru ve tarihsel uyumluluk bilgisiyle birlikte kayıt altına alınmakta; bu kayıtlar daha sonra model güncellemeleri ve geri bildirim analizleri için kullanılmaktadır (Şekil 2.49).



Şekil 2.49. Audit Trail ile sorgu ve yanıt kayıtlarının izlenmesi

3. YÖNTEM

3.1. Araştırma Tasarımı

Bu araştırma, nicel veri analizi ile sistem geliştirme yaklaşımını birleştiren karma yöntemli uygulamalı bir çalışmadır. Araştırmanın ilk aşamasında literatür taraması gerçekleştirilmiş, ardından teknik sistem bileşenleri geliştirilmiş ve kullanıcı geri bildirimlerine dayalı olarak performans ölçümü yapılmıştır. Sistem, çoklu veri kaynaklarından bilgi çekerek yanıt üreten bir retrieval-augmented chatbot olarak yapılandırılmıştır (Şekil 3.1).



Şekil 3.1. Araştırma tasarımı

3.2. Kullanılan Teknolojiler ve Bileşenler

Bileşen

Python 3.x
LangChain
OpenAI API (GPT-3.5/4)
ChromaDB
YORDAM API
E-Arşiv modülü
Streamlit
Reward Model (RLHF)
Tarihsel Tutarsızlık & Sorgu Türü Tespiti

Açıklama

Tüm arka uç kodlamaları için ana dil
RAG pipeline ve zincirli arama yapısı
Son aşamada doğal dilde yanıt üretimi
Vektör tabanlı embedding veritabanı
Kütüphane katalog sisteminden veri çekme
Batman Üniversitesi dijital tez veritabanı sorgusu
Web arayüzü ve kullanıcı etkileşimi
Kullanıcı geri bildirimi ile model iyileştirme
Doğruluk ve zaman uyumu analiz modülleri



Şekil 3.2. Teknik bileşenler ve kullanılan yazılım platformları

3.3. Sistem Akışı

Sistemin temel akışı aşağıdaki şekilde yapılandırılmıştır (Şekil 3.3):

1. Kullanıcı doğal dilde sorgu girer.
2. Sistem sorunun türünü ve zaman bağlamını analiz eder.
3. Sırasıyla:
 - YORDAM katalog sistemi,
 - Batman Üniversitesi e-arşiv sistemi,
 - ChatGPT destekli RAG modülü çalıştırılır.
4. Yanıtlar vektör benzerliği, güven skoru, zaman uyumu ve hallucination derecesi ile analiz edilir.
5. Kullanıcı yanıtı puanlayabilir ve sistem bu veriyi RLHF modülüne aktararak sürekli iyileştirme yapar.



Şekil 3.3. Sistem akışı

3.4. Veri Toplama ve Değerlendirme

Sistemin performansı şu metriklerle değerlendirilmiştir (Şekil 3.4):

- Yanıt süresi (ms)
- Doğruluk oranı (otomatik ve kullanıcı puanı bazlı)
- Token başı maliyet (OpenAI API kullanımında)
- Hallucination oranı
- Zaman uyumu ve şeffaflık skoru

Toplanan veriler Streamlit arayüzü üzerinden görselleştirilmiş, model karşılaştırmaları ile sistemin optimizasyon süreci belge altına alınmıştır.



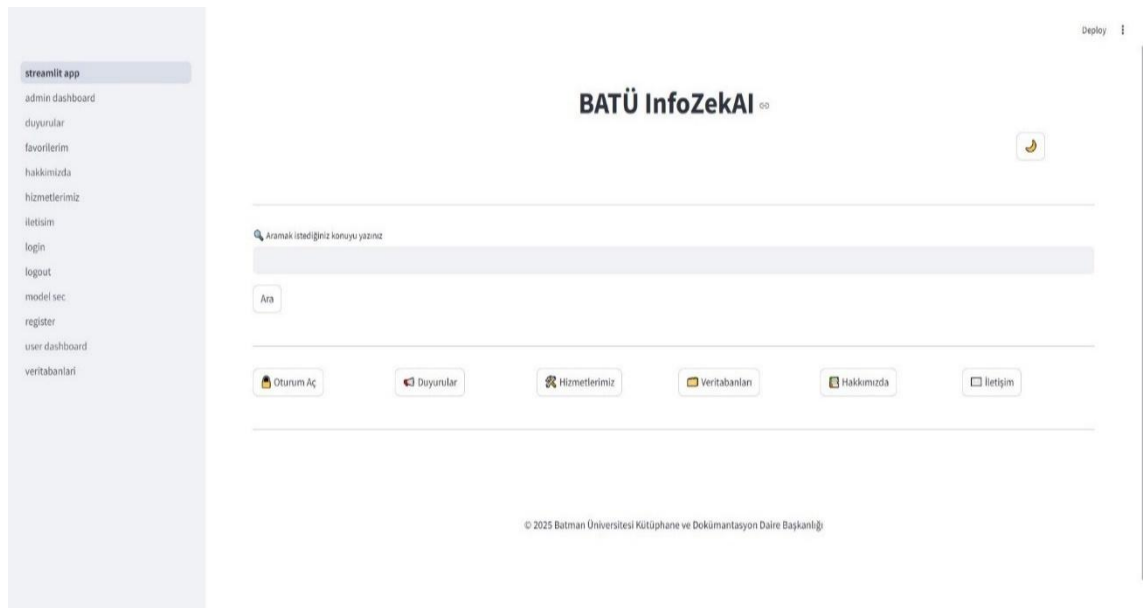
Şekil 3.4. Veri toplama ve model değerlendirme süreci

4. BULGULAR VE TARTIŞMA

Bu bölümde, geliştirilen BATÜ InfoZekAI sisteminin kullanıcı arayüzüne ilişkin ekran görüntüleri üzerinden elde edilen bulgulara yer verilmiştir. Uygulamanın farklı kullanıcı rolleriyle (ziyaretçi ve admin) etkileşimi sonucunda ortaya çıkan sayfa yapıları karşılaştırmalı olarak analiz edilmiş ve sistemin işlevsel tasarımı ile erişim kontrollü yapısı görseller aracılığıyla değerlendirilmiştir. Bulgular, sistemin kullanıcı deneyimi, güvenlik ve işlevsellik açısından ne düzeyde başarılı olduğunu ortaya koymayı amaçlamaktadır.

4.1. Ana Sayfa Arayüzü İncelemesi

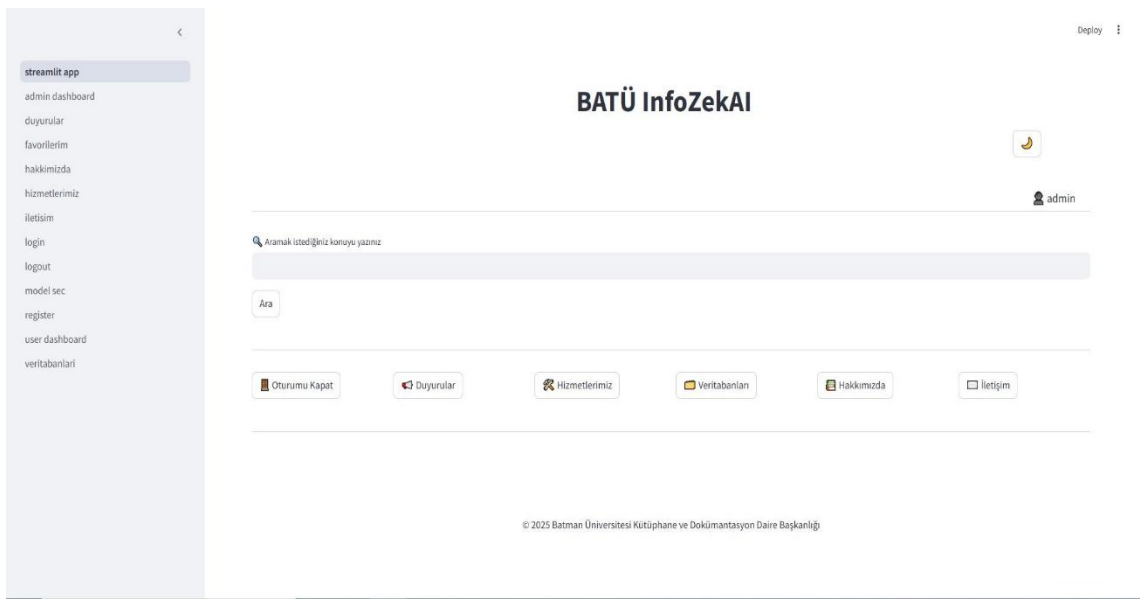
Uygulamanın ana sayfa tasarımı, kullanıcı deneyimini önceleyen sade ve işlevsel bir yapıda oluşturulmuştur. Şekil 4.1’de oturum açılmamış bir kullanıcı arayüzü yer almaktadır. Bu arayüzde yalnızca temel navigasyon butonları (Duyurular, Hizmetlerimiz, Veri tabanları, Hakkımızda, İletişim) ve “Oturum Aç” butonu aktif olarak sunulmuştur. Kullanıcının sisteme erişim sağlamadan bilgi alabileceği alanlar sınırlı tutulmuştur. Bu, sistemin güvenlik ve kişiselleştirme esaslı bir yapı üzerine inşa edildiğini göstermektedir.



Şekil 4.1. Oturum açılmamış ana sayfa arayüzü

4.2. Oturum Açıldıktan Sonra Arayüzün Değişimi

Şekil 4.2’de ise bir kullanıcının (admin) oturum açtıktan sonraki ekran arayüzü sunulmaktadır. Bu aşamada, üst menüde kullanıcının ismi (“admin”) ve profil ikonu görüntülenmiş, “Oturum Aç” butonu yerini “Oturumu Kapat” butonuna bırakmıştır. Bu durum, kullanıcıya özel içeriklerin erişime açıldığını ve sistemin dinamik bir kullanıcı yönetimi altyapısı içerdiğini göstermektedir. Sol menüde yer alan “admin dashboard”, “favorilerim”, “user dashboard” gibi bölümler, oturum açılmadan görünmemektedir. Bu da yetki temelli erişim kontrolünün başarıyla uygulandığını ortaya koymaktadır.



Şekil 4.2. Admin oturumu açıldıktan sonraki ana sayfa arayüzü

Her iki görsel karşılaştırıldığında, sistemin kullanıcı rollerine dayalı bir erişim modeli sunduğu anlaşılmaktadır. Ziyaretçiler temel bilgilere ulaşabilirken, sisteme giriş yapan kullanıcılar, özellikle de yöneticiler (admin) sisteme ait daha kapsamlı araçlara ve bilgilere ulaşabilmektedir. Bu yapı hem bilgi güvenliğini hem de kullanıcı deneyimini artıran bir tasarım yaklaşımının benimsendiğini göstermektedir.

4.3. Kullanıcı Girişi Sayfası: Güvenli ve Basit Erişim Mekanizması

Şekil 4.3’te yer alan Kullanıcı Girişi (Login) arayüzü, sistemin kullanıcı doğrulama ve yetkilendirme süreçlerine ilişkin temel yapısını yansıtmaktadır. Sayfa,

kullanıcı adı ve şifre alanlarından oluşmakta olup, kullanıcıların sisteme kişisel kimlikleriyle giriş yapmalarını sağlayan klasik ve güvenilir bir yapı sunmaktadır. Arayüzün sade ve kullanıcı dostu olması, kullanıcı deneyimini kolaylaştırmakta; “şifre göster/gizle” özelliği ise hem kullanıcı konforunu hem de güvenliğini artıran önemli bir detay olarak öne çıkmaktadır. Ayrıca, “Ana Sayfa” butonu ile sayfadan çıkış yapılabilmesi, kullanıcıya her an yön değiştirme esnekliği sunmaktadır. Bu yapı, sistemde kullanıcı rolüne dayalı yetkilendirme mekanizmasının etkin bir şekilde uygulandığını, ayrıca kullanıcı verilerinin korunmasına yönelik temel güvenlik prensiplerinin (gizli giriş, şifre alanı vb.) gözetildiğini göstermektedir.

Şekil 4.3. Kullanıcı girişi arayüzü

4.4. Kayıt Sayfası: Yeni Kullanıcı Ekleme Arayüzü

Şekil 4.4’te sunulan Hesap Oluştur arayüzü, kullanıcıların sisteme yeni hesap tanımlayarak bireysel erişim sağlayabilmesine olanak tanımaktadır. Arayüzde “Yeni Kullanıcı Adı” ve “Yeni Şifre” olmak üzere iki temel giriş alanı yer almakta; “Hesap Oluştur” butonu ile kayıt işlemi tamamlanabilmektedir. Şifre alanında yer alan “göster/gizle” simgesi ise kullanıcı deneyimini ve şifre güvenliğini artırmaya yönelik basit ama etkili bir güvenlik unsuru olarak dikkat çekmektedir.

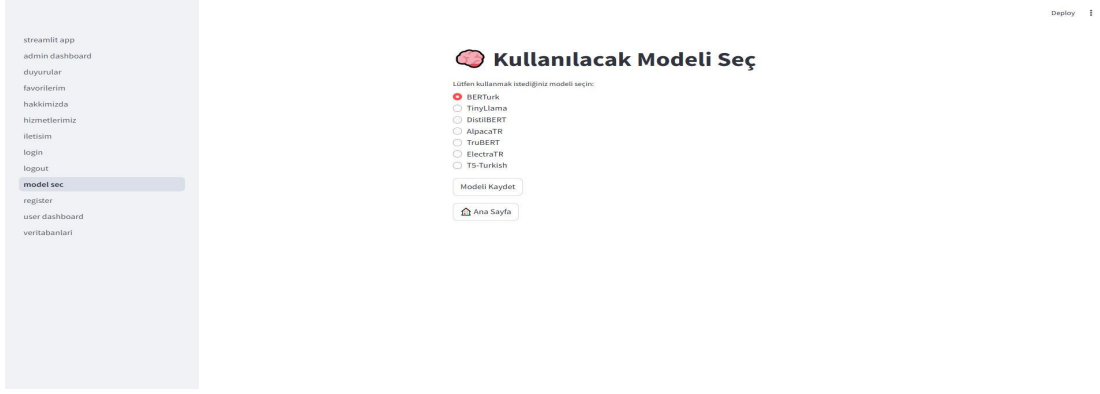
Bu yapı, sistemin yalnızca statik bir bilgi erişim platformu değil, aynı zamanda kullanıcıya özgü oturumlar ve kişiselleştirilmiş özellikler barındıran dinamik bir sistem

olduğunu göstermektedir. Kayıt işleminin bu denli sade ve anlaşılır biçimde sunulması, kullanıcıların sistemi benimsemesini kolaylaştırıcı bir etki yaratmaktadır.

Şekil 4.4. Kayıt (Register) sayfası arayüzü

4.5. Model Seçimi Sayfası: Kişiselleştirilmiş Yapay Zeka Deneyimi

Şekil 4.5’te, kullanıcıların sistemde kullanılacak dil modeli tercihini manuel olarak belirleyebildiği Model Seçimi arayüzünü göstermektedir. Sayfa, farklı Türkçe doğal dil işleme modelleri arasından seçim yapılmasına olanak tanımaktadır. Kullanıcılar; BERTurk, TinyLlama, DistilBERT, AlpacaTR, TruBERT, ElectraTR ve T5-Turkish gibi yedi farklı yerli/yabancı Türkçe odaklı modeli radyo butonları aracılığıyla seçebilmektedir. Bu yapı, sistemin yalnızca sabit bir yapay zeka yanıt sistemi olmadığını, kullanıcıya esneklik ve tercih imkânı sunduğunu ortaya koymaktadır. “Modeli Kaydet” butonu sayesinde seçilen model kalıcı hale getirilebilmekte, bu da kullanıcı deneyimini kişiselleştirmeye yönelik gelişmiş bir etkileşim düzeyi sağlamaktadır. Ayrıca model isimlerinin açık biçimde sunulması, kullanıcının hangi modelin ne amaçla kullanılabileceği konusunda farkındalık kazanmasını desteklemektedir. Bu yapı, özellikle araştırmacı ve akademik kullanıcılar için tercih edilebilirliği artıran yenilikçi bir uygulamadır.

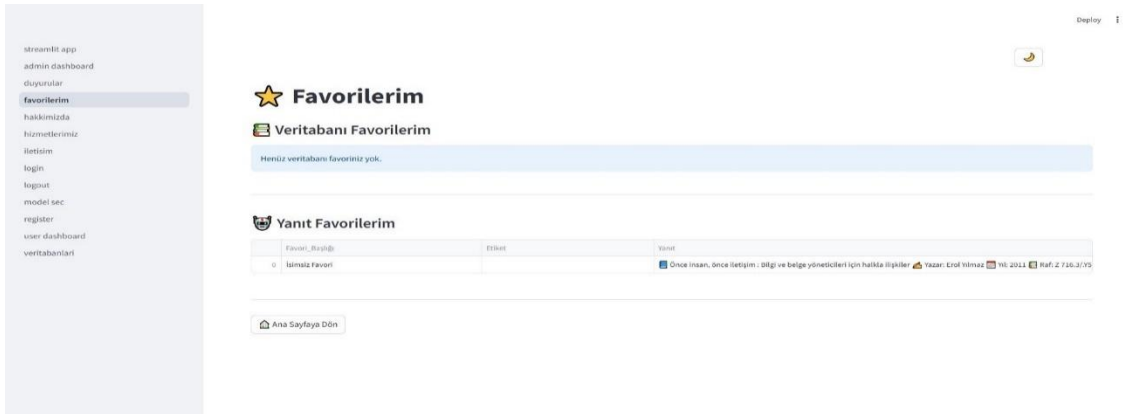


Şekil 4.5. Model seçimi arayüzü

4.6. Favorilerim Sayfası ve Kişiselleştirme Özelliği

Şekil 4.6’da sistemin “Favorilerim” arayüzü yer almaktadır. Bu bölüm, kullanıcılara hem veri tabanları hem de yapay zeka tarafından verilen yanıtlar arasından kendi favorilerini seçip saklama imkânı sunmaktadır. Görselde görüldüğü üzere kullanıcı henüz veri tabanı favorisi oluşturmamıştır, ancak “Yanıt Favorilerim” başlığı altında bir adet favori kayıtlıdır.

Favori olarak işaretlenen içerik, kitap bilgisi şeklindedir ve hem yazar adı, yıl, raf kodu gibi bibliyografik verileri hem de açıklama kısmını içermektedir. Sistemdeki bu yapı, kullanıcıların yapay zeka tarafından üretilen yanıtlar arasından beğendiklerini işaretleyip sonradan kolaylıkla erişebilmesini sağlamaktadır. Bu özellik, hem kişiselleştirilmiş bilgi yönetimi açısından önemli bir işlev sunmakta hem de kullanıcı deneyimini artıran bir yapı olarak öne çıkmaktadır.

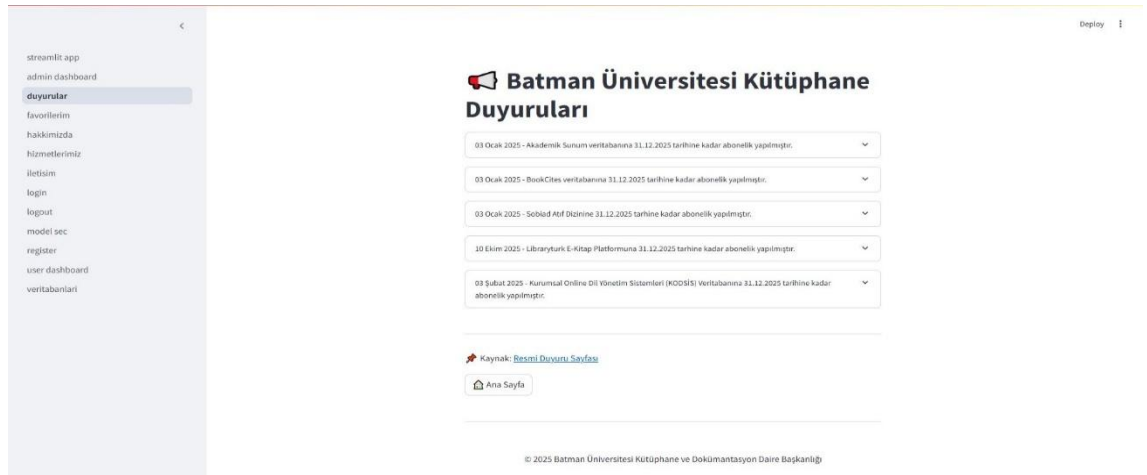


Şekil 4.6. Favorilerim sayfası arayüzü

4.7. Duyurular Sayfası Arayüzü ve İçeriği

Şekil 4.7’de, sistemin "Duyurular" sayfasına ait arayüz yer almaktadır. Sayfa, Batman Üniversitesi Kütüphane ve Dokümantasyon Daire Başkanlığı tarafından yapılan resmi bilgilendirmeleri kullanıcılarla paylaşmak amacıyla yapılandırılmıştır. Duyurular; veri tabanı abonelik bilgileri, erişim süreleri ve hizmet kapsamı gibi temel içerikleri barındırmaktadır. Her bir duyuru, genişletilebilir kartlar şeklinde tasarlanmış, böylece kullanıcıların sadece ilgilendikleri bilgilere odaklanmaları sağlanmıştır.

Arayüzde, “Resmi Duyuru Sayfası”na yönlendiren bir bağlantının yer alması ise kullanıcıların detaylı bilgilere doğrudan ulaşabilmesini kolaylaştırmaktadır. Bu yapı, sistemin yalnızca bilgi erişim aracı değil, aynı zamanda resmi bildirim platformu olarak da işlev gördüğünü ortaya koymaktadır. Ayrıca içeriklerin güncel tarih bilgileriyle sunulması, sistemin dinamik olarak yönetildiğini ve sürekli güncellendiğini göstermektedir.



Şekil 4.7. Duyurular sayfası arayüzü

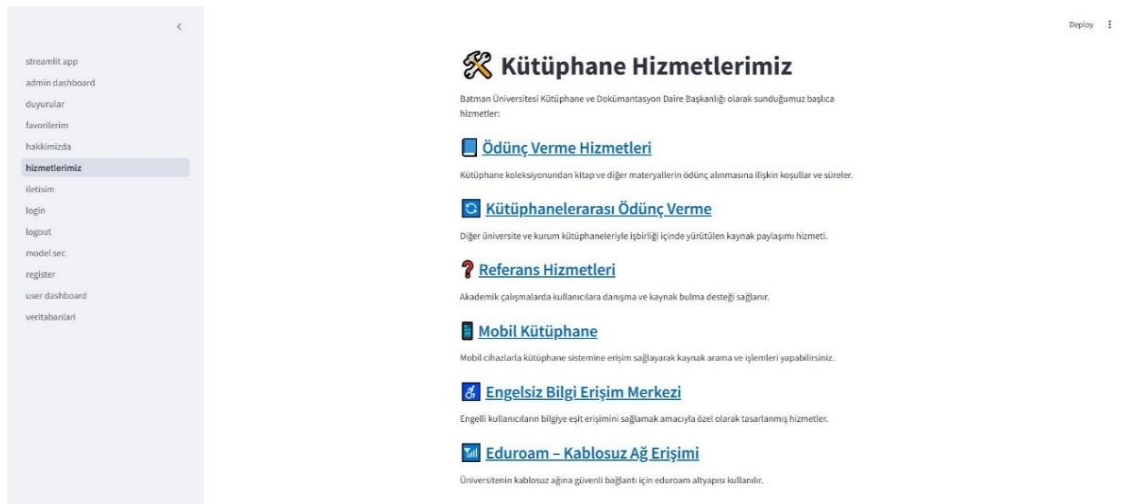
4.8. Hizmetlerimiz Sayfası: Kapsamlı Kullanıcı Destekleri

Şekil 4.8 ve Şekil 4.9’ da, Batman Üniversitesi Kütüphanesi tarafından sunulan hizmetlerin detaylandırıldığı “Hizmetlerimiz” sayfasına aittir. Bu sayfa, kullanıcıların kütüphane olanaklarından en etkin şekilde yararlanabilmesini sağlamak amacıyla zengin ve açıklayıcı içeriklerle donatılmıştır.

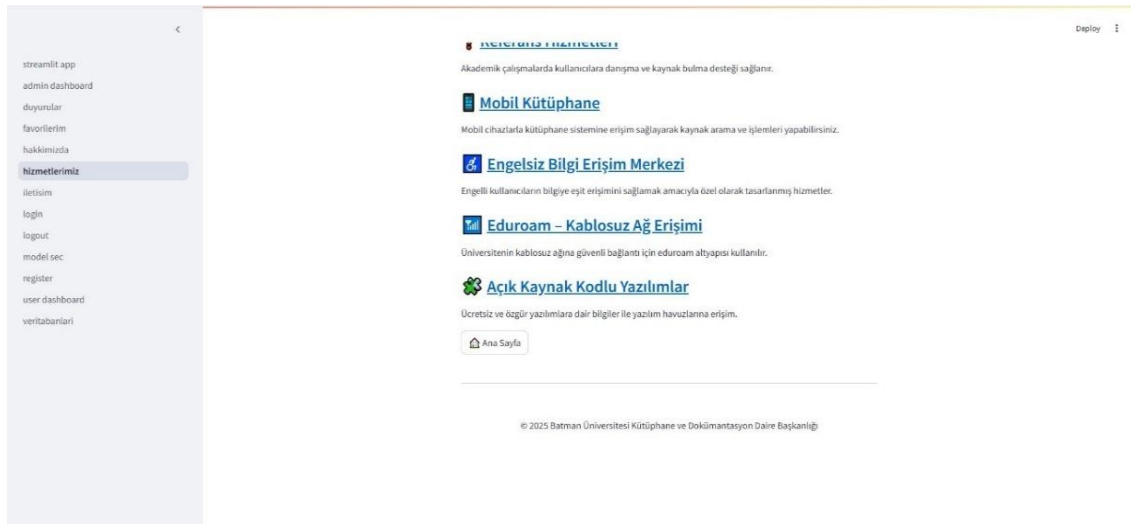
Sayfa, klasik hizmetlerin ötesine geçerek kullanıcı odaklı modern çözümleri de kapsamaktadır. “Ödünç Verme” ve “Kütüphaneler arası Ödünç” gibi temel hizmetlerin

yanı sıra; referans hizmetleri, mobil kütüphane erişimi, engelsiz bilgi erişimi, kablosuz ağ (Eduroam) bağlantısı ve açık kaynak yazılımlara erişim gibi çağdaş hizmetler de sunulmaktadır. Bu durum, sistemin yalnızca fiziksel kitap erişimini değil, dijital bilgiye erişim ve kapsayıcılığı da önceliklendirdiğini göstermektedir.

Ayrıca hizmet başlıklarının ikonlarla desteklenmesi, içeriklerin sade ve anlaşılır bir biçimde sunulmasına katkı sağlamıştır. Görsel tasarım ve bilgi mimarisi bakımından da kullanıcı dostu bir yaklaşım sergilenmiştir. Bu yapı hem akademik hem de teknik kullanıcıların ihtiyaçlarını bütüncül biçimde karşılamaya yönelik stratejik bir planlamanın ürünü olduğunu göstermektedir.



Şekil 4.8. Hizmetlerimiz sayfası arayüzü-1



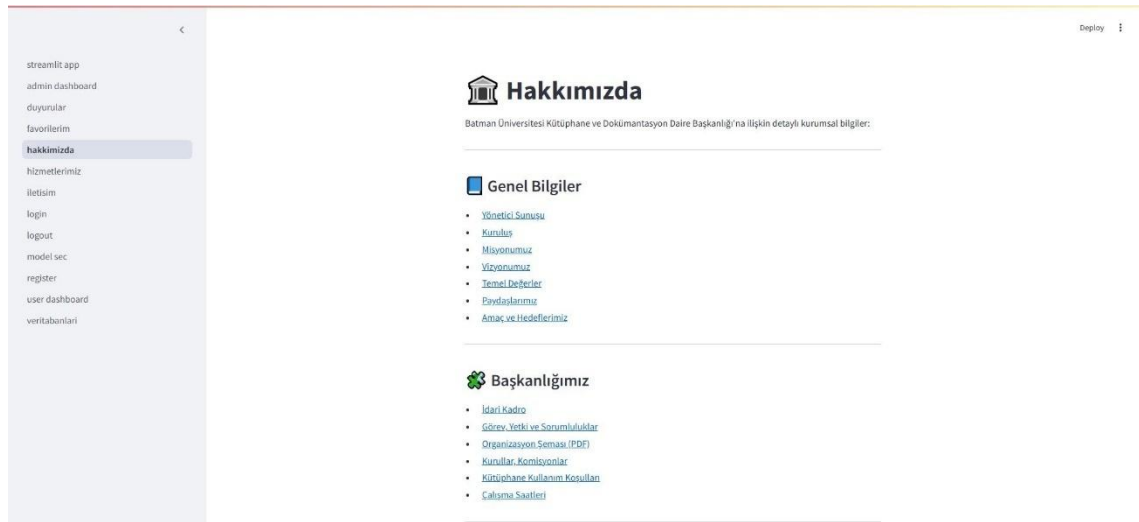
Şekil 4.9. Hizmetlerimiz sayfası arayüzü-2

4.9. Hakkımızda Sayfası: Kurumsal Bilgi Sunumu

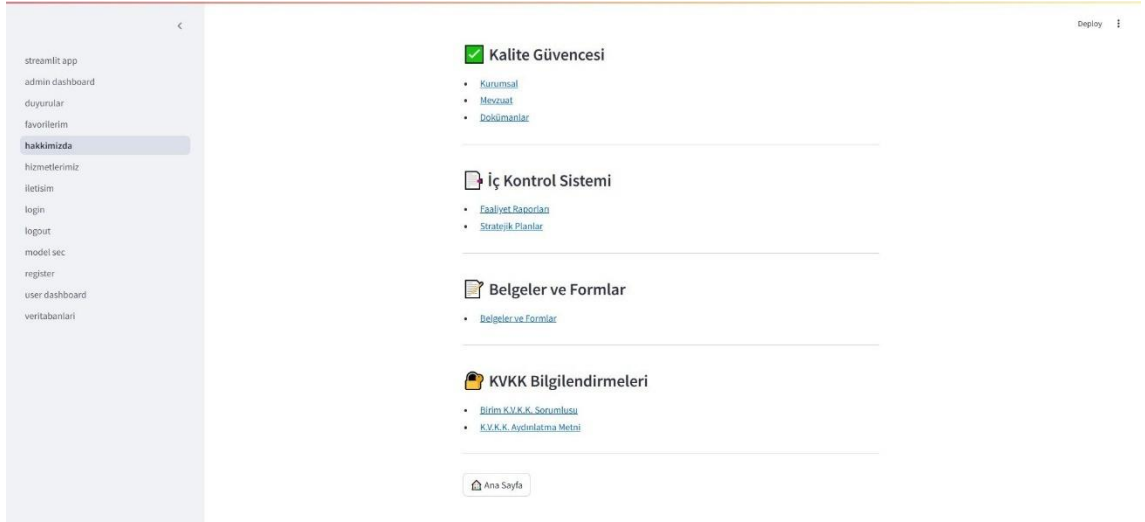
Şekil 4.10 ve Şekil 4.11’de yer alan “Hakkımızda” sayfası, Batman Üniversitesi Kütüphane ve Dokümantasyon Daire Başkanlığı hakkında kapsamlı kurumsal bilgi sunmayı amaçlayan bir yapıya sahiptir. Sayfa içeriği hem akademik hem de yönetsel anlamda şeffaf bilgi erişimi sağlamaktadır.

İlk bölümde “Genel Bilgiler” başlığı altında yöneticinin sunuş yazısı, kuruluş bilgileri, misyon, vizyon, temel değerler ve paydaş bilgileri gibi stratejik belgeler yer almakta; ayrıca “Başkanlığımız” başlığı altında idari kadro, görev tanımları, organizasyon şeması ve çalışma saatleri gibi yönetsel ayrıntılar listelenmektedir. Bu, kullanıcılara kurumun yapısı ve amaçları hakkında güvenilir ve detaylı bir bilgi altyapısı sunulduğunu göstermektedir.

İkinci bölümde “Kalite Güvencesi”, “İç Kontrol Sistemi”, “Belgeler ve Formlar” ve “KVKK Bilgilendirmeleri” gibi başlıklar altında kullanıcı hakları, veri güvenliği, stratejik planlar ve yasal sorumluluklara dair belgeler yer almaktadır. Bu yapı, üniversite kütüphanesinin yalnızca bir bilgi merkezi değil, aynı zamanda kurumsal şeffaflık ve yasal uyumluluk ilkelerine uygun bir organizasyon olduğunu ortaya koymaktadır.



Şekil 4.10. Hakkımızda-1

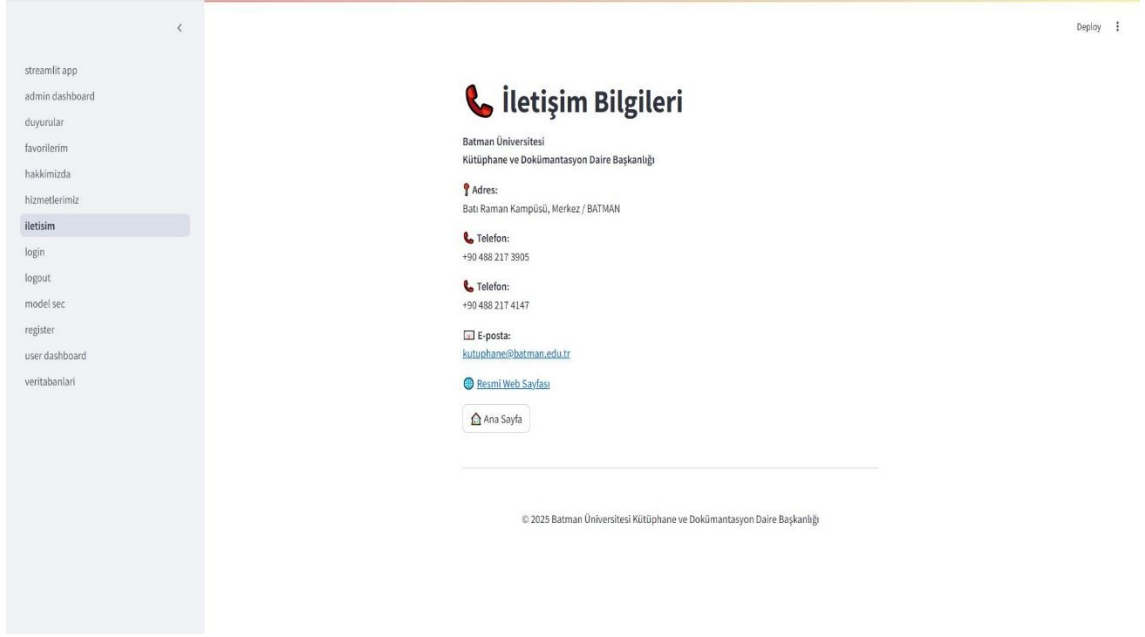


Şekil 4.11. Hakkımızda-2

4.10. İletişim Sayfası: Erişilebilirlik ve Şeffaflık

Şekil 4.12’de yer alan İletişim Bilgileri sayfası, Batman Üniversitesi Kütüphane ve Dokümantasyon Daire Başkanlığı’nın kullanıcılarla doğrudan iletişim kurmasını sağlayan resmi bir arayüz olarak yapılandırılmıştır. Sayfada fiziksel adres (Batı Raman Kampüsü), iki ayrı telefon numarası, kurumsal e-posta adresi ve resmi web sayfası bağlantısı açık ve sade bir biçimde sunulmuştur.

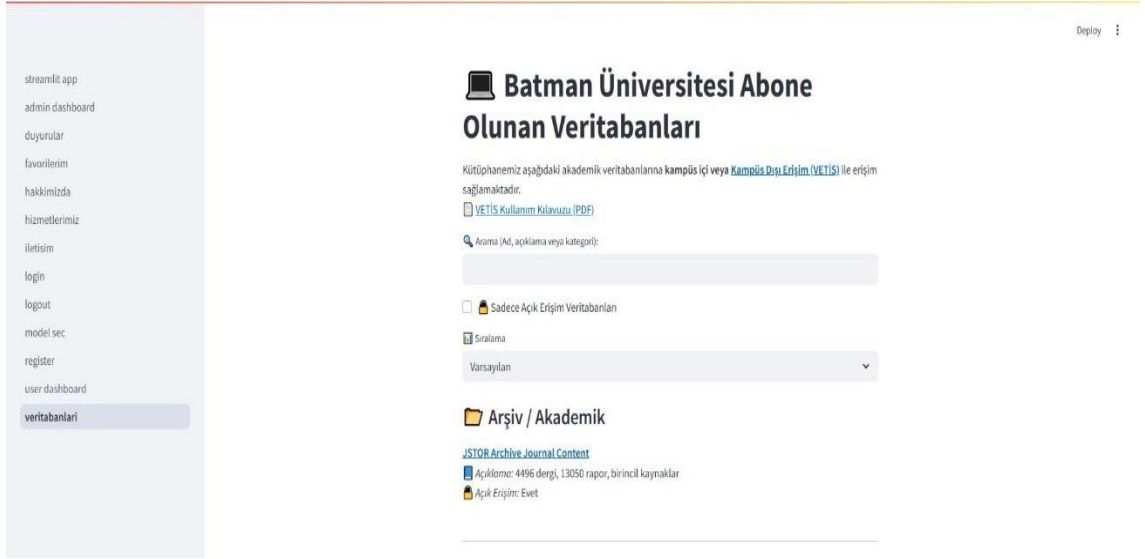
Bu yapı, kullanıcıların bilgi taleplerini veya sorunlarını iletebilmesini kolaylaştırmakta ve kurum ile kullanıcılar arasında doğrudan bir iletişim köprüsü kurmaktadır. Özellikle e-posta adresinin ve resmi sayfa bağlantısının tıklanabilir olması, dijital erişim kolaylığını artıran önemli bir özelliktir. Sayfanın görsel olarak sade ve anlaşılır tasarımı da kullanıcı dostu bir arayüz politikasının benimsendiğini göstermektedir.



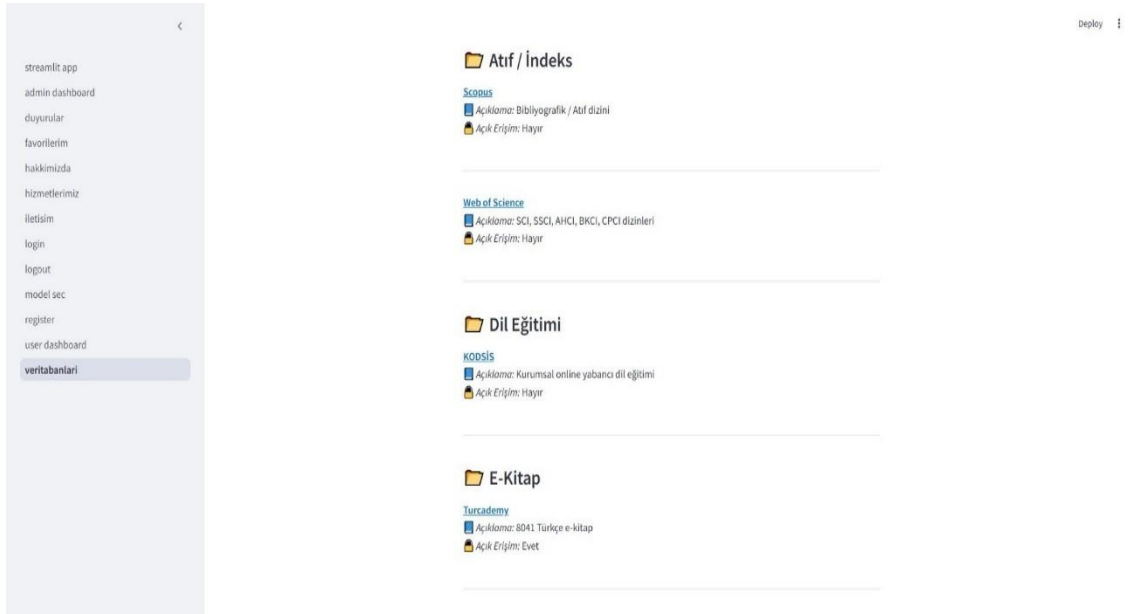
Şekil 4.12. İletişim sayfası arayüzü

4.11. Veri Tabanları

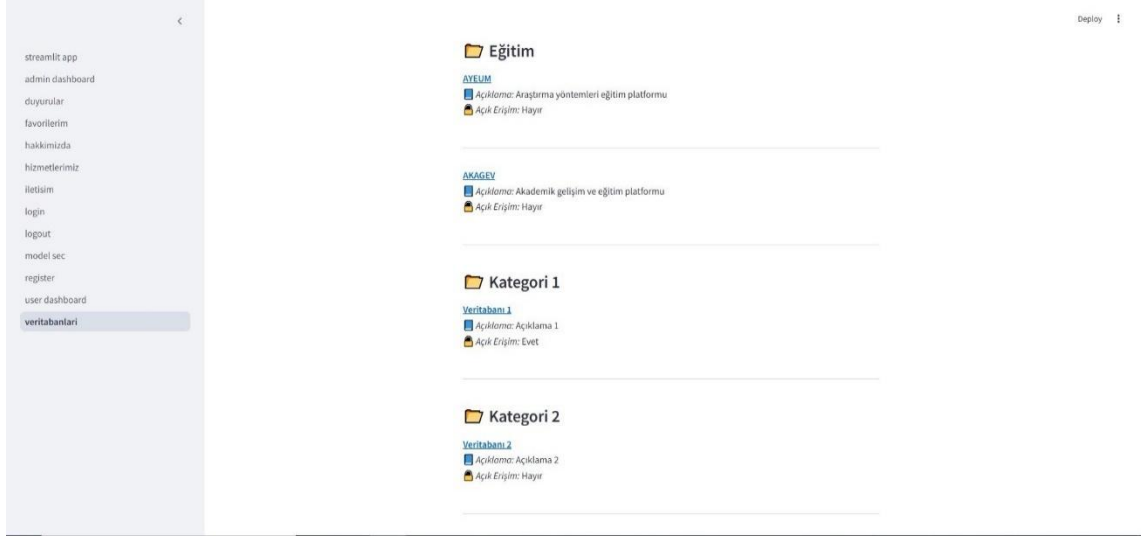
Batman Üniversitesi'nin sunduğu akademik veri tabanları arayüzü (Şekil 4.13 ile Şekil 4.20 arası), kullanıcı dostu ve minimalist bir tasarımla öğrencilerin ve akademik personelin bilgiye hızlı ve etkili biçimde erişimini kolaylaştırmaktadır. Veri tabanları, arşiv, atıf/indeks, dil eğitimi, e-kitap, referans yönetimi, savunma sanayi, tam metin kaynaklar ve intihal araçları gibi çeşitli kategorilere ayrılmıştır. Açıklamalar ve açık erişim durum bilgileri, kullanıcıların aradıkları kaynakları kolayca bulabilmelerini sağlamaktadır. Filtreleme ve sıralama özellikleri, aranan içeriğin hızlı bir şekilde daraltılmasına olanak tanırken, CSV formatında dışa aktarma seçeneği de kaynak kullanımını ve araştırma süreçlerini destekleyici niteliktedir. Bu yapı, kullanıcıların akademik çalışmalarda zaman yönetimini ve araştırma kalitesini artırıcı yönde tasarlanmıştır.



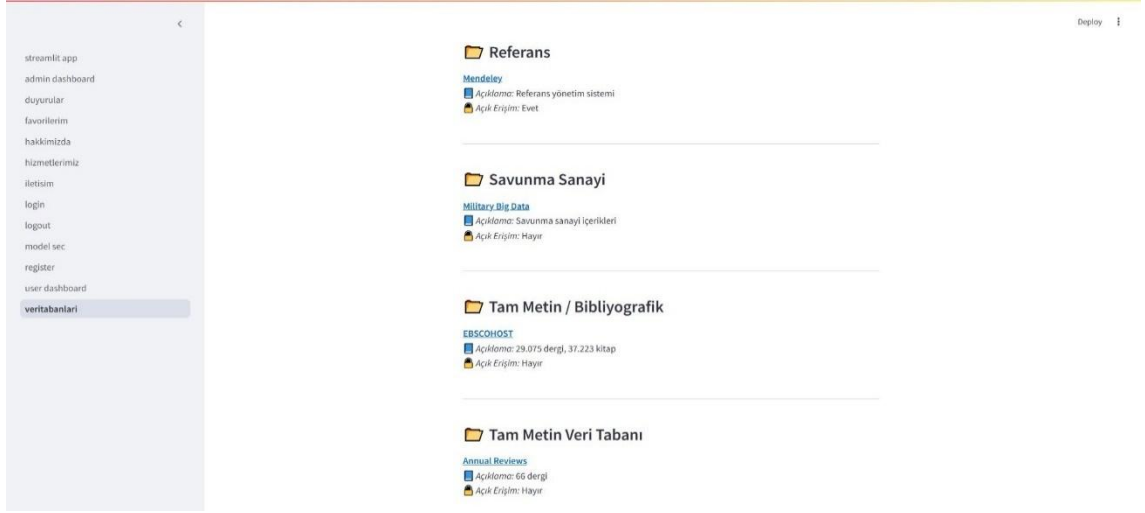
Şekil 4.13. Veri tabanı-1



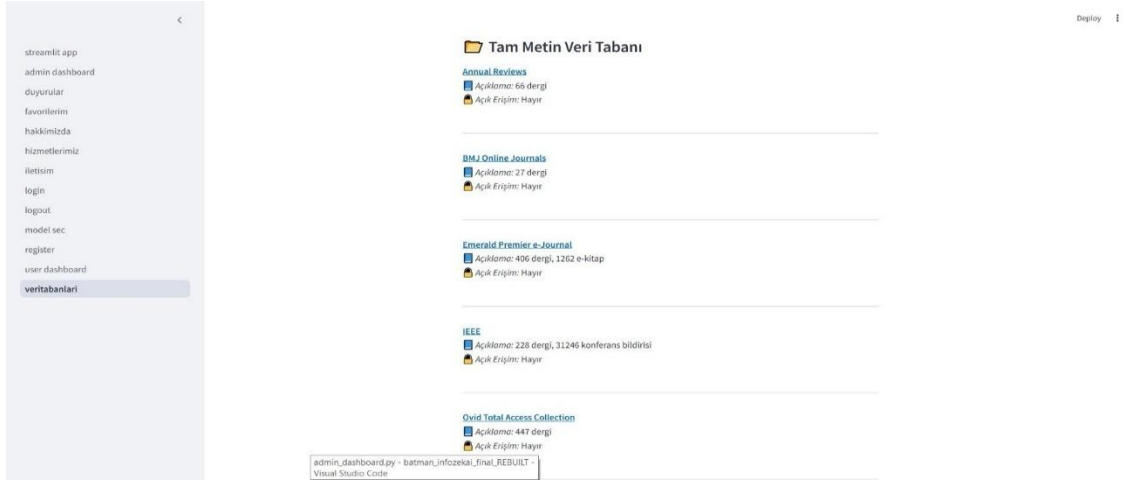
Şekil 4.14. Veri tabanı-2



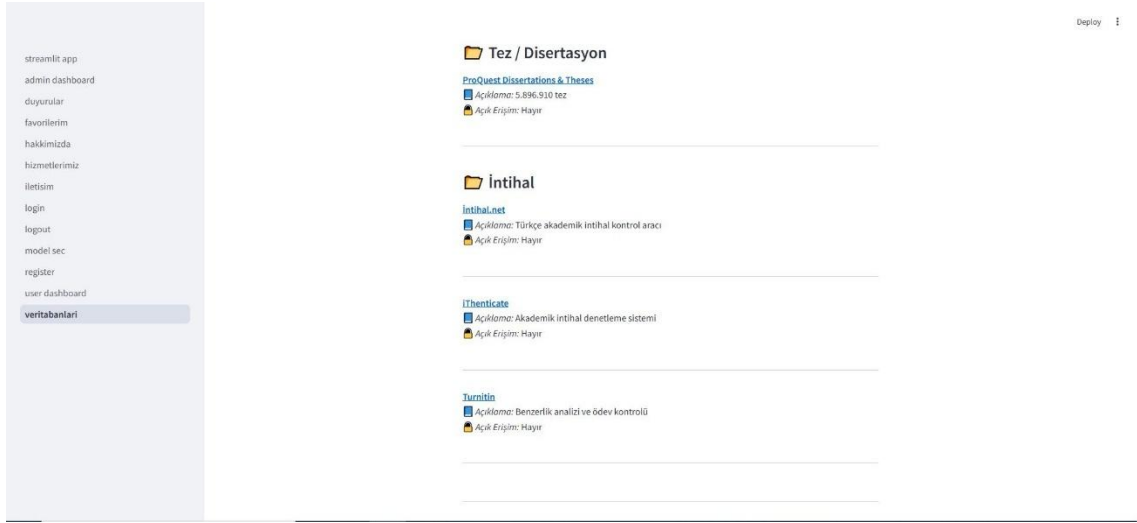
Şekil 4.15. Veri tabanı-3



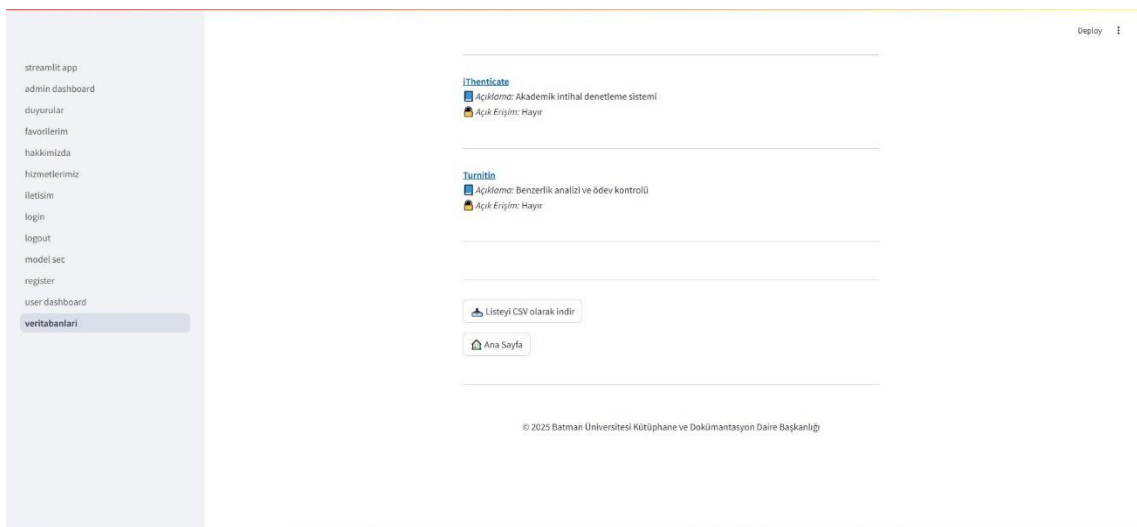
Şekil 4.16. Veri tabanı-4



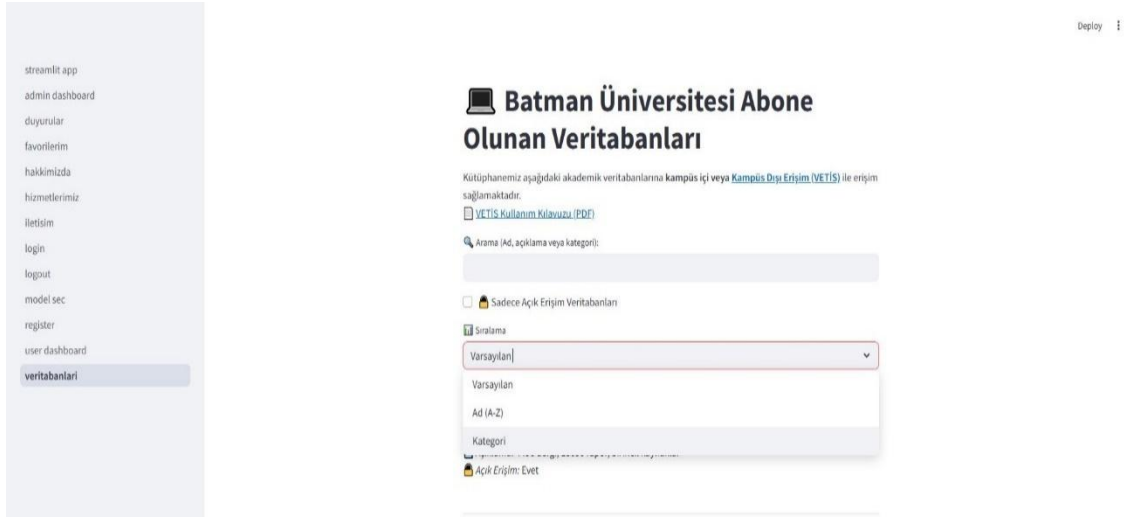
Şekil 4.17. Veri tabanı-5



Şekil 4.18. Veri tabanı-6



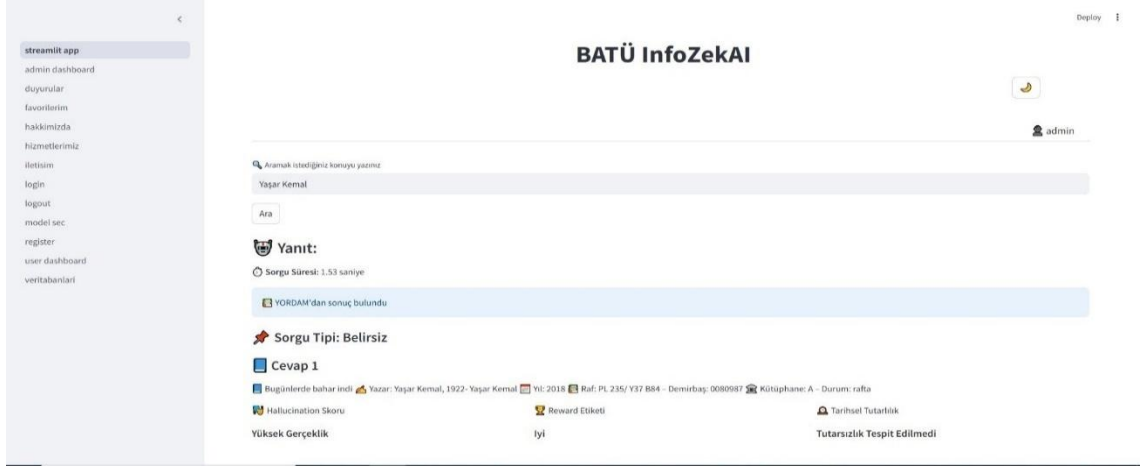
Şekil 4.19. Veri tabanı-7



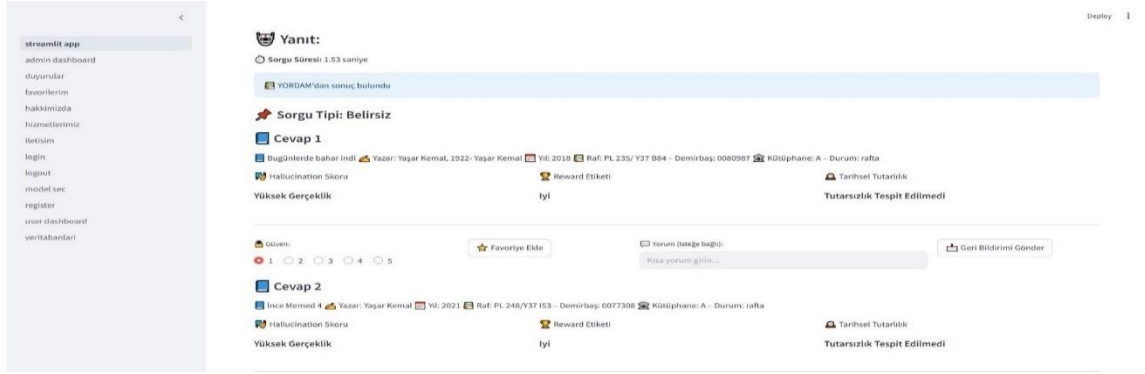
Şekil 4.20. Veri tabanı-8

4.12. Yordam, E-Arşiv, ChatGPT Sorgu Sonuçları

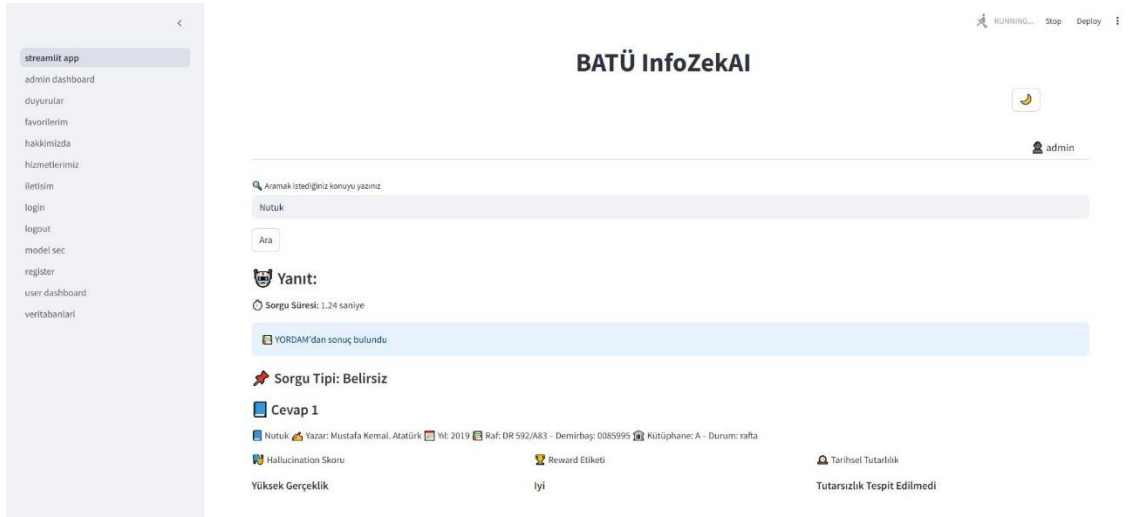
BATÜ InfoZekAI kütüphane otomasyon sistemi(Şekil 4.21 ile Şekil 4.32 arası); bilgiye erişimde üç aşamalı bir yaklaşımı kullanmaktadır. İlk olarak YORDAM sorguları ile kütüphane kataloglarında fiziksel kitaplara ilişkin sonuçlar sunulmaktadır; kaynak adı, yazar bilgisi, yayın yılı ve kütüphane konumu gibi detaylar belirtilmektedir. Eğer YORDAM’da uygun sonuç bulunamazsa sistem ikinci adım olarak Batman Üniversitesi'nin E-Arşiv veri tabanında bulunan akademik yayınlar (tezler, kongre bildirileri, makaleler vb.) içinden sonuçları kullanıcıya detaylı bibliyografik bilgi ve tam metin bağlantılarıyla sunmaktadır. Son aşamada, eğer ilk iki kaynaktan sonuç alınamazsa, sistem ChatGPT'yi kullanarak doğal dil tabanlı ve genel bilgiler içeren yanıtları kullanıcıyla paylaşmaktadır; ayrıca sunulan her cevaba hallucination skoru, tarihsel tutarlılık ve güven derecesi gibi değerlendirme araçlarıyla kullanıcı etkileşimini artıran özellikler entegre edilmiştir.



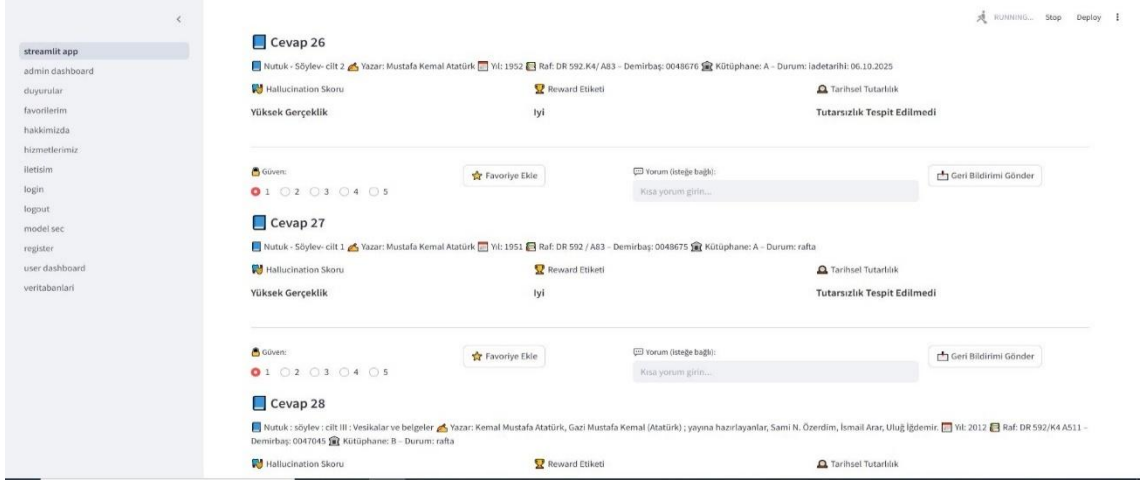
Şekil 4.21. YORDAM sonuç-1



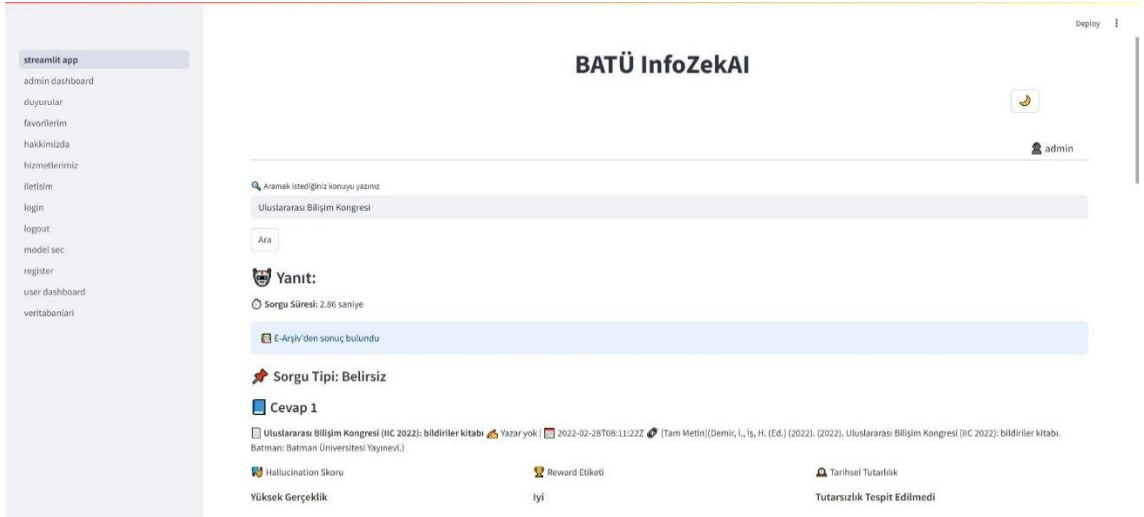
Şekil 4.22. YORDAM sonuç-2



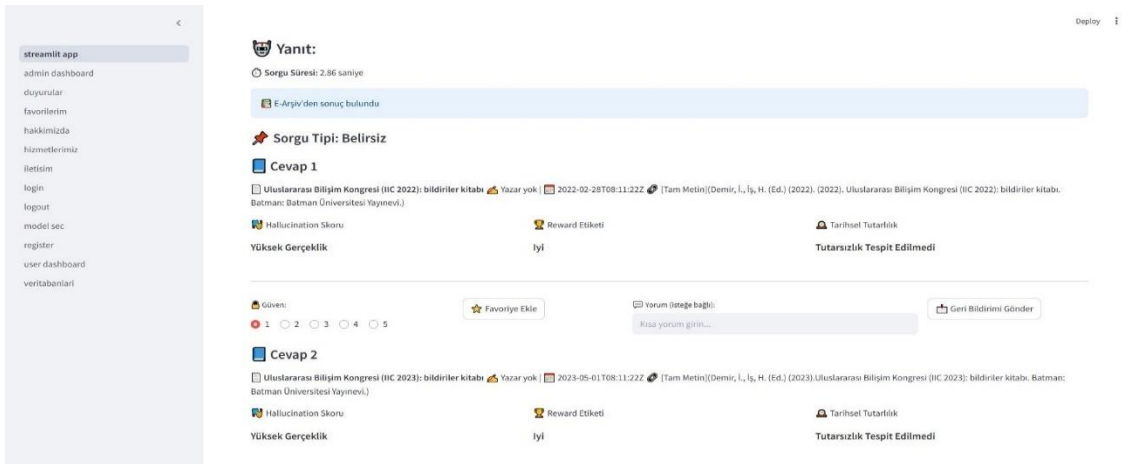
Şekil 4.23. YORDAM sonuç-3



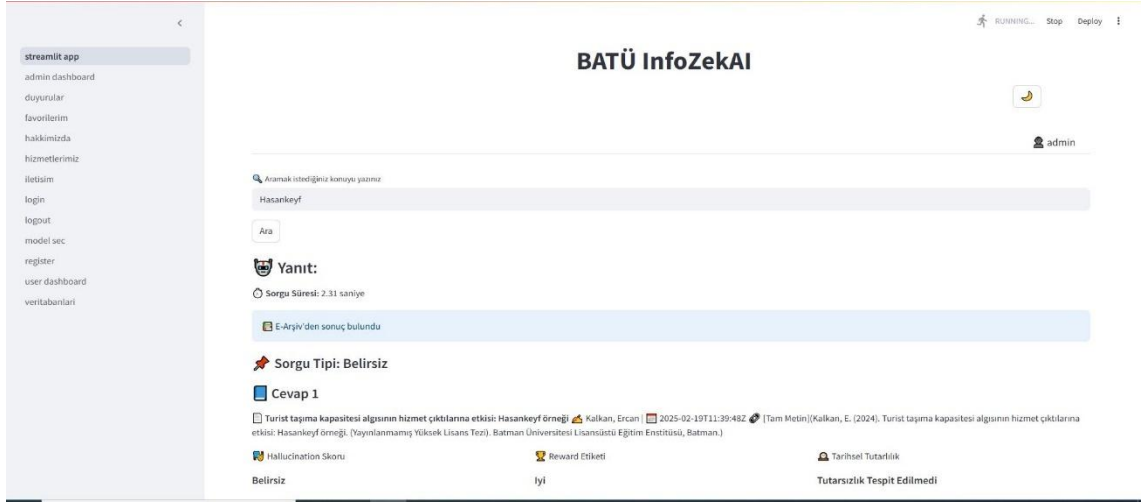
Şekil 4.24. YORDAM sonuç-4



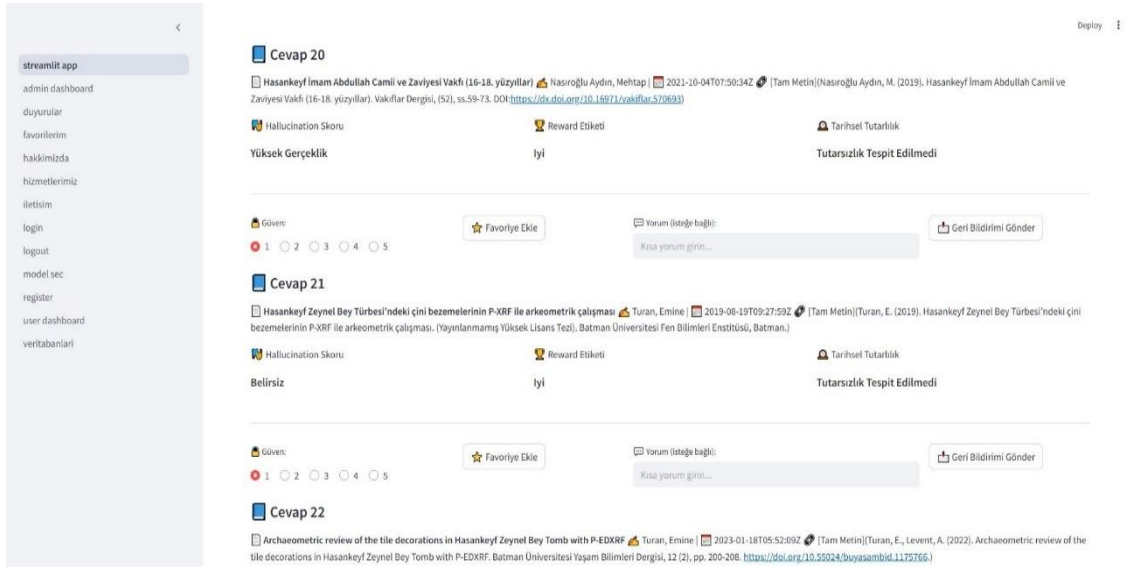
Şekil 4.25. E-Arşiv-1



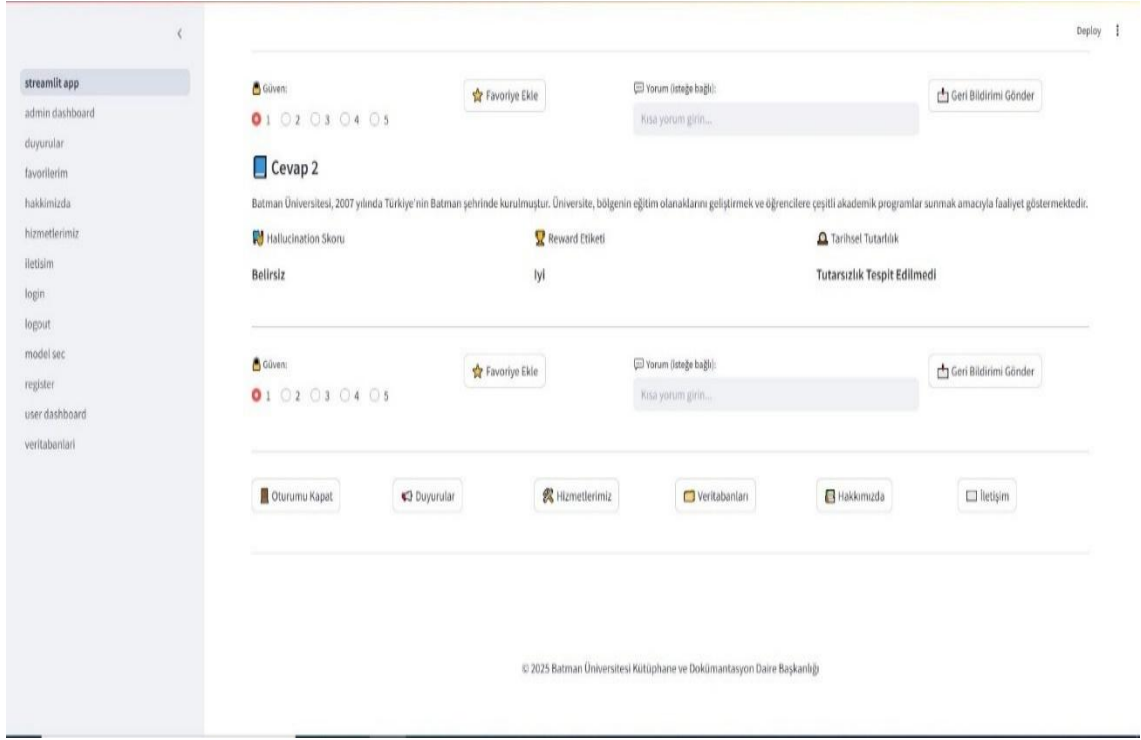
Şekil 4.26. E-Arşiv-2



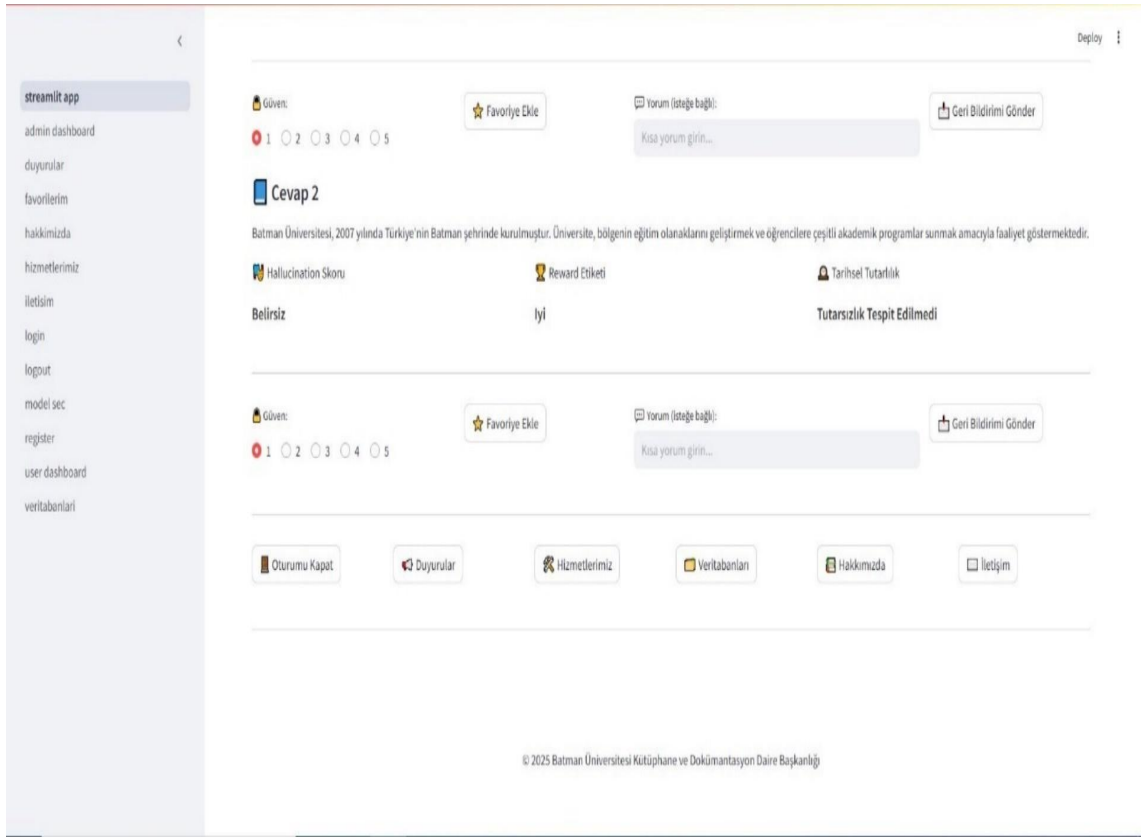
Şekil 4.27. E-Arşiv-3



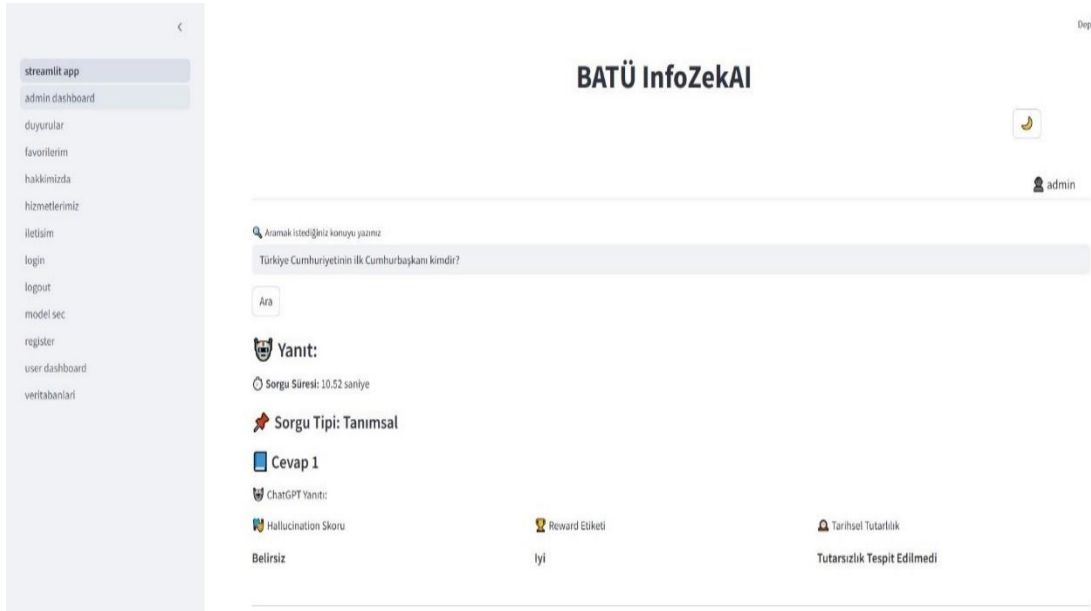
Şekil 4.28. E-Arşiv-4



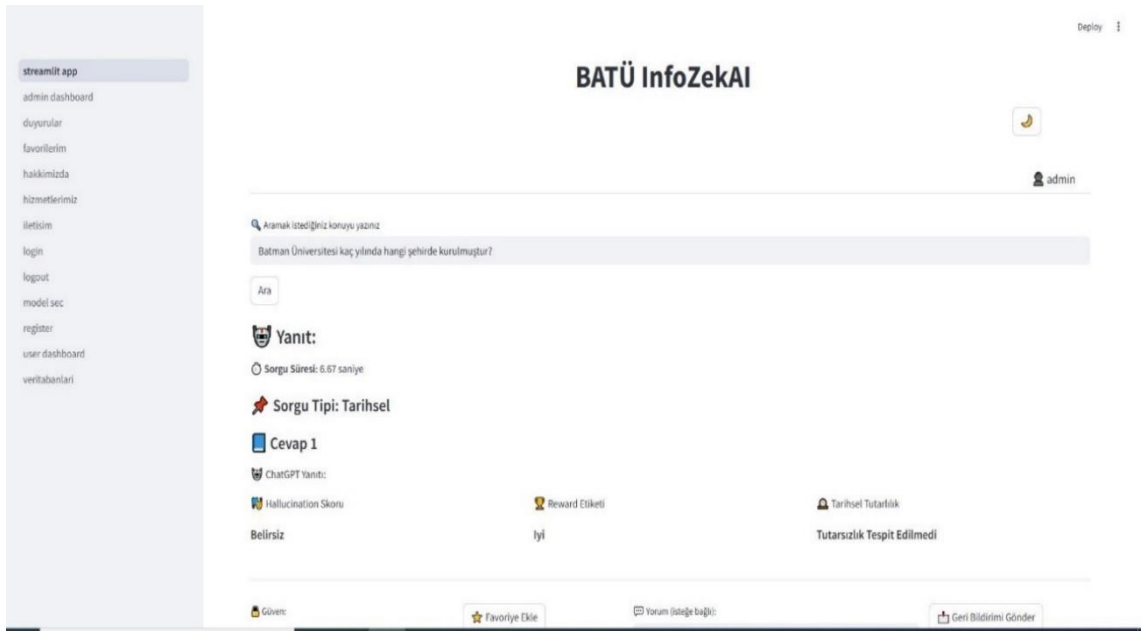
Şekil 4.29. ChatGPT-1



Şekil 4.30. ChatGPT-2



Şekil 4.31. ChatGPT-3

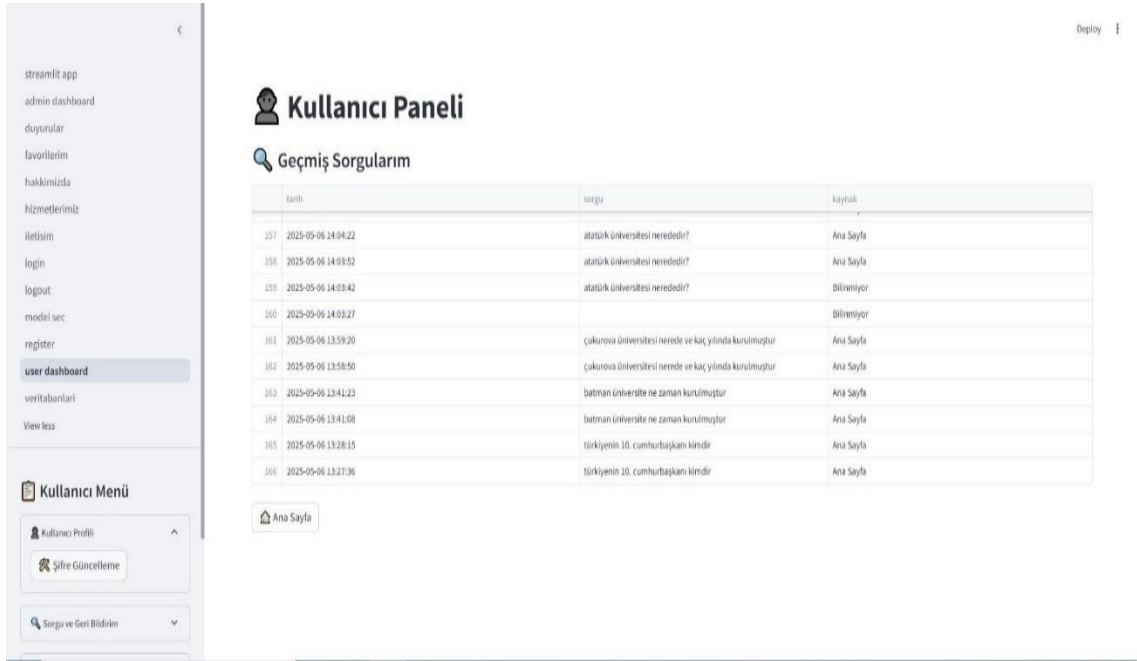


Şekil 4.32. ChatGPT-4

4.13. Kullanıcı Paneli İşlemleri

Kullanıcılar, yaptıkları sorguları ve gelen yanıtları "Favori Verilerim ve Yanıtlarım" sekmesiyle favorilere ekleyerek daha sonra tekrar inceleyebiliyor. "Geçmiş Sorgularım" bölümünde, kullanıcıların daha önce yaptıkları sorgular tarih ve kaynak

bilgisiyle kayıt altına alınmakta. "Geri Bildirim ve Puanlamalarım" kısmında, kullanıcıların verdiği puanlar ve yorumlar sorgu bazında ayrıntılı olarak gösterilmekte. Ayrıca, "Reward Model Skorlarım" sayfasında yapılan sorguların ödül modelleriyle değerlendirilmiş sonuçları, tarih, sorgu tipi ve tutarsızlık kontrolüyle detaylandırılarak sunulmaktadır. "Model Karşılaştırma ve Performans Metrikleri" alanında ise kullanıcıların farklı yapay zekâ modellerini seçerek, performans karşılaştırması yapabileceği bir arayüz sağlanmıştır. Son olarak, kullanıcıların güvenliğini sağlayan basit bir "Şifre Güncelleme" bölümü bulunmaktadır. Bu sistem, kullanıcı deneyimini zenginleştiren ve interaktif kullanım sağlayan bir yapı sunmaktadır (Şekil 4.33 ile Şekil 4.39 arası).

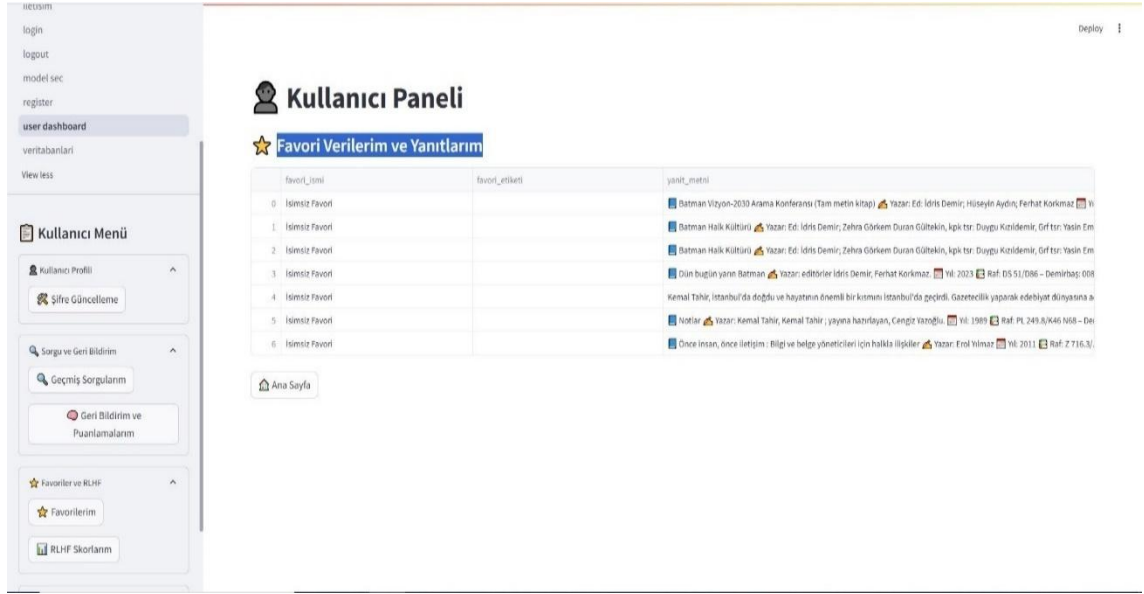


The screenshot displays the 'Kullanıcı Paneli' (User Panel) interface. On the left is a sidebar menu with options like 'streamlit app', 'admin dashboard', 'duyurular', 'favorilerim', 'hakkımızda', 'hizmetlerimiz', 'iletisim', 'login', 'logout', 'model sec', 'register', 'user dashboard' (highlighted), 'veritabanları', and 'View less'. Below the menu is a 'Kullanıcı Menü' (User Menu) section with 'Kullanıcı Profili', 'Şifre Güncelleme', and 'Sorgu ve Geri Bildirim'. The main content area is titled 'Kullanıcı Paneli' and 'Geçmiş Sorgularım' (Past Queries). It contains a table with the following data:

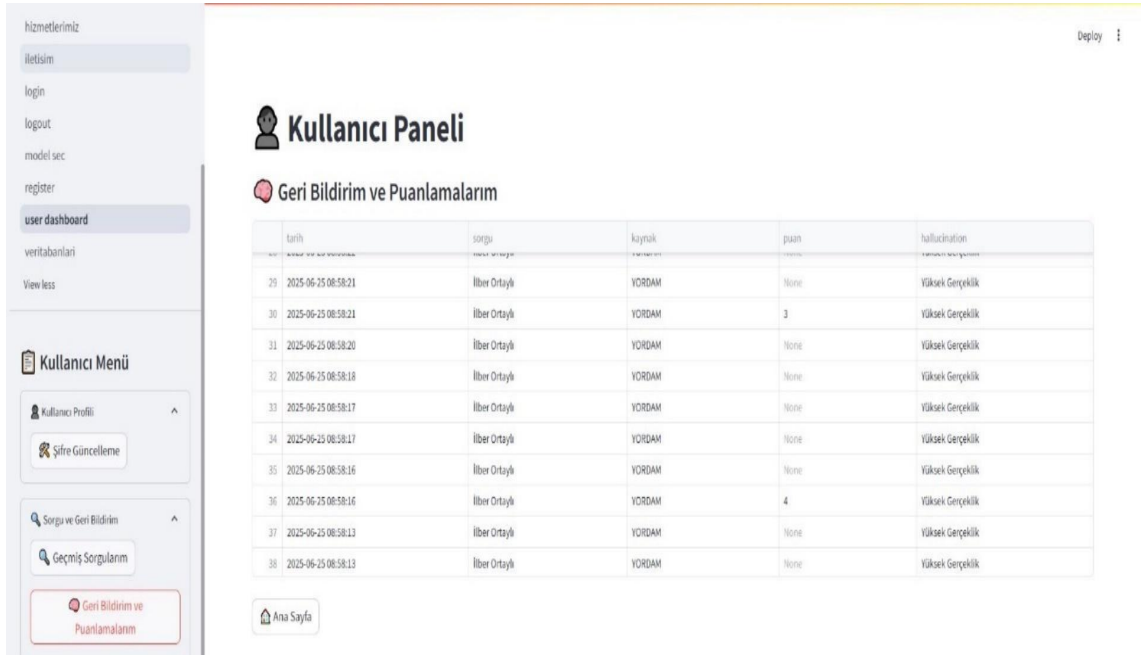
	tarih	sorgu	kaynak
157	2025-05-06 14:04:22	atatürk üniversitesi nerededir?	Ana Sayfa
158	2025-05-06 14:03:52	atatürk üniversitesi nerededir?	Ana Sayfa
159	2025-05-06 14:03:42	atatürk üniversitesi nerededir?	Bilinmiyor
160	2025-05-06 14:03:27		Bilinmiyor
161	2025-05-06 13:59:20	çukurova üniversitesi nerede ve kaç yılında kurulmuştur	Ana Sayfa
162	2025-05-06 13:58:50	çukurova üniversitesi nerede ve kaç yılında kurulmuştur	Ana Sayfa
163	2025-05-06 13:41:23	batman üniversite ne zaman kurulmuştur	Ana Sayfa
164	2025-05-06 13:41:08	batman üniversite ne zaman kurulmuştur	Ana Sayfa
165	2025-05-06 13:28:15	türkiyenin 10. cumhurbaşkanı kimdir	Ana Sayfa
166	2025-05-06 13:27:36	türkiyenin 10. cumhurbaşkanı kimdir	Ana Sayfa

At the bottom left of the main content area is a button labeled 'Ana Sayfa'.

Şekil 4.33. Kullanıcı paneli işlemleri-1



Şekil 4.34. Kullanıcı paneli işlemleri-2



Şekil 4.35. Kullanıcı paneli işlemleri-3

register

user dashboard

veritabanları

View less

Kullanıcı Menü

Kullanıcı Profili

Sıfre Güncelleme

Sorgu ve Geri Bildirim

Geçmiş Sorgularım

Geri Bildirim ve Puanlamalarım

Favoriler ve RLHF

Favorilerim

RLHF Skorlarım

Deploy

Kullanıcı Paneli

Model Karşılaştırma Geçmişim

Model Karşılaştırma

Model Performans Metrikleri

Hangi modeli göstermek istersiniz?

BERTurk

BERTurk

DistiBERT

ElectraTR

RoBERTa

SentimentBERT

TinyLLaMA

BERTurk Metrikleri

Şekil 4.36. Kullanıcı paneli işlemleri-4

netişim

login

logout

model sec

register

user dashboard

veritabanları

View less

Kullanıcı Menü

Kullanıcı Profili

Sıfre Güncelleme

Sorgu ve Geri Bildirim

Geçmiş Sorgularım

Geri Bildirim ve Puanlamalarım

Favoriler ve RLHF

Favorilerim

RLHF Skorlarım

Model Analizi

Deploy

Kullanıcı Paneli

Reward Model Skorlarım

	tarix	sorgu	model	reward_label	sorgu_tipi	tarixsel_tutarsizlik
0	2025-06-25 09:00:41	İdris Demir	reward_model	iyi	Belirsiz	Tutarsızlık Tespit Edilmedi
1	2025-06-25 09:00:31	İdris Demir	reward_model	iyi	Belirsiz	Tutarsızlık Tespit Edilmedi
2	2025-06-25 09:00:26	İdris Demir	reward_model	iyi	Belirsiz	Tutarsızlık Tespit Edilmedi
3	2025-06-25 09:00:15	İdris Demir	reward_model	iyi	Belirsiz	Tutarsızlık Tespit Edilmedi
4	2025-06-25 09:00:11	İdris Demir	reward_model	iyi	Belirsiz	Tutarsızlık Tespit Edilmedi
5	2025-06-25 08:58:38	İlber Örtaylı	reward_model	iyi	Belirsiz	Tutarsızlık Tespit Edilmedi
6	2025-06-25 08:58:30	İlber Örtaylı	reward_model	iyi	Belirsiz	Tutarsızlık Tespit Edilmedi
7	2025-06-25 08:58:25	İlber Örtaylı	reward_model	iyi	Belirsiz	Tutarsızlık Tespit Edilmedi
8	2025-06-25 08:58:20	İlber Örtaylı	reward_model	iyi	Belirsiz	Tutarsızlık Tespit Edilmedi
9	2025-06-25 08:58:15	İlber Örtaylı	reward_model	iyi	Belirsiz	Tutarsızlık Tespit Edilmedi

Ana Sayfa

Şekil 4.37. Kullanıcı paneli işlemleri-5

İletişim

login

logout

model sec

register

user dashboard

veritabanları

View less

Kullanıcı Menü

Kullanıcı Profili

Şifre Güncelleme

Sorgu ve Geri Bildirim

Geçmiş Sorgularım

Geri Bildirim ve Puanlamalarım

Favoriler ve RLHF

Favorilerim

RLHF Skorlarım

Kullanıcı Paneli

Reward Model Skorlarım

	tarih	sorgu	model	reward_label	sorgu_tipi	tarihsel_tutarlılık
0	2025-06-25 09:00:41	İdris Demir	reward_model	iyi	Belirsiz	Tutarlılık Tespit Edilmedi
1	2025-06-25 09:00:31	İdris Demir	reward_model	iyi	Belirsiz	Tutarlılık Tespit Edilmedi
2	2025-06-25 09:00:26	İdris Demir	reward_model	iyi	Belirsiz	Tutarlılık Tespit Edilmedi
3	2025-06-25 09:00:15	İdris Demir	reward_model	iyi	Belirsiz	Tutarlılık Tespit Edilmedi
4	2025-06-25 09:00:11	İdris Demir	reward_model	iyi	Belirsiz	Tutarlılık Tespit Edilmedi
5	2025-06-25 08:58:38	İlber Ortaylı	reward_model	iyi	Belirsiz	Tutarlılık Tespit Edilmedi
6	2025-06-25 08:58:30	İlber Ortaylı	reward_model	iyi	Belirsiz	Tutarlılık Tespit Edilmedi
7	2025-06-25 08:58:25	İlber Ortaylı	reward_model	iyi	Belirsiz	Tutarlılık Tespit Edilmedi
8	2025-06-25 08:58:20	İlber Ortaylı	reward_model	iyi	Belirsiz	Tutarlılık Tespit Edilmedi
9	2025-06-25 08:58:15	İlber Ortaylı	reward_model	iyi	Belirsiz	Tutarlılık Tespit Edilmedi

Ana Sayfa

<https://katalog.batman.edu.tr/yordam/webservis/websevis.php - My Workspace>

Şekil 4.38. Kullanıcı paneli işlemleri-6

streamlit app

admin dashboard

duyurular

favorilerim

hakkımızda

hizmetlerimiz

iletişim

login

logout

model sec

register

user dashboard

veritabanları

View less

Kullanıcı Menü

Kullanıcı Profili

Şifre Güncelleme

Sorgu ve Geri Bildirim

Kullanıcı Paneli

Şifre Güncelleme

Yeni Şifre

Şifreyi Güncelle

Ana Sayfa

Şekil 4.39. Kullanıcı paneli işlemleri-7

4.14. Admin Dashboard Kullanıcı İşlemleri

Kullanıcı panellerinde favori içeriklerin, geçmiş sorguların, geri bildirimlerin, puanlamaların ve ödül modeli (reward model) skorlarının görüntülenebildiği arayüzler sunulmuştur. Admin panellerinde ise kullanıcı bazlı sorgu analizleri, kullanıcı yönetimi, rol güncelleme, şifre sıfırlama ve kullanıcı listesini içeren kapsamlı yönetim ekranları bulunmaktadır. Tüm bu ekranlar, kullanıcıların ve yöneticilerin etkileşimlerini kolaylaştırarak kişiselleştirilmiş veri yönetimi ve sistem içi analiz imkânı sağlamaktadır (Şekil 4.40 ile Şekil 4.47 arası).

The screenshot displays the Admin Dashboard interface. On the left, there is a sidebar menu with options like 'admin dashboard', 'duyurular', 'favorilerim', 'hakımızda', 'hizmetlerimiz', 'iletisim', 'login', 'logout', 'model sec', 'register', 'user dashboard', 'veritabanlari', and 'View less'. Below this is an 'Admin Menü' section with 'Kullanıcı Yönetimi' and 'Kullanıcı Bazlı İnceleme' buttons. The main content area is titled 'Admin Dashboard' and shows a 'Kullanıcı Bazlı Veriler' section. It includes a dropdown menu for selecting a user, currently set to 'admin'. Below this is a table with columns: 'sıra', 'tarih', 'sorgu', 'kullanıcı', 'puan', and 'durum'. The table contains 7 rows of data, all showing a score of 1000 and a status of 'Yüksek Gerçeklik'.

sıra	tarih	sorgu	kullanıcı	puan	durum
0	2025-06-25 09:00:48	İdris Demir	YORDAN	1000	Yüksek Gerçeklik
1	2025-06-25 09:00:48	İdris Demir	YORDAN	1000	Yüksek Gerçeklik
2	2025-06-25 09:00:47	İdris Demir	YORDAN	1000	Yüksek Gerçeklik
3	2025-06-25 09:00:47	İdris Demir	YORDAN	1000	Yüksek Gerçeklik
4	2025-06-25 09:00:46	İdris Demir	YORDAN	1000	Yüksek Gerçeklik
5	2025-06-25 09:00:46	İdris Demir	YORDAN	1000	Yüksek Gerçeklik
6	2025-06-25 09:00:45	İdris Demir	YORDAN	1000	Yüksek Gerçeklik

Şekil 4.40. Admin dashboard-1

favorilerim
hakkımızda
hizmetlerimiz
iletisim
login
logout
model sec
register
user dashboard
veritabanları
View less

Admin Menü

Kullanıcı Yönetimi

Kullanıcı Yönetimi

Kullanıcı Bazlı İnceleme

Analiz ve Raporlama

Kullanıcı Bazlı Veriler

Bir kullanıcı seçin:

kutuphane kullanıcısının Geri Bildirimleri

	tarix	sorgu	kaynak	puan	hallucination
0	2025-05-06 14:09:36	Mersin hangi bölgededir	Ana Sayfa	4	Belirsiz
1	2025-05-06 14:09:36	Mersin hangi bölgededir	Ana Sayfa	None	Belirsiz
2	2025-05-06 14:09:09	Mersin hangi bölgededir	Ana Sayfa	None	Belirsiz
3	2025-05-06 14:09:08	Mersin hangi bölgededir	Ana Sayfa	None	Belirsiz
4	2025-05-06 10:39:03	Türkiye'nin başkenti neresidir?	Ana Sayfa	None	Belirsiz
5	2025-05-06 10:39:02	Türkiye'nin başkenti neresidir?	Ana Sayfa	None	Belirsiz
6	2025-05-06 10:35:39	1. cumhurbaşkanımız kimdir	Ana Sayfa	None	Belirsiz
7	2025-05-06 10:35:38	1. cumhurbaşkanımız kimdir	Ana Sayfa	None	Belirsiz
8	2025-05-06 10:35:29	yaşar kemal	YORDAM	None	Yüksek Gerçeklik
9	2025-05-06 10:35:29	yaşar kemal	YORDAM	None	Yüksek Gerçeklik

Şekil 4.41. Admin dashboard-2

admin dashboard
duyurular
favorilerim
hakkımızda
hizmetlerimiz
iletisim
login
logout
model sec
register
user dashboard
veritabanları
View less

Admin Menü

Kullanıcı Yönetimi

Kullanıcı Yönetimi

Kullanıcı Bazlı İnceleme

Analiz ve Raporlama

Veri Yönetimi

admin kullanıcısının RLHF Skorları

Sorgu Süresi İstatistikleri

Ortalama Sorgu Süresi: 3.17 saniye

En Uzun Sorgu Süresi: 39.45 saniye

En Kısa Sorgu Süresi: 0.00 saniye

	sorgu	model	reward_label	sorgu_tipi	tarixsel_tutarlilik	tarix	sorgu_suresi
0	İdris Demir	reward_model	iyi	Belirsiz	Tutarlılık Tespit Edilmedi	2025-06-25 09:00:41	0.4456
1	İdris Demir	reward_model	iyi	Belirsiz	Tutarlılık Tespit Edilmedi	2025-06-25 09:00:31	0.3619
2	İdris Demir	reward_model	iyi	Belirsiz	Tutarlılık Tespit Edilmedi	2025-06-25 09:00:26	0.4396
3	İdris Demir	reward_model	iyi	Belirsiz	Tutarlılık Tespit Edilmedi	2025-06-25 09:00:15	0.3704
4	İdris Demir	reward_model	iyi	Belirsiz	Tutarlılık Tespit Edilmedi	2025-06-25 09:00:11	0.7827
5	İlber Ortaylı	reward_model	iyi	Belirsiz	Tutarlılık Tespit Edilmedi	2025-06-25 08:58:38	0.429
6	İlber Ortaylı	reward_model	iyi	Belirsiz	Tutarlılık Tespit Edilmedi	2025-06-25 08:58:30	0.5978
7	İlber Ortaylı	reward_model	iyi	Belirsiz	Tutarlılık Tespit Edilmedi	2025-06-25 08:58:25	0.4606
8	İlber Ortaylı	reward_model	iyi	Belirsiz	Tutarlılık Tespit Edilmedi	2025-06-25 08:58:20	0.4772
9	İlber Ortaylı	reward_model	iyi	Belirsiz	Tutarlılık Tespit Edilmedi	2025-06-25 08:58:15	0.4802

Ortalama RLHF Skoru

2.96

Ana Sayfa

<https://katalog.batman.edu.tr/yordam/webservis/webserwis.php> - My Workspace

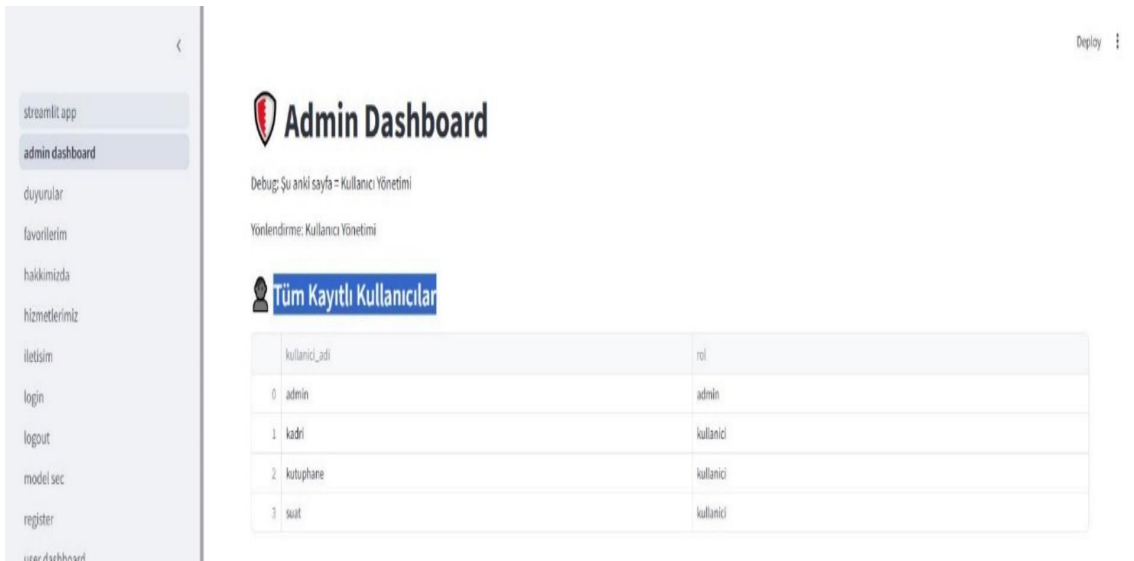
Şekil 4.42. Admin dashboard-3



Şekil 4.43. Admin dashboard-4



Şekil 4.44. Admin dashboard-5



Şekil 4.45. Admin dashboard-6



Şekil 4.46. Admin dashboard-7

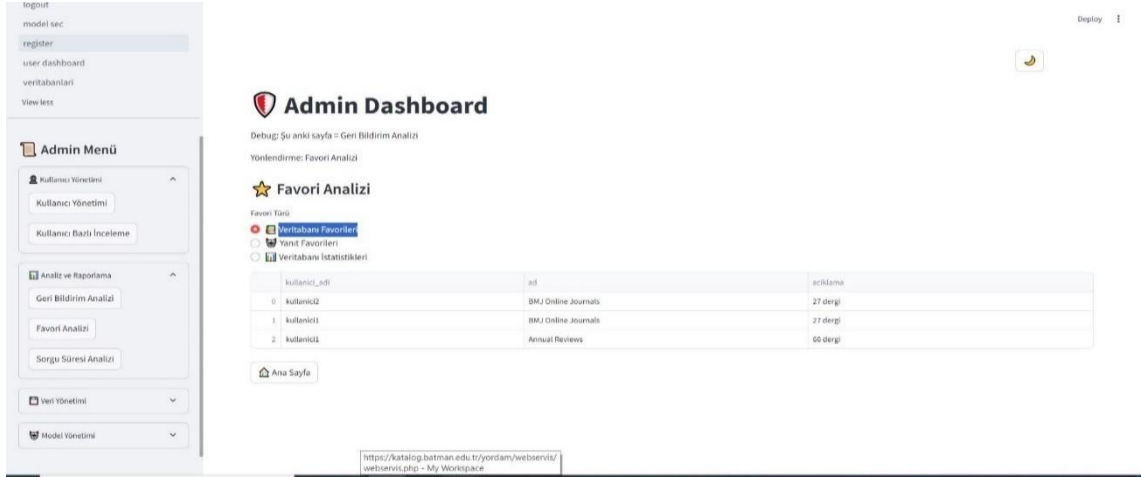


Şekil 4.47. Admin dashboard-8

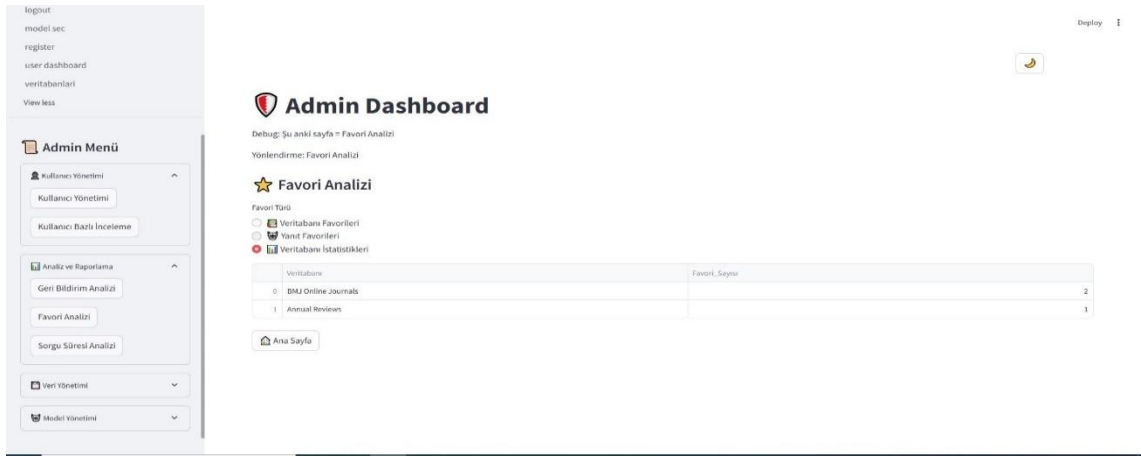
4.15. Admin Dashboard Analiz ve Raporlama

Batman Üniversitesi Akıllı Kütüphane Sistemi'nin kullanıcı ve admin panellerine ait çeşitli analiz ve yönetim ekranları bulunmaktadır. Kullanıcı panelinde geçmiş sorgular, favoriye alınan veriler ve yanıtlar, geri bildirimler ve puanlamalar, model karşılaştırma geçmişi ve şifre güncelleme özellikleri bulunmaktadır. Admin panelinde ise kullanıcı bazlı veri inceleme, rol güncelleme, şifre sıfırlama, kayıtlı kullanıcıları görüntüleme gibi yönetimsel fonksiyonlar yanında, favori analizleri, geri bildirim ve puan sıralamaları, sorgu sürelerinin analizleri ve çeşitli model performans

metrikleri gibi ayrıntılı raporlar ve analizler yer almaktadır. Bu paneller kullanıcıların performanslarını, sistem kullanım alışkanlıklarını ve verilen yanıtların doğruluğunu ayrıntılı bir şekilde değerlendirme imkânı sunmakta ve kullanıcı yönetimini kolaylaştırmaktadır (Şekil 4.48 ile Şekil 4.57 arası).



Şekil 4.48. Analiz ve raporlama-1



Şekil 4.49. Analiz ve raporlama-2

[logout](#)
[model sec](#)
[register](#)
[user dashboard](#)
[veritabanları](#)
[View less](#)

Admin Menü

Kullanıcı Yönetimi

Kullanıcı Yönetimi

Kullanıcı Bazlı İnceleme

Analiz ve Raporlama

Geri Bildirim Analizi

Favori Analizi

Sorgu Süresi Analizi

Veri Yönetimi

Model Yönetimi

localhost:8501/logout

[https://katalog.batman.edu.tr/yordam/webservis/websevis.php - My Workspace](#)

Deploy

Admin Dashboard

Debug: Şu anki sayfa = Favori Analizi

Yönlendirme: Favori Analizi

Favori Analizi

Favori Tarihi

☐ Veritabanı Favorileri
 ☒ **Favori Favorileri**
☐ Veritabanı İstatistikleri

	kullanici_adi	favori_ismi	favori_etiketi	yanit_metni
0	admin	İsimsiz Favori		Butman Vizyon 2030 Arama Kondensası (Tam metin kitap) Yazar: Ed. İdris Demir, Hüseyin Aydın, Ferhat #
1	admin	İsimsiz Favori		Butman Halk Kültürü Yazar: Ed. İdris Demir, Zehra Görkem Duran Gültekin, İpek tır: Duygu Kızıdemir, Gr
2	admin	İsimsiz Favori		Butman Halk Kültürü Yazar: Ed. İdris Demir, Zehra Görkem Duran Gültekin, İpek tır: Duygu Kızıdemir, Gr
3	admin	İsimsiz Favori		Ölün bugün yarı Batman Yazar: editörler İdris Demir, Ferhat Korkmaz Yıl: 2023 Raf: DS 51/086 - I
4	admin	İsimsiz Favori		Kemal Tahir, İstanbul'da doğdu ve hayatının önemli bir kısmını İstanbul'da geçirdi. Gazetecilik yaparak edebiyat
5	admin	İsimsiz Favori		Notlar Yazar: Kemal Tahir, Kemal Tahir ; yayına hazırlayan, Cengiz Yavaşlı Yıl: 1989 Raf: PL 249.B
6	kutuphane	İsimsiz Favori		Microsoft SQL server 2005 analysis services step by step Yazar: Reed. Jacobson, Reed. Jacobson, Stacia A
7	admin	İsimsiz Favori		Önce İnsan, Önce Betişim : Bilgi ve belge yöneticileri için halkla ilişkiler Yazar: Erol Yılmaz Yıl: 2011
8	suat	İsimsiz Favori		Yahya Kemal Beyatlı : Doğumunun 100. yılında Yazar: Yazar bilgisi yok Yıl: 1994 Raf: PL 248/134 - I

[Ana Sayfa](#)

Şekil 4.50. Analiz ve raporlama-3

admin dashboard

[duyurular](#)
[favorilerim](#)
[hakkımızda](#)
[hizmetlerimiz](#)
[iletişim](#)
[login](#)
[logout](#)
[model sec](#)
[register](#)
[user dashboard](#)
[veritabanları](#)
[View less](#)

Admin Menü

Kullanıcı Yönetimi

Kullanıcı Yönetimi

Kullanıcı Bazlı İnceleme

Analiz ve Raporlama

Geri Bildirim Analizi

Veri Yönetimi

localhost:8501/logout

[https://katalog.batman.edu.tr/yordam/webservis/websevis.php - My Workspace](#)

Deploy

Admin Dashboard

Debug: Şu anki sayfa = Kullanıcı Bazlı İnceleme

Yönlendirme: Geri Bildirim Analizi

Geri Bildirim Analizi

İnceleme Tipi Seçin

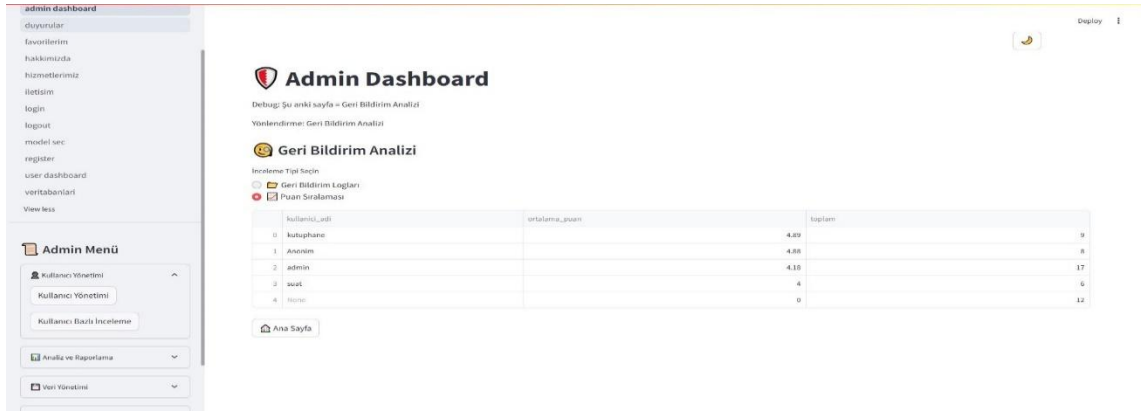
☒ **Geri Bildirim Logları**
☐ Puan Sıralaması

	kullanici_adi	tarih	sorgu	kaynak	puan	hallucination
0	admin	2025-06-25 09:03:48	İdris Demir	YORDAM	None	Yüksek Geçerlilik
1	admin	2025-06-25 09:03:48	İdris Demir	YORDAM	None	Yüksek Geçerlilik
2	admin	2025-06-25 09:03:47	İdris Demir	YORDAM	None	Yüksek Geçerlilik
3	admin	2025-06-25 09:03:47	İdris Demir	YORDAM	None	Yüksek Geçerlilik
4	admin	2025-06-25 09:03:46	İdris Demir	YORDAM	None	Yüksek Geçerlilik
5	admin	2025-06-25 09:03:46	İdris Demir	YORDAM	None	Yüksek Geçerlilik
6	admin	2025-06-25 09:03:45	İdris Demir	YORDAM	None	Yüksek Geçerlilik
7	admin	2025-06-25 09:03:45	İdris Demir	YORDAM	None	Yüksek Geçerlilik
8	admin	2025-06-25 09:03:44	İdris Demir	YORDAM	None	Yüksek Geçerlilik
9	admin	2025-06-25 09:03:44	İdris Demir	YORDAM	None	Yüksek Geçerlilik

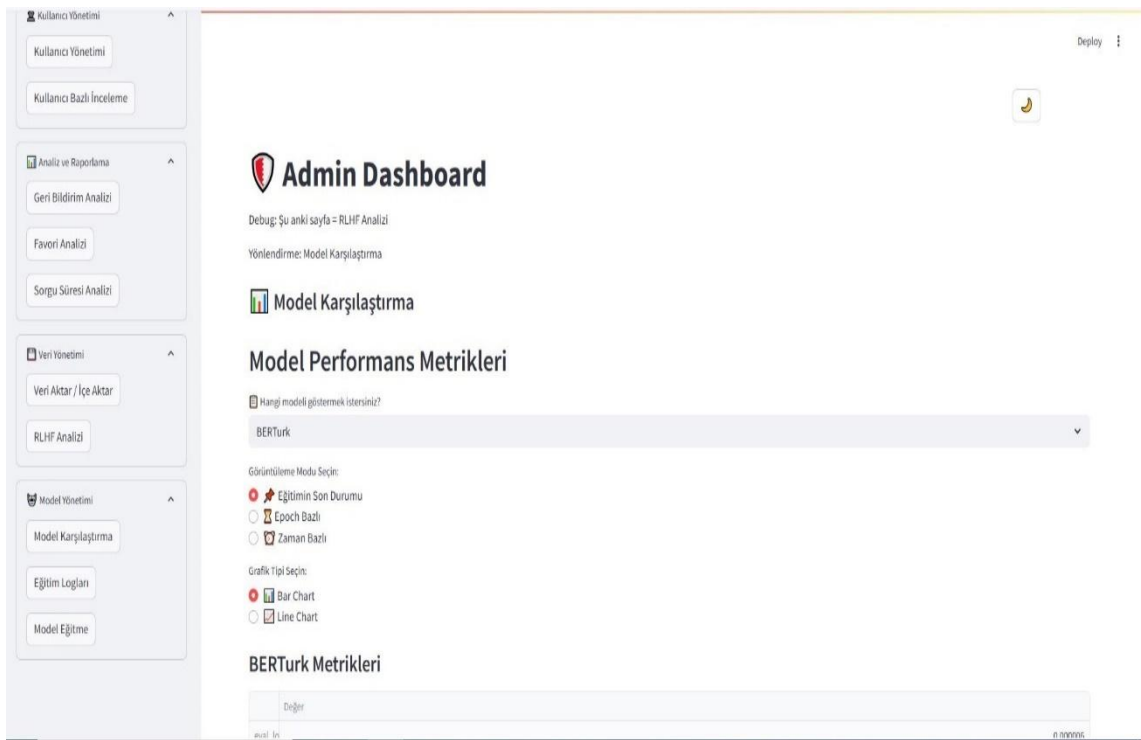
[Ana Sayfa](#)

Şekil 4.51. Analiz ve raporlama-4

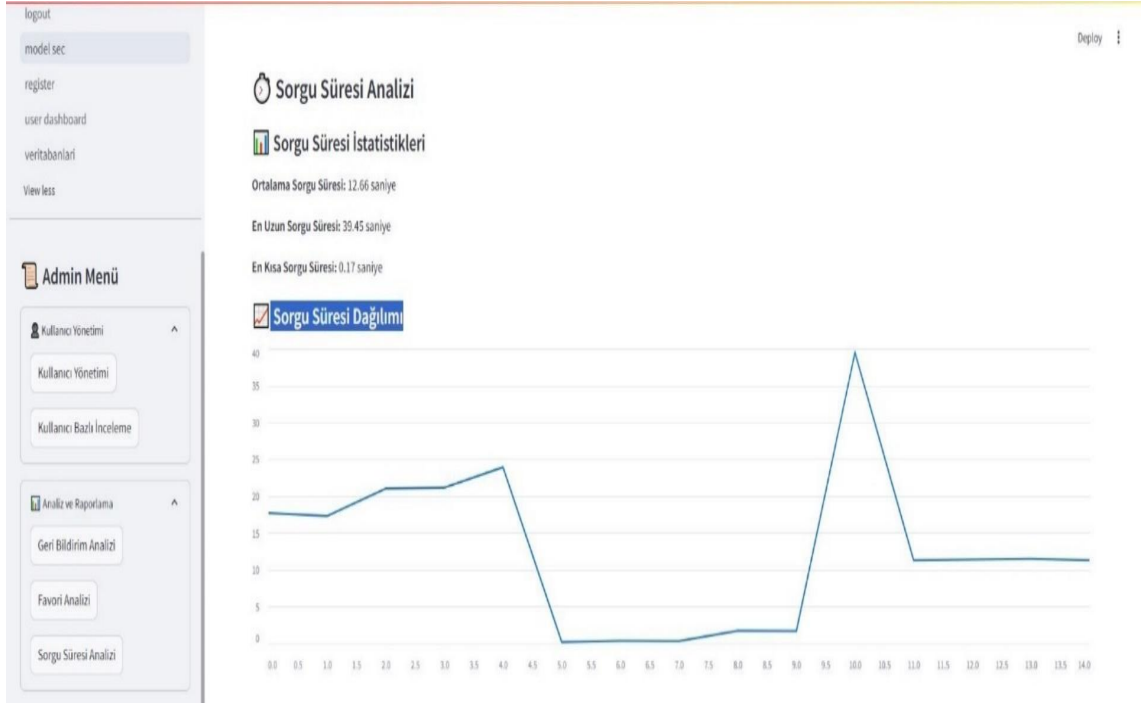
79



Şekil 4.52. Analiz ve raporlama-5



Şekil 4.53. Model yönetimi



Şekil 4.54. Sorgu süresi-1

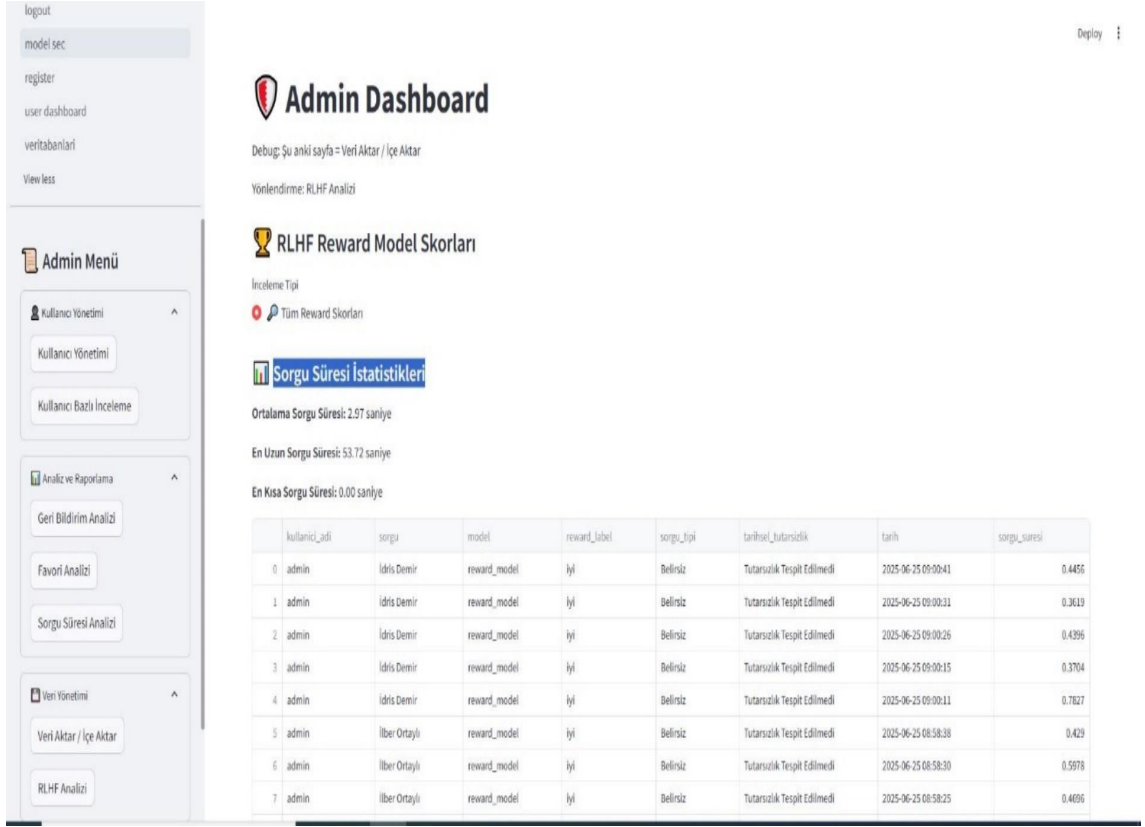
Tüm Sorgu Süreleri

	kullanici_adi	sorgu	kaynak	tarih	sorgu_suresi
0	admin	İlber Ortaylı	Ana Sayfa	2025-06-25 08:56:25	17.6872
1	admin	Kemal Tahir	Ana Sayfa	2025-06-25 08:55:58	17.2619
2	admin	Kemal Tahir	Ana Sayfa	2025-06-25 08:55:10	21.007
3	saat	hayvanlardan insana kitabı	Ana Sayfa	2025-05-20 11:53:47	21.1173
4	saat	hayvanlardan insana kitabı	Ana Sayfa	2025-05-20 11:52:52	23.8945
5	saat	mehmet emin tuğluk	YORDAM	2025-05-20 11:49:04	0.1745
6	saat	mehmet emin tuğluk	YORDAM	2025-05-20 11:48:38	0.3362
7	saat	mehmet emin tuğluk	YORDAM	2025-05-20 11:48:31	0.2845
8	admin	yasar kemal	YORDAM	2025-05-14 12:37:41	1.7053
9	admin	yasar kemal	YORDAM	2025-05-13 05:57:48	1.6669

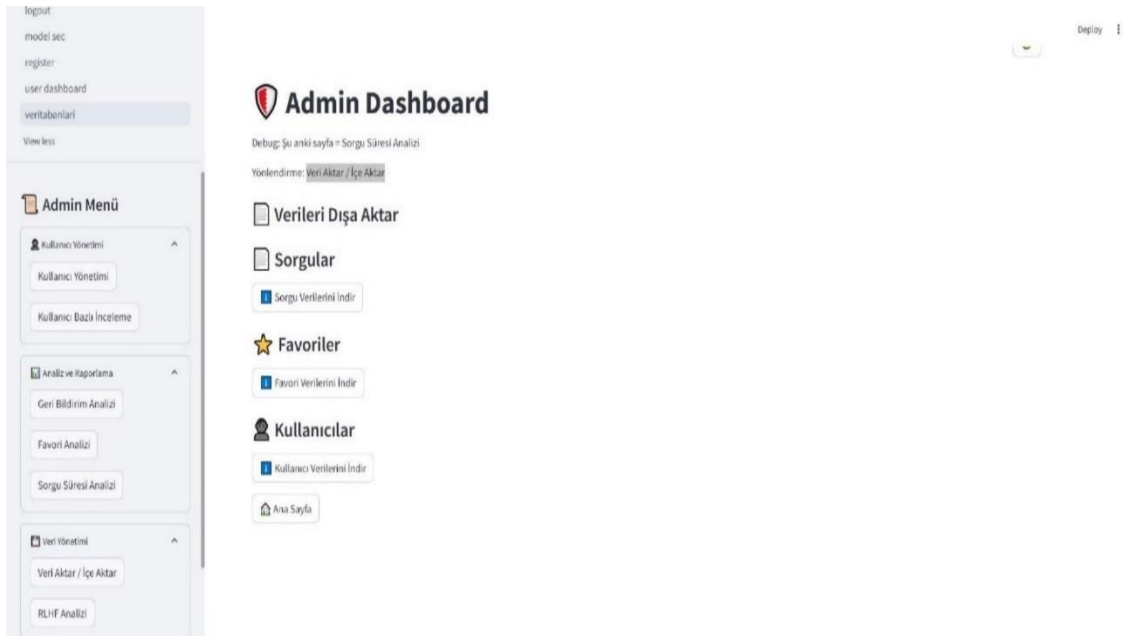
[Sorgu Sürelerini İndir](#)

[Ana Sayfa](#)

Şekil 4.55. Sorgu süresi-2



Şekil 4.56. RLHF reward



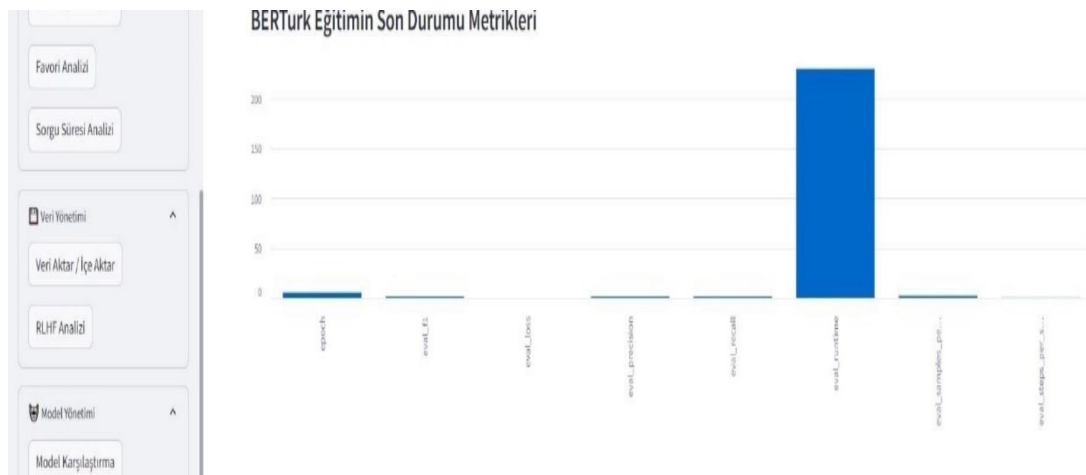
Şekil 4.57. Veri aktarımı

4.16. Model Eğitimi ve Performans Göstergeleri

Beş dil modeli (BERTurk, DistilBERT, ElectraTR, RoBERTa ve SentimentBERT) üzerinde gerçekleştirilen inceleme, eğitim sonu metriklerinin tüm modellerde kusursuz değerlere (precision, recall ve F1 = 1,00; eval_loss $\approx$ 0) ulaştığını ortaya koymuştur. Şekil 58–112 arasında yer alan sütun ve çizgi grafikler incelendiğinde, kayıp değerinin ilk epoch’tan itibaren hızlı bir azalış sergilediği ve üçüncü epoch sonrasında plato evresine girdiği görülmektedir. Bu durum, kullanılan öğrenme oranı zamanlamasının (learning-rate scheduler) model parametrelerini kısa sürede doygunluğa ulaştırdığını göstermektedir. Ancak hem kayıp hem de başarı metriklerinin ideal değerlerde seyretmesi, veri kümesinin sınırlılığına veya homojenliğine işaret etmekte ve aşırı öğrenme olasılığını gündeme getirmektedir.



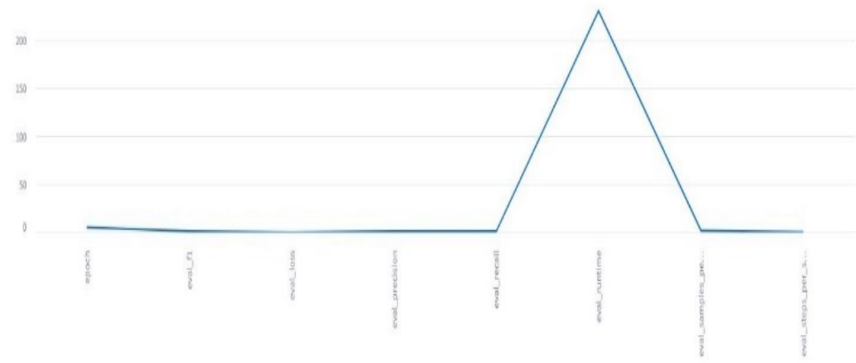
Şekil 4.58. BERTurk metrikleri



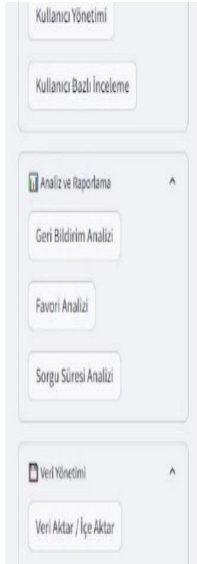
Şekil 4.59. BERTurk eğitim son durum metrikleri-1



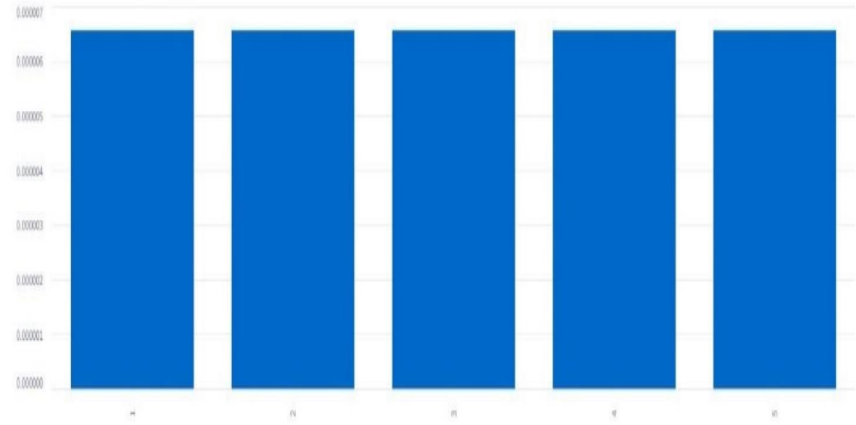
BERTurk Eğitimin Son Durumu Metrikleri



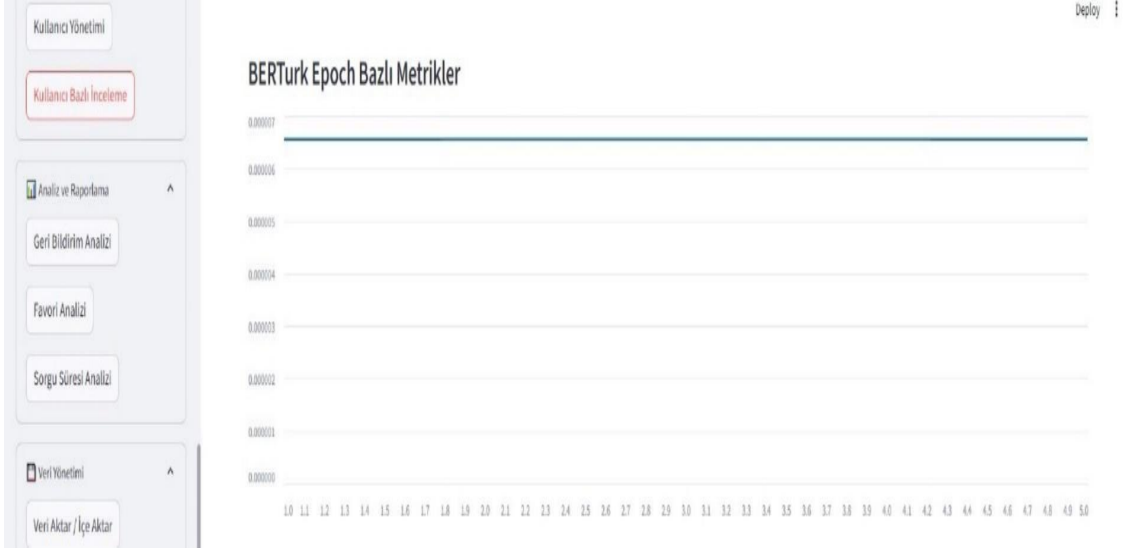
Şekil 4.60. BERTurk eğitim son durum metrikleri-2



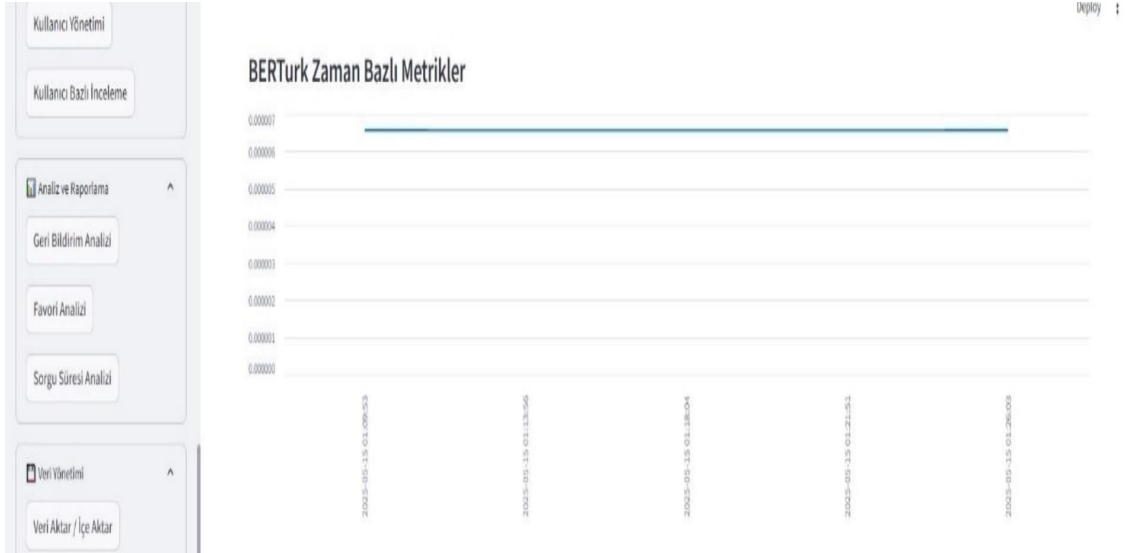
BERTurk Epoch Bazlı Metrikler



Şekil 4.61. BERTurk Epoch bazlı metrikler-1



Şekil 4.62. BERTurk Epoch bazlı metrikler-2



Şekil 4.63. BERTurk zaman bazlı metrikler-1

	2025-05-12_01	2025-05-12_02	2025-05-12_11	2025-05-13_06	egitim_baslangi	egitim_tamamlan	toplam_kayip	egitim_suresi	berturk_train_runtime	berturk_train_s	berturk_train_t	berturk_total_flos	t
2025-05-12 01_09_53_347	10.322000	None	None	None	None	None	None	None	None	None	None	None	None
2025-05-12 09_24_09_781	None	23.672200	None	None	None	None	None	None	None	None	None	None	None
2025-05-12 11_00_22_685	None	None	21.203300	None	None	None	None	None	None	None	None	None	None
2025-05-12 11_03_47_818	None	None	7.925400	None	None	None	None	None	None	None	None	None	None
2025-05-12 11_13_49_792	None	None	12.899400	None	None	None	None	None	None	None	None	None	None
2025-05-13 06_09_58_399	None	None	None	0.260000	None	None	None	None	None	None	None	None	None
2025-05-13 06_14_43_953	None	None	None	0.260000	None	None	None	None	None	None	None	None	None
2025-05-13 06_38_06_089	None	None	None	13.143800	15.565786	20.394732	0.079935	609.546300	617.471100	0.340000	0.089000	5660812800.000000	

Şekil 4.64. BERTurk zaman bazlı metrikler-2

Epoch Bazlı Metrikler											
	loss	grad_norm	learning_rate	epoch	eval_precision	eval_recall	eval_f1	eval_loss	eval_runtime	eval_samples_per_second	eval_steps_per_second
0	1.138800	93.254463	0.000050	1.000000	None	None	None	None	None	None	None
1	1.198300	20.341831	0.000033	2.000000	None	None	None	None	None	None	None
2	1.214700	27.140467	0.000017	3.000000	None	None	None	None	None	None	None
3	1.039700	89.325920	0.000050	1.000000	None	None	None	None	None	None	None
4	1.435600	43.575188	0.000033	2.000000	None	None	None	None	None	None	None
5	1.057500	27.658325	0.000017	3.000000	None	None	None	None	None	None	None
6	1.153200	72.754883	0.000033	1.000000	None	None	None	None	None	None	None
7	1.441600	40.273533	0.000017	2.000000	None	None	None	None	None	None	None
8	1.153300	23.478230	0.000000	3.000000	None	None	None	None	None	None	None
9	1.036100	47.967773	0.000033	1.000000	None	None	None	None	None	None	None

Şekil 4.65. Epoch bazlı metrikler

Tam Log İçeriği											
2025-05-12 07:01:58,347 - INFO - [BERTurk] Eğitim başlama zamanı: 2025-05-12 07:01:58.347544											
2025-05-12 07:02:02,139 - INFO - [BERTurk] Toplam Adım Sayısı: 3											
2025-05-12 07:02:07,171 - INFO - Step 1: {'loss': 1.1388, 'grad_norm': 93.25446319580078, 'learning_rate': 5e-05, 'epoch': 1.0}											
2025-05-12 07:02:10,959 - INFO - Step 2: {'loss': 1.1983, 'grad_norm': 20.34183120727539, 'learning_rate': 3.333333333333333e-05, 'epoch': 2.0}											
2025-05-12 07:02:12,695 - INFO - Step 3: {'loss': 1.2147, 'grad_norm': 27.14046669063477, 'learning_rate': 1.6666666666666667e-05, 'epoch': 3.0}											
2025-05-12 07:02:12,834 - INFO - [BERTurk] Eğitim tamamlanma zamanı: 2025-05-12 07:02:12.834915											
2025-05-12 07:02:12,834 - INFO - [BERTurk] Toplam Kayıp: 1.1839519341786702											
2025-05-12 07:02:12,834 - INFO - [BERTurk] Eğitim Süresi: 10.3226 saniye											
2025-05-12 07:02:12,834 - INFO - [BERTurk] Adım Sayısı: 3											
2025-05-12 07:02:19,587 - INFO - [BERTurk] Eğitim tamamlandı: fine_tuned_model[BERTurk]											
2025-05-12 09:24:09,781 - INFO - [BERTurk] Eğitim başlama zamanı: 2025-05-12 09:24:09.781710											
2025-05-12 09:24:12,149 - INFO - [BERTurk] Toplam Adım Sayısı: 3											
2025-05-12 09:24:23,681 - INFO - Step 1: {'loss': 1.0397, 'grad_norm': 89.32592010498047, 'learning_rate': 5e-05, 'epoch': 1.0}											

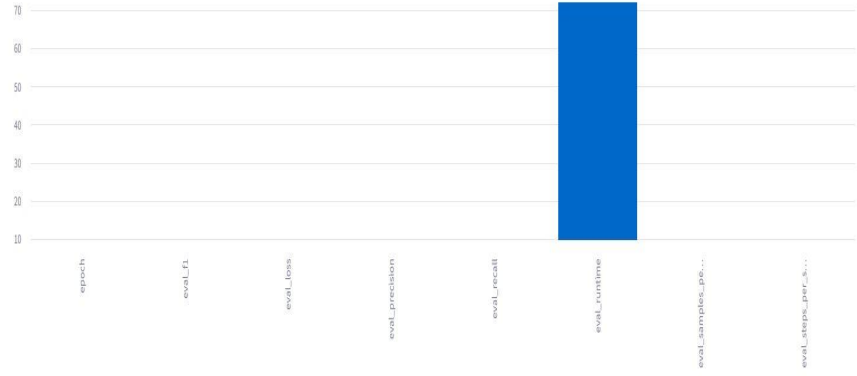
Şekil 4.66. BRTurk tam log içeriği

DistilBERT Metrikleri											
	Değer										
eval_precision	1.000000										
eval_recall	1.000000										
eval_f1	1.000000										
eval_loss	0.000009										
eval_runtime	75.552500										
eval_samples_per_second	5.308000										
eval_steps_per_second	0.675000										
epoch	5.000000										

Şekil 4.67. DistilBERT metrikleri



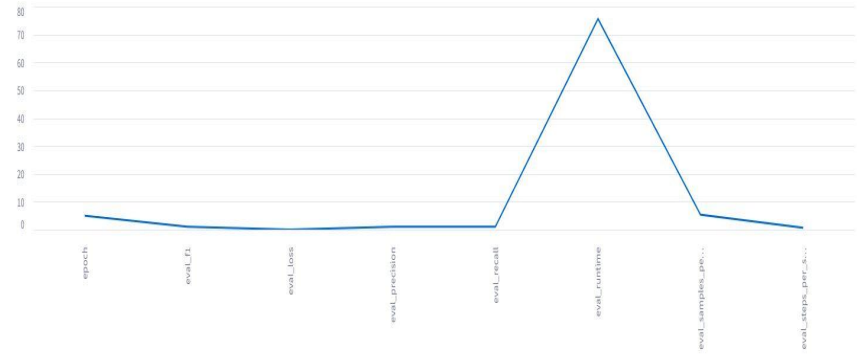
DistilBERT Eğitimin Son Durumu Metrikleri



Şekil 4.68. DistilBERT eğitim son durum metrikleri-1



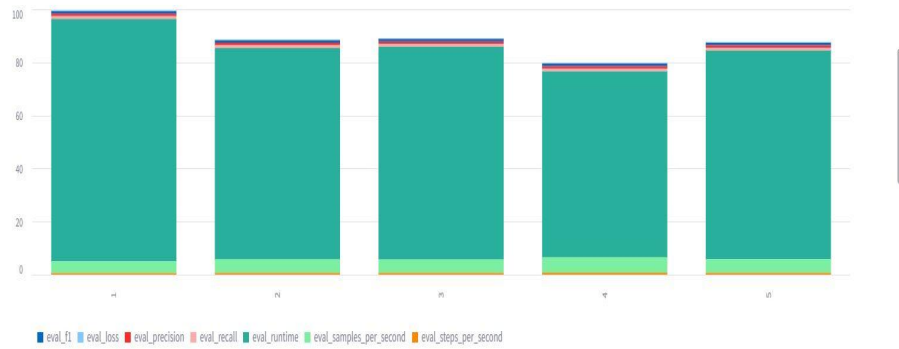
DistilBERT Eğitimin Son Durumu Metrikleri



Şekil 4.69. DistilBERT eğitim son durum metrikleri-2



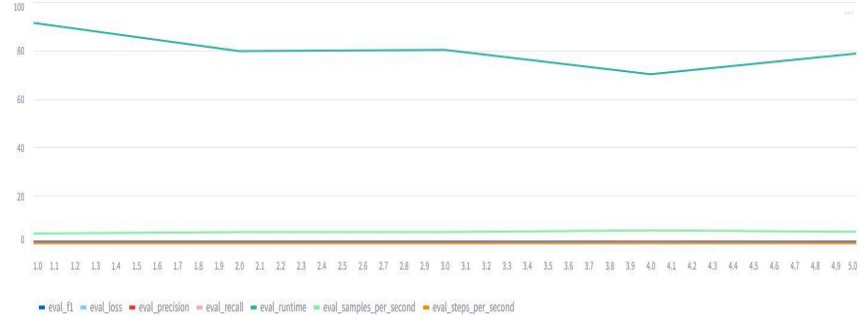
DistilBERT Epoch Bazlı Metrikler



Şekil 4.70. DistilBERT Epoch bazlı metrikler-1



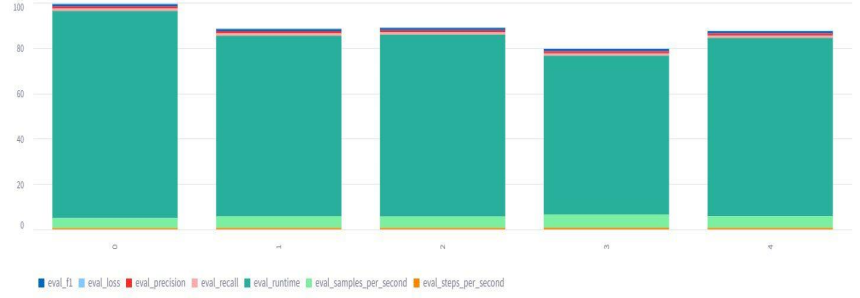
DistilBERT Epoch Bazlı Metrikler



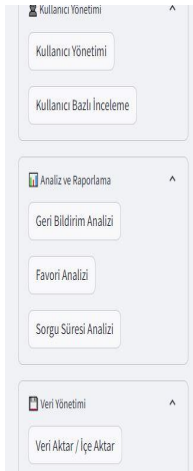
Şekil 4.71. DistilBERT Epoch bazlı metrikler-2



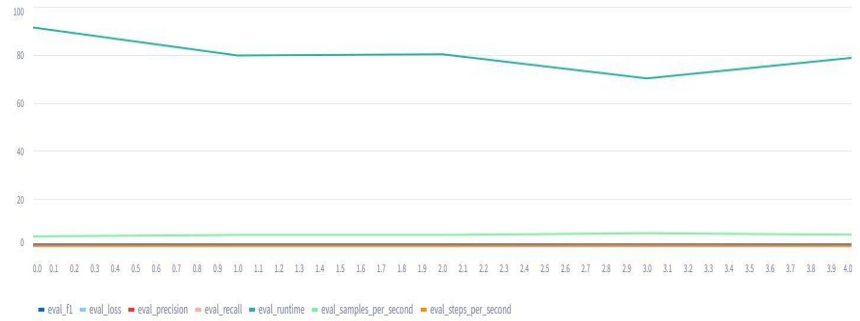
DistilBERT Zaman Bazlı Metrikler



Şekil 4.72. DistilBERT zaman bazlı metrikler-1



DistilBERT Zaman Bazlı Metrikler



Şekil 4.73. DistilBERT zaman bazlı metrikler-2

Kullanıcı Yönetimi Kullanıcı Yönetimi Kullanıcı Bazlı İnceleme Analiz ve Raporlama Geri Bildirim Analizi Favori Analizi Sorgu Süresi Analizi Veri Yönetimi Veri Aktar / İçer Aktar RLHF Analizi Model Yönetimi Model Karşılaştırma Eğitim Logları	Deploy Eğitim Logları Log dosyalarını seçin: DistilBERT(training.log) Tespit edilen kodlama: utf-8 (Güven: %99.00) Debug Mesajları <table> <tr> <th>Tür</th><th>Mesaj</th></tr> <tr> <td>0</td><td>Zaman Damgası</td></tr> <tr> <td>1</td><td>Metrik</td></tr> <tr> <td>2</td><td>Epoch Verisi</td></tr> <tr> <td>3</td><td>Epoch Verisi</td></tr> <tr> <td>4</td><td>Epoch Verisi</td></tr> <tr> <td>5</td><td>Epoch Verisi</td></tr> <tr> <td>6</td><td>Metrik</td></tr> <tr> <td>7</td><td>Metrik</td></tr> <tr> <td>8</td><td>Metrik</td></tr> <tr> <td>9</td><td>Zaman Damgası</td></tr> <tr> <td>10</td><td>Metrik</td></tr> </table>	Tür	Mesaj	0	Zaman Damgası	1	Metrik	2	Epoch Verisi	3	Epoch Verisi	4	Epoch Verisi	5	Epoch Verisi	6	Metrik	7	Metrik	8	Metrik	9	Zaman Damgası	10	Metrik
Tür	Mesaj																								
0	Zaman Damgası																								
1	Metrik																								
2	Epoch Verisi																								
3	Epoch Verisi																								
4	Epoch Verisi																								
5	Epoch Verisi																								
6	Metrik																								
7	Metrik																								
8	Metrik																								
9	Zaman Damgası																								
10	Metrik																								

Şekil 4.74. DistilBERT eğitim logları

Kullanıcı Yönetimi

Kullanıcı Yönetimi

Kullanıcı Bazlı İnceleme

Analiz ve Raporlama

Geri Bildirim Analizi

Favori Analizi

Sorgu Süresi Analizi

Deploy

Eğitim Metrikleri (Zaman Bazlı)

	2025-05-12_01	2025-05-12_6	2025-05-12_11	2025-05-13_06	egitim_baslam	egitim_tamamlan	toplam_kayip	egitim_suresi	distilbert_train_runtime	distilbert_train_samples_per_second	distilbert_t
2025-05-12 07_02_19_904	4.120700	None	None	None	None	None	None	None	None	None	None
2025-05-12 09_24_43_888	None	5.700100	None	None	None	None	None	None	None	None	None
2025-05-12 11_00_52_831	None	None	4.955900	None	None	None	None	None	None	None	None
2025-05-12 11_04_03_096	None	None	4.572400	None	None	None	None	None	None	None	None
2025-05-13 06_38_30_342	None	None	None	5.288600	25.887338	37.618344	0.122493	305.781800	259.282400	0.810000	

Şekil 4.75. DistilBERT eğitim metrikleri zaman bazlı

Kullanıcı Yönetimi

Kullanıcı Yönetimi

Kullanıcı Bazlı İnceleme

Analiz ve Raporlama

Geri Bildirim Analizi

Favori Analizi

Sorgu Süresi Analizi

Veri Yönetimi

Veri Aktar / İçer Aktar

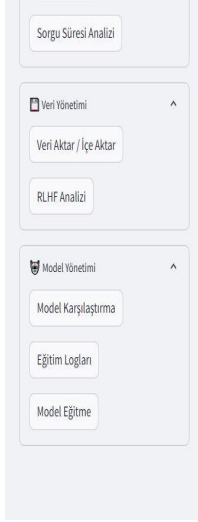
RLHF Analizi

Deploy

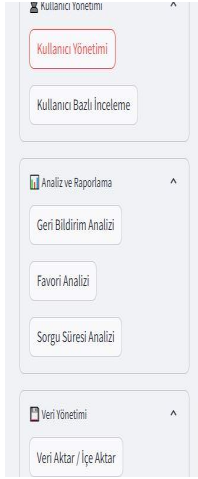
Epoch Bazlı Metrikler

	loss	grad_norm	learning_rate	epoch	eval_precision	eval_recall	eval_f1	eval_loss	eval_runtime	eval_samples_per_second	eval_steps_per_second
0	1.120700	2.504564	0.000050	1.000000	None	None	None	None	None	None	None
1	1.061300	2.836698	0.000033	2.000000	None	None	None	None	None	None	None
2	1.032300	2.680250	0.000017	3.000000	None	None	None	None	None	None	None
3	1.120700	2.504564	0.000050	1.000000	None	None	None	None	None	None	None
4	1.061300	2.836698	0.000033	2.000000	None	None	None	None	None	None	None
5	1.032300	2.680250	0.000017	3.000000	None	None	None	None	None	None	None
6	1.171900	2.634158	0.000033	1.000000	None	None	None	None	None	None	None
7	1.128600	2.570486	0.000017	2.000000	None	None	None	None	None	None	None
8	1.009500	2.439574	0.000000	3.000000	None	None	None	None	None	None	None
9	1.171900	2.634158	0.000033	1.000000	None	None	None	None	None	None	None

Şekil 4.76. DistilBERT Epoch bazlı metrikler



Şekil 4.77. DistilBERT tam log içeriği

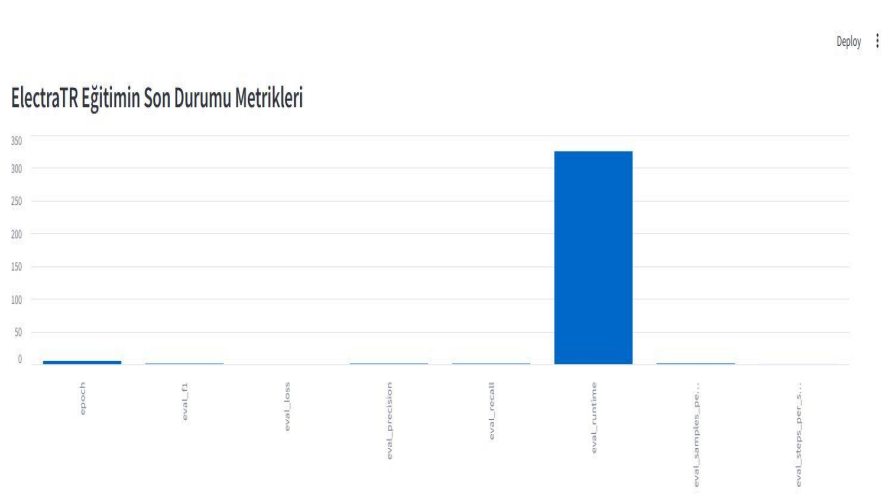
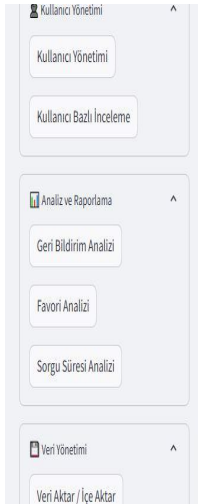


ElectraTR Metrikleri

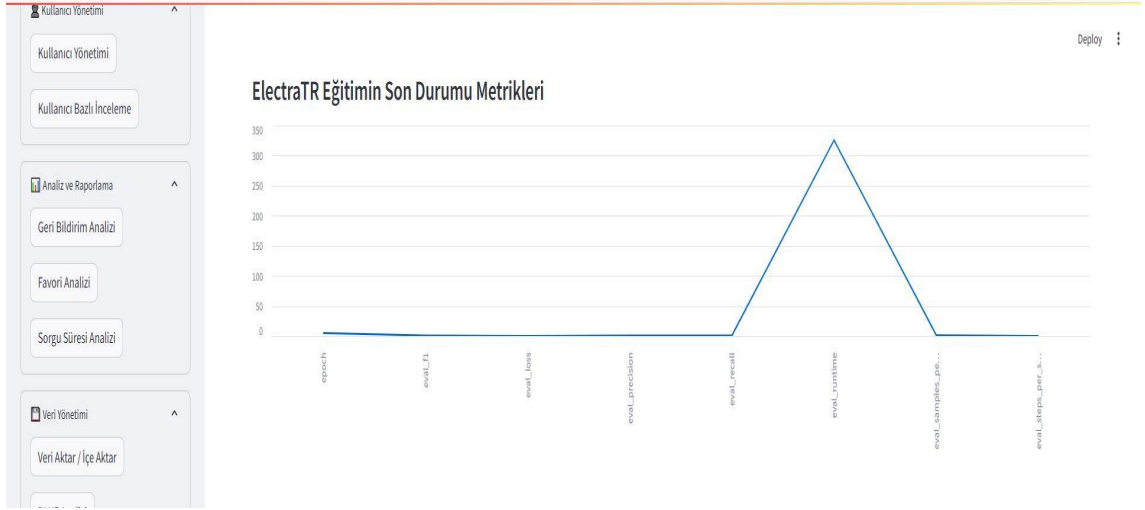
	Değer
eval_loss	0.000027
eval_precision	1.000000
eval_recall	1.000000
eval_f1	1.000000
eval_runtime	325.418400
eval_samples_per_second	1.232000
eval_steps_per_second	0.157000
epoch	5.000000

Deploy

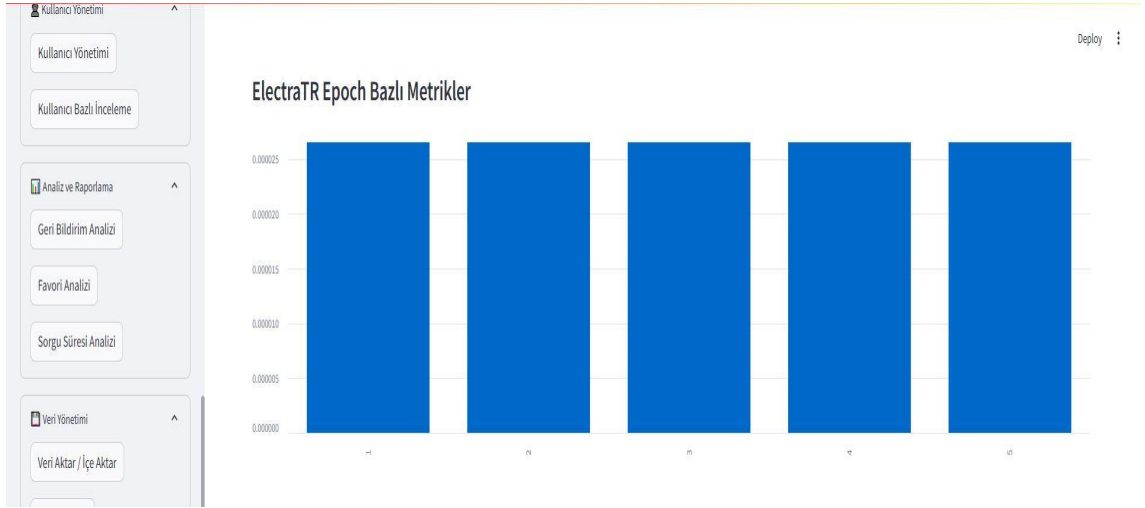
Şekil 4.78. ElectraTR metrikleri



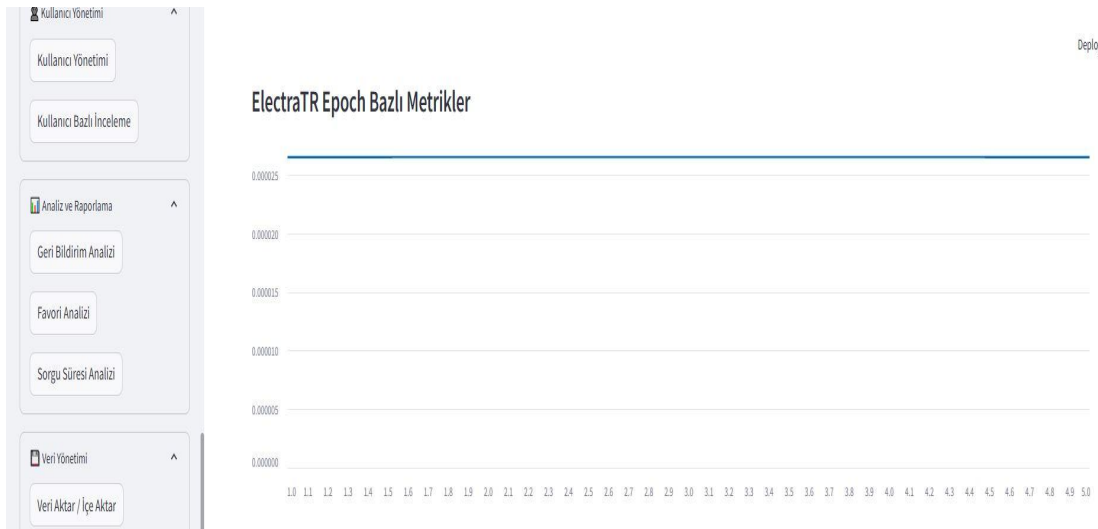
Şekil 4.79. ElectraTR eğitimi son durum metrikleri-1



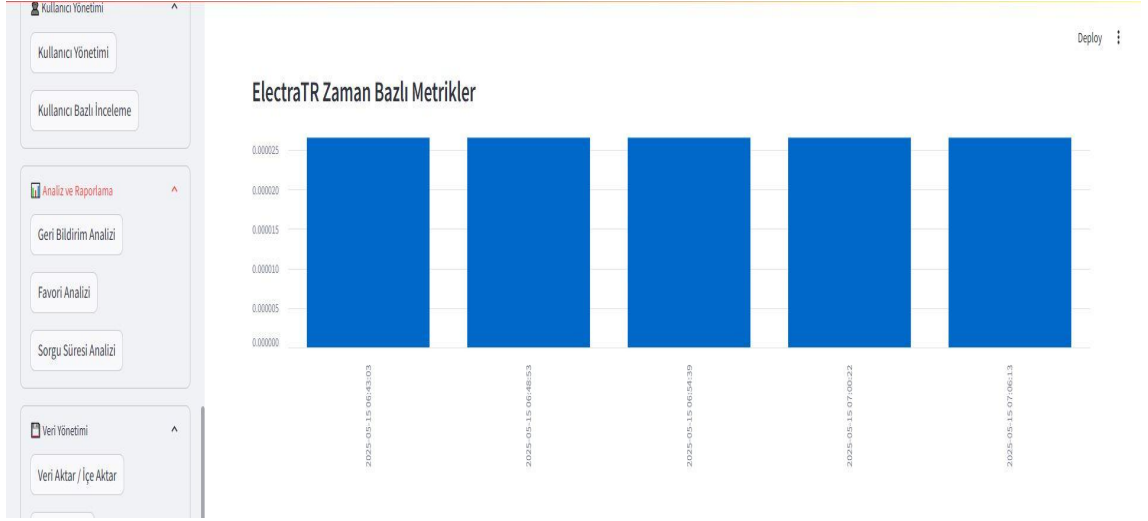
Şekil 4.80. ElectraTR eğitimi son durum metrikleri-2



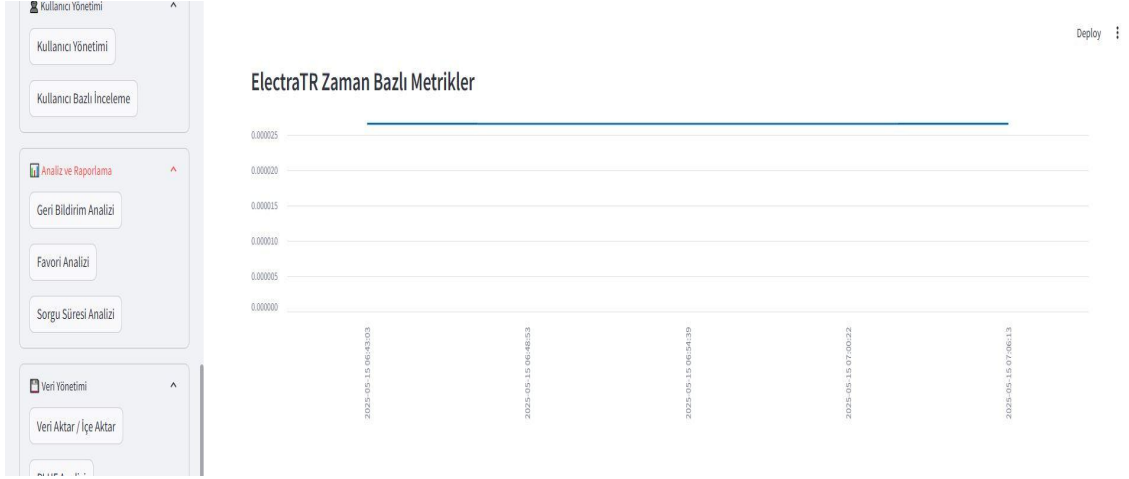
Şekil 4.81. ElectraTR Epoch bazlı metrikler-1



Şekil 4.82. ElectraTR Epoch bazlı metrikler-2



Şekil 4.83. ElectraTR zaman bazlı metrikler-1



Şekil 4.84. ElectraTR zaman bazlı metrikler-2

Eğitim Logları

Log dosyasını seçin: ElectraTR(training.log)

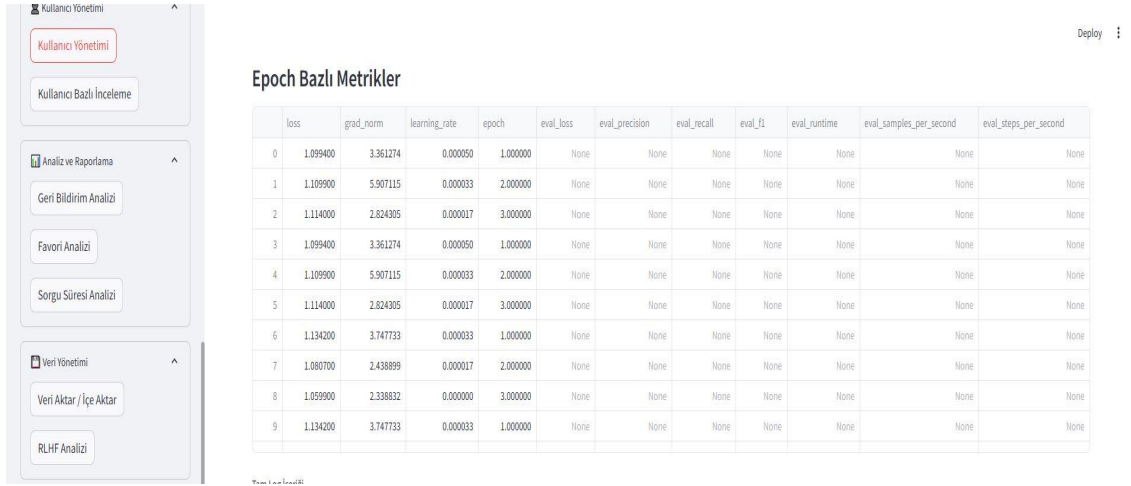
Tespit edilen kodlama: utf-8 (Güven: %99.00)

Tarih	Mesaj
0	Zaman Damgası - Yeni zaman damgası bulundu - 2025-05-12 07_02_29_873 (Satır 1)
1	Metrik - Metrik eklendi - 2025-05-12_07_29_873090 (Satır 1)
2	Epoch Verisi - Epoch verisi bulundu - [loss: 1.0994, 'grad_norm': 3.36127376562965, 'learning_rate': 5e-05, 'epoch': 1.0] (Satır 3)
3	Epoch Verisi - Epoch verisi bulundu - [loss: 1.1099, 'grad_norm': 5.907115459442139, 'learning_rate': 3.3333333333333335e-05, 'epoch': 2.0] (Satır 4)
4	Epoch Verisi - Epoch verisi bulundu - [loss: 1.114, 'grad_norm': 2.824804819107057, 'learning_rate': 1.666666666666667e-05, 'epoch': 3.0] (Satır 5)
5	Metrik - Metrik eklendi - 2025-05-12_07_41.167882 (Satır 6)
6	Metrik - Metrik eklendi - 2025-05-12_07_1.1077938477198284 (Satır 7)
7	Metrik - Metrik eklendi - 2025-05-12_07_10.3980 (Satır 8)
8	Zaman Damgası - Yeni zaman damgası bulundu - 2025-05-12 09_24_57_222 (Satır 11)
9	Metrik - Metrik eklendi - 2025-05-12_09_57.222283 (Satır 11)

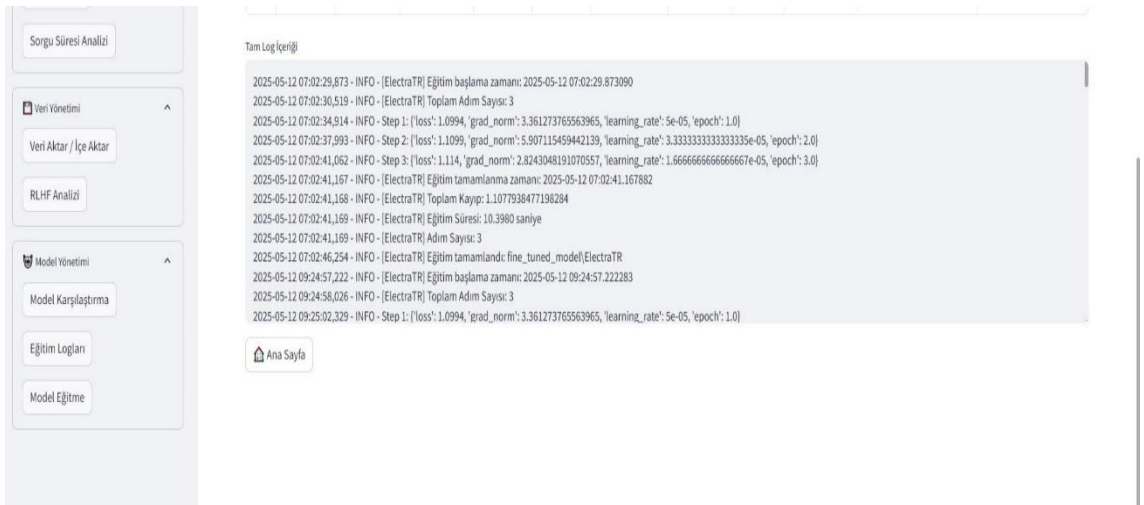
Şekil 4.85. ElectraTR eğitim logları



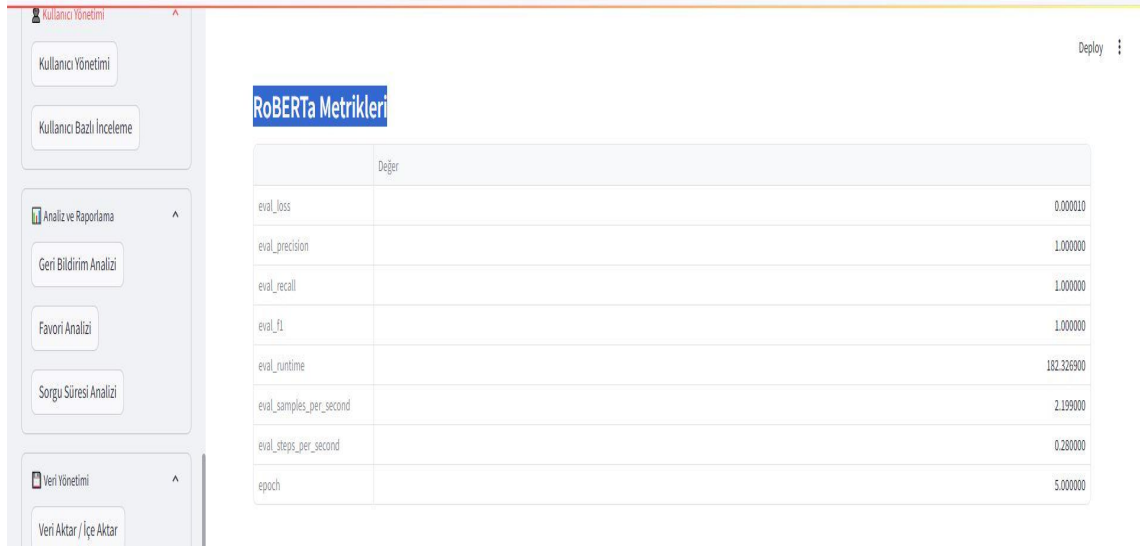
Şekil 4.86. ElectraTR eğitim metrikleri zaman bazlı



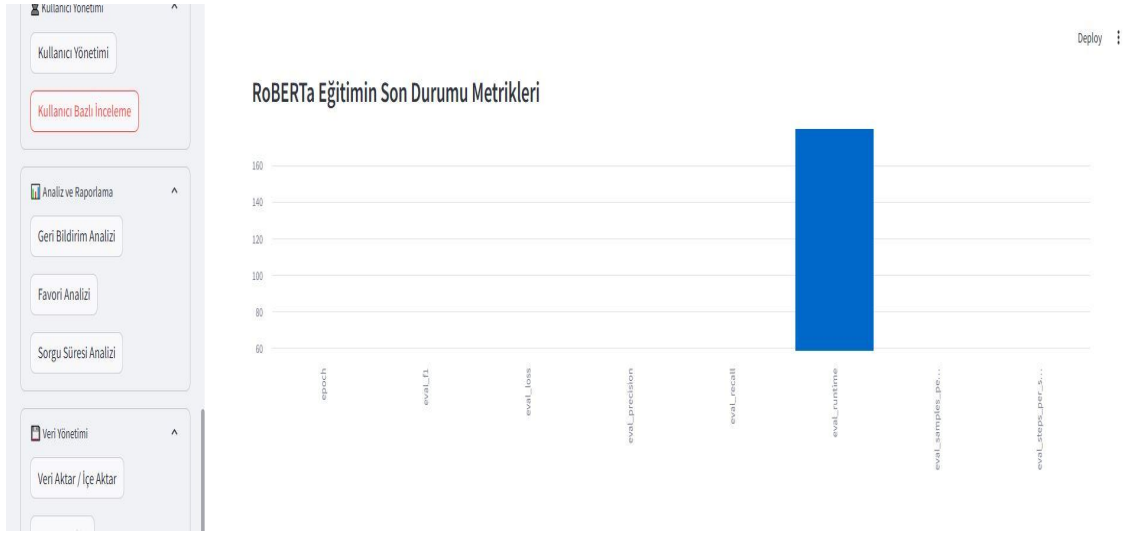
Şekil 4.87. ElectraTR Epoch bazlı metrikler



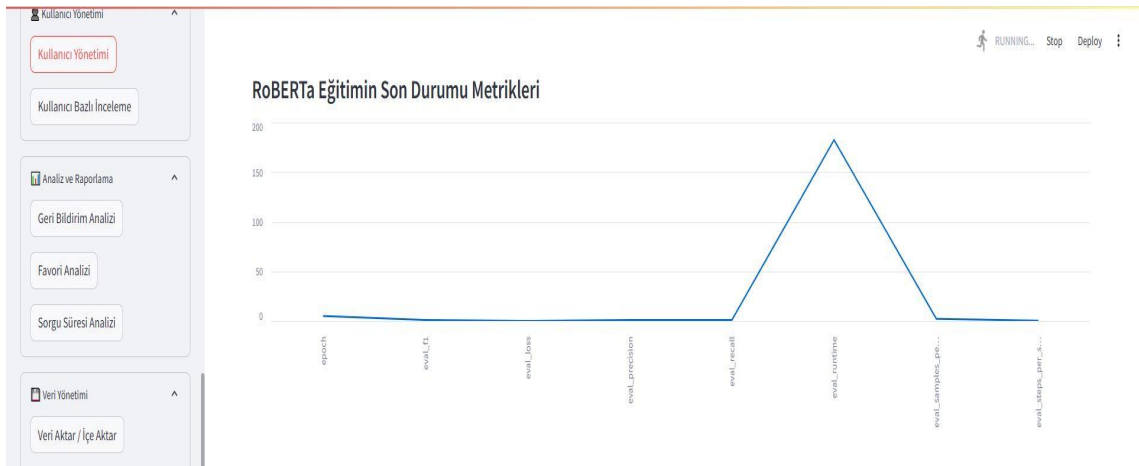
Şekil 4.88. ElectraTR tam log içeriği



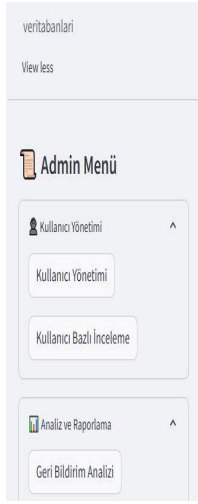
Şekil 4.89. RoBERTa metrikleri



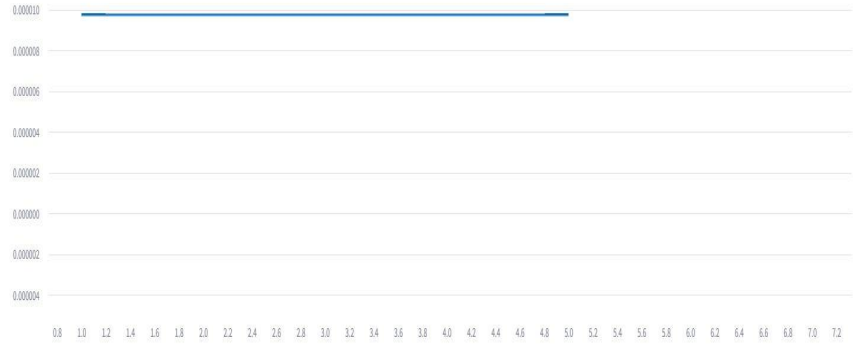
Şekil 4.90. RoBERTa eğitimin son durumu metrikleri-1



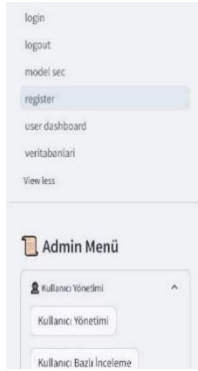
Şekil 4.91. RoBERTa eğitimin son durum metrikleri-2



RoBERTa Epoch Bazlı Metrikler



Şekil 4.92. RoBERTa Epoch bazlı metrikler-1



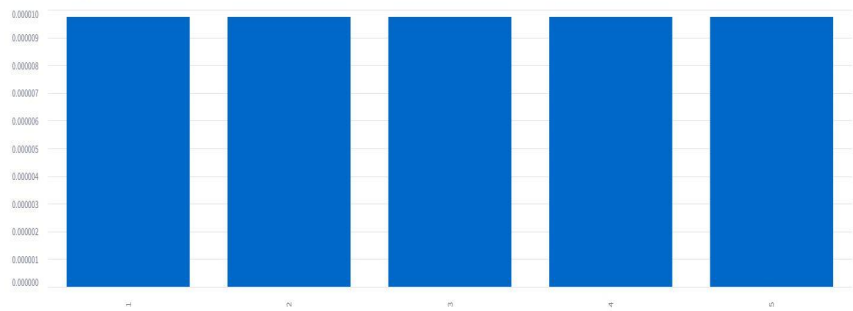
RoBERTa Zaman Bazlı Metrikler



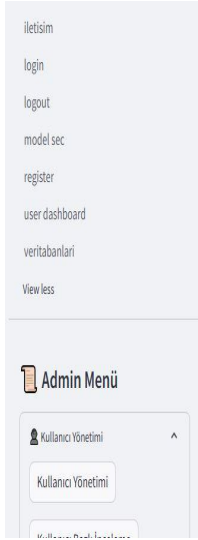
Şekil 4.93. RoBERTa Epoch bazlı metrikler-2



RoBERTa Epoch Bazlı Metrikler



Şekil 4.94. RoBERTa zaman bazlı metrikler-1



Tam Log İçeriği

```

2025-05-12 07:02:46,469 - INFO - [RoBERTa] Eğitim başlama zamanı: 2025-05-12 07:02:46.469556
2025-05-12 07:02:47,864 - INFO - [RoBERTa] Toplam Adım Sayısı: 3
2025-05-12 07:02:51,803 - INFO - Step 1: {'loss': 1.0661, 'grad_norm': 1.8520723581314087, 'learning_rate': 5e-05, 'epoch': 1.0}
2025-05-12 07:02:54,786 - INFO - Step 2: {'loss': 1.0735, 'grad_norm': 2.998135566711426, 'learning_rate': 3.3333333333333335e-05, 'epoch': 2.0}
2025-05-12 07:02:57,826 - INFO - Step 3: {'loss': 1.0931, 'grad_norm': 10.9091157913208, 'learning_rate': 1.6666666666666667e-05, 'epoch': 3.0}
2025-05-12 07:02:57,930 - INFO - [RoBERTa] Eğitim tamamlanma zamanı: 2025-05-12 07:02:57.930715
2025-05-12 07:02:57,930 - INFO - [RoBERTa] Toplam Kayıp: 1.077553782780964
2025-05-12 07:02:57,930 - INFO - [RoBERTa] Eğitim Süresi: 9.8245 saniye
2025-05-12 07:02:57,930 - INFO - [RoBERTa] Adım Sayısı: 3
2025-05-12 07:03:03,599 - INFO - [RoBERTa] Eğitim tamamlandı: fine_tuned_model(RoBERTa)
2025-05-12 09:25:15,410 - INFO - [RoBERTa] Eğitim başlama zamanı: 2025-05-12 09:25:15.410780
2025-05-12 09:25:19,261 - INFO - [RoBERTa] Toplam Adım Sayısı: 3
2025-05-12 09:25:27,150 - INFO - Step 1: {'loss': 1.0661, 'grad_norm': 1.8520723581314087, 'learning_rate': 5e-05, 'epoch': 1.0}

```

Ana Sayfa

Şekil 4.98. RoBERTa tam log içeriği

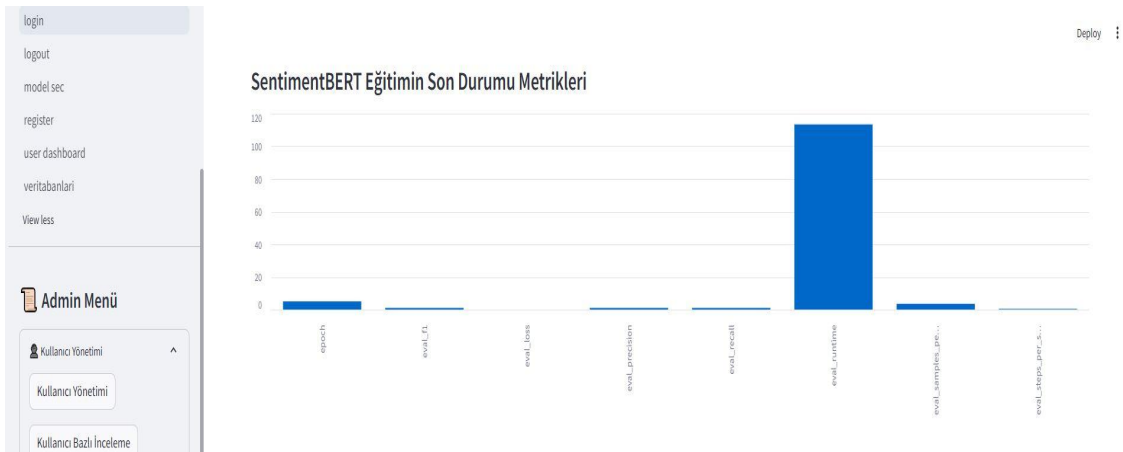
login logout model sec register user dashboard veritabanlari View less Admin Menü Kullanıcı Yönetimi Kullanıcı Yönetimi

SentimentBERT Metrikleri

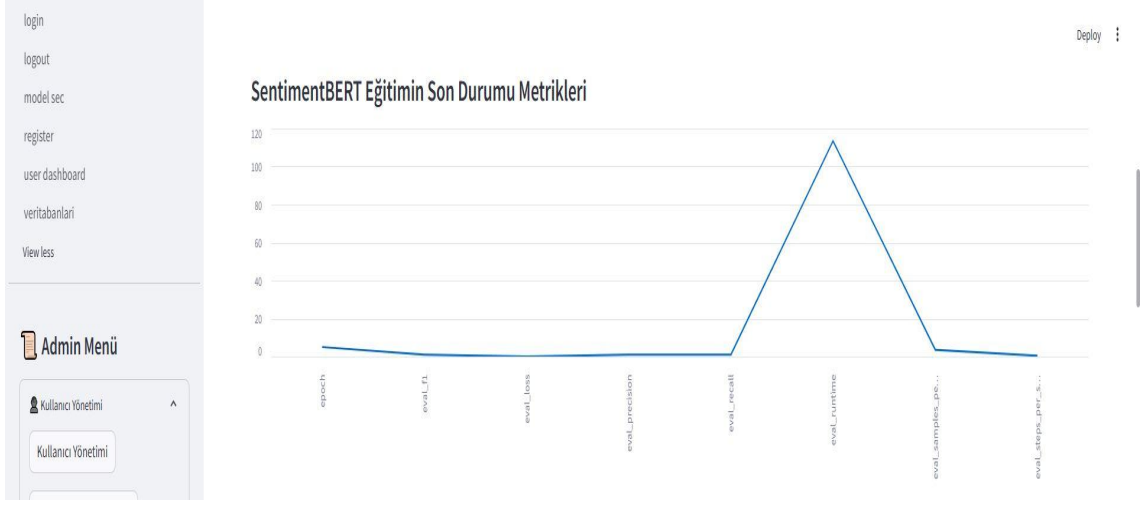
	Değer
eval_loss	0.000014
eval_precision	1.000000
eval_recall	1.000000
eval_f1	1.000000
eval_runtime	113.403100
eval_samples_per_second	3.536000
eval_steps_per_second	0.450000
epoch	5.000000

Deploy

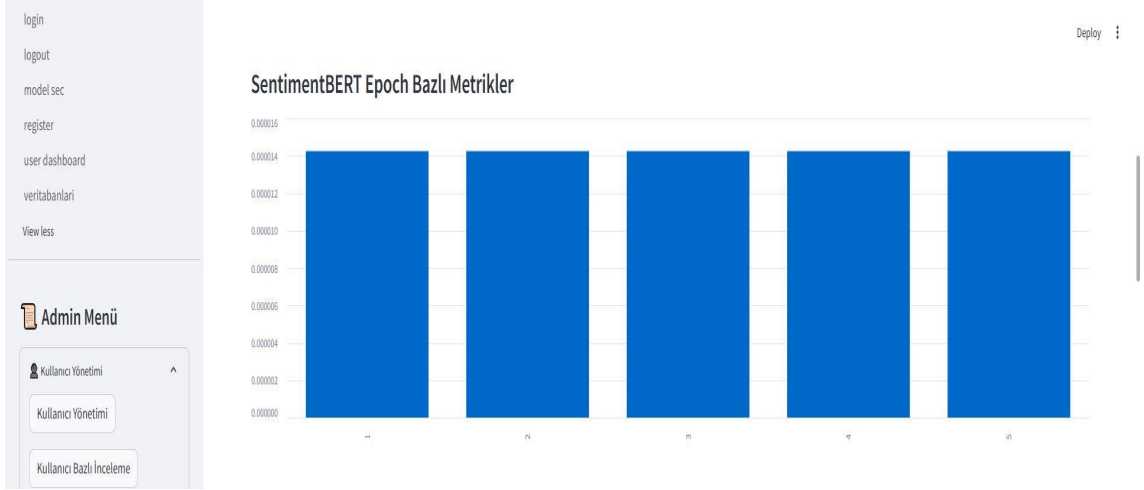
Şekil 4.99. SentimentBERT metrikleri



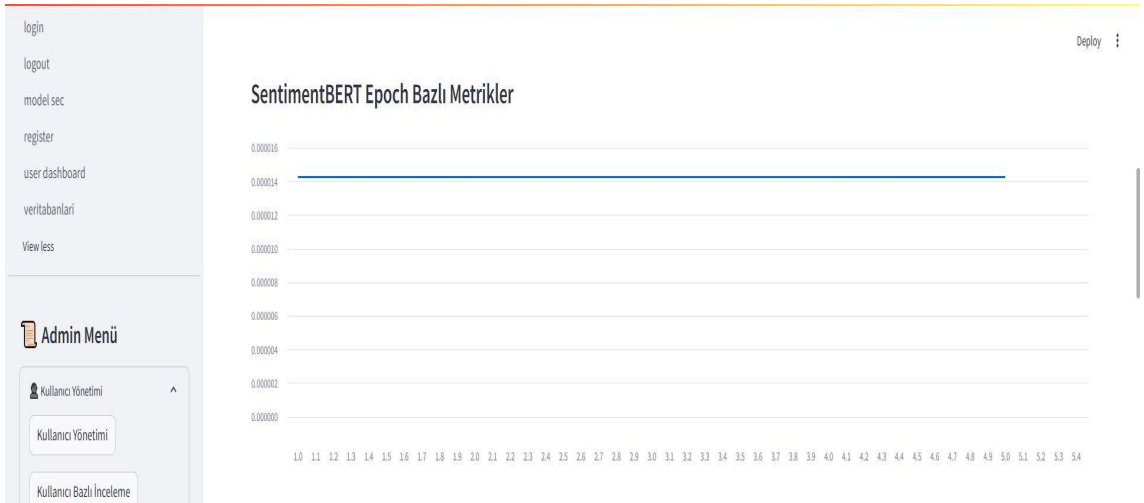
Şekil 4.100. SentimentBERT eğitim son durum metrikleri-1



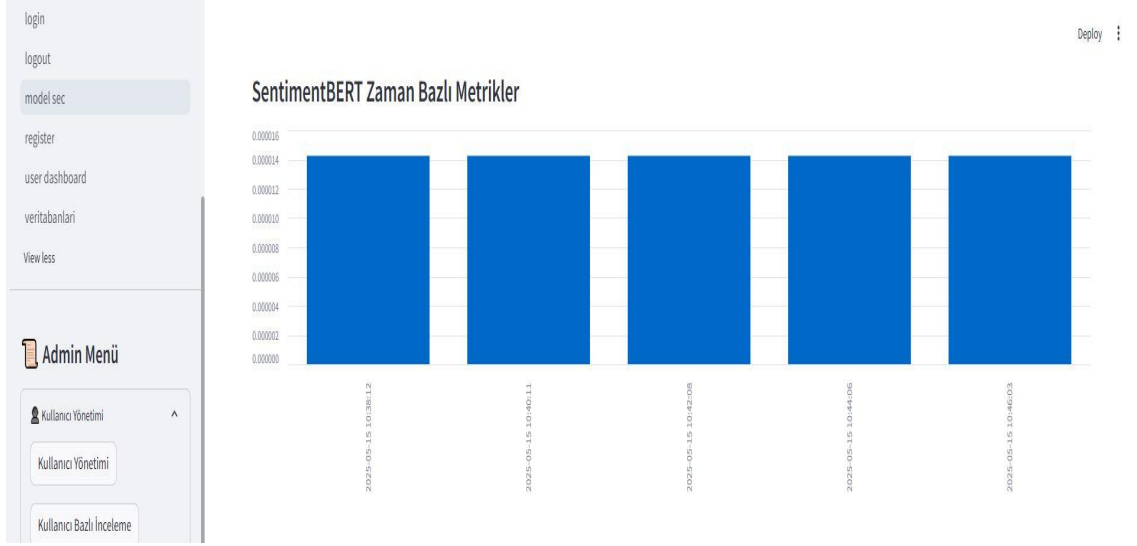
Şekil 4.101. SentimentBERT eğitim son durum metrikleri-2



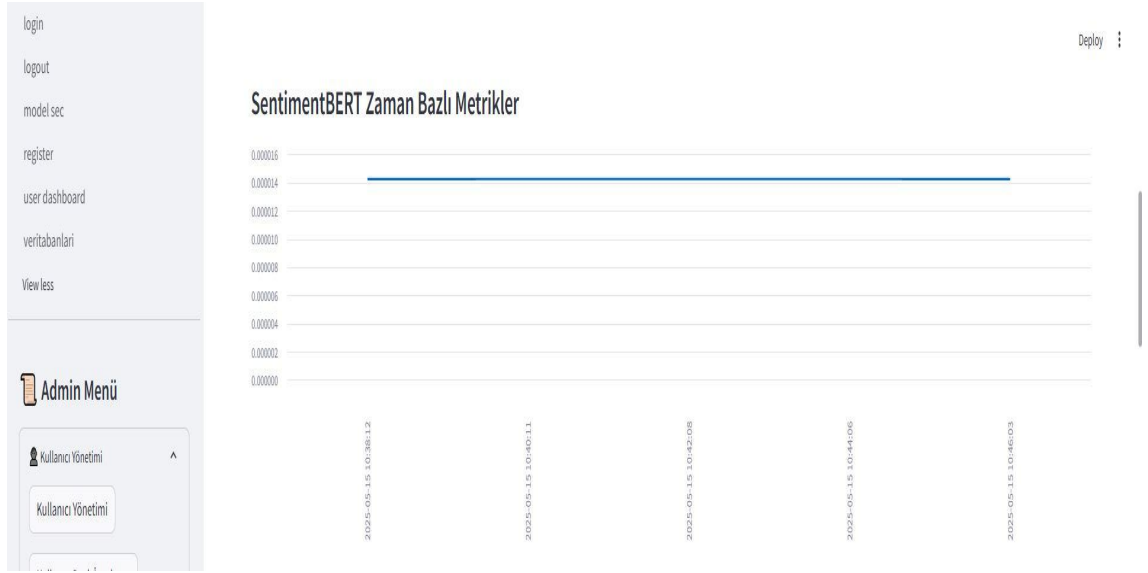
Şekil 4.102. SentimentBERT Epoch bazlı metrikler-1



Şekil 4.103. SentimentBERT Epoch bazlı metrikler-2



Şekil 4.104.SentimentBERT zaman bazlı metrikler-1



Şekil 4.105. SentimentBERT zaman bazlı metrikler-2

streamlit app
admin dashboard
duyurular
favorilerim
hakkımızda
hizmetlerimiz
İletişim
login
logout
model sec
register
user dashboard
veritabanları
View less

Admin Menü

- Kullanıcı Yönetimi
- Kullanıcı Yönetimi

Eğitim Logları

Log dosyasını seçin:
SentimentBERT\training.log

Tespit edilen kodlama: utf-8 (Güven: %99.00)

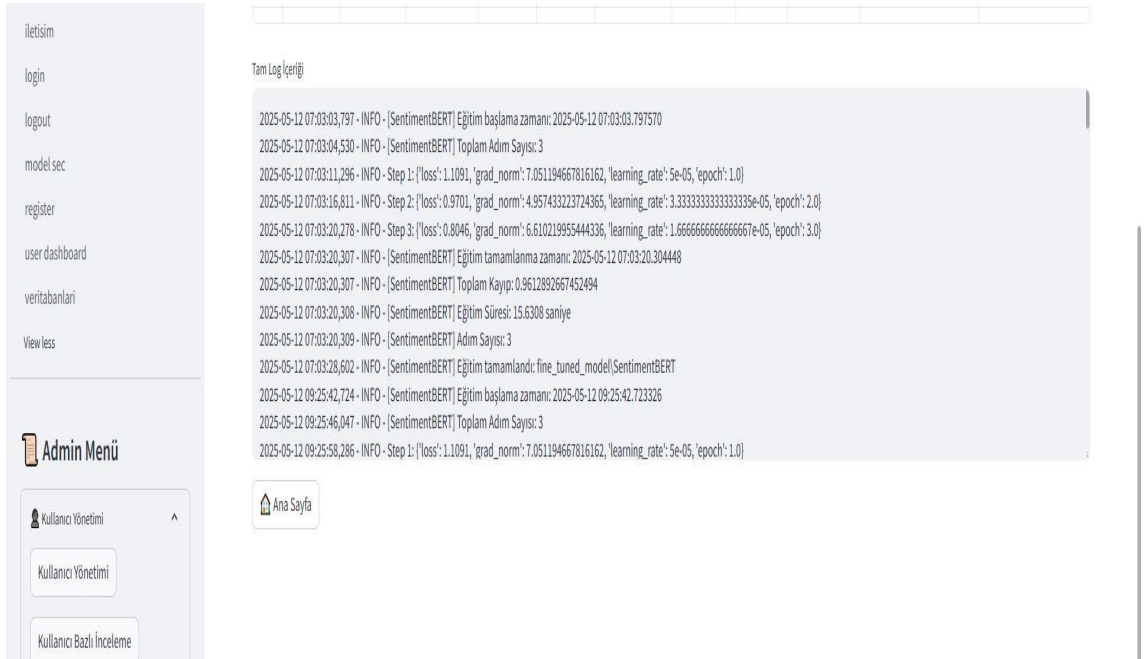
Detaylı Mesajlar

Tür	Mesaj
0 Zaman Damgası	Yeni zaman damgası bulundu - 2025-05-12 07:03:797 (Satır 1)
1 Metrik	Metrik eklendi - 2025-05-12 07:03:797570 (Satır 1)
2 Epoch Verisi	Epoch verisi bulundu - {'loss': 1.1091, 'grad_norm': 7.05134667816362, 'learning_rate': 5e-05, 'epoch': 1.0} (Satır 3)
3 Epoch Verisi	Epoch verisi bulundu - {'loss': 0.9701, 'grad_norm': 4.957433223724965, 'learning_rate': 3.3333333333333335e-05, 'epoch': 2.0} (Satır 4)
4 Epoch Verisi	Epoch verisi bulundu - {'loss': 0.8046, 'grad_norm': 6.810219955444336, 'learning_rate': 1.6666666666666667e-05, 'epoch': 3.0} (Satır 5)
5 Metrik	Metrik eklendi - 2025-05-12 07:20:304448 (Satır 6)
6 Metrik	Metrik eklendi - 2025-05-12 07:03:642892667452494 (Satır 7)
7 Metrik	Metrik eklendi - 2025-05-12 07:15:6308 (Satır 8)
8 Zaman Damgası	Yeni zaman damgası bulundu - 2025-05-12 09:45_42_724 (Satır 11)
9 Metrik	Metrik eklendi - 2025-05-12 09:42_723226 (Satır 11)

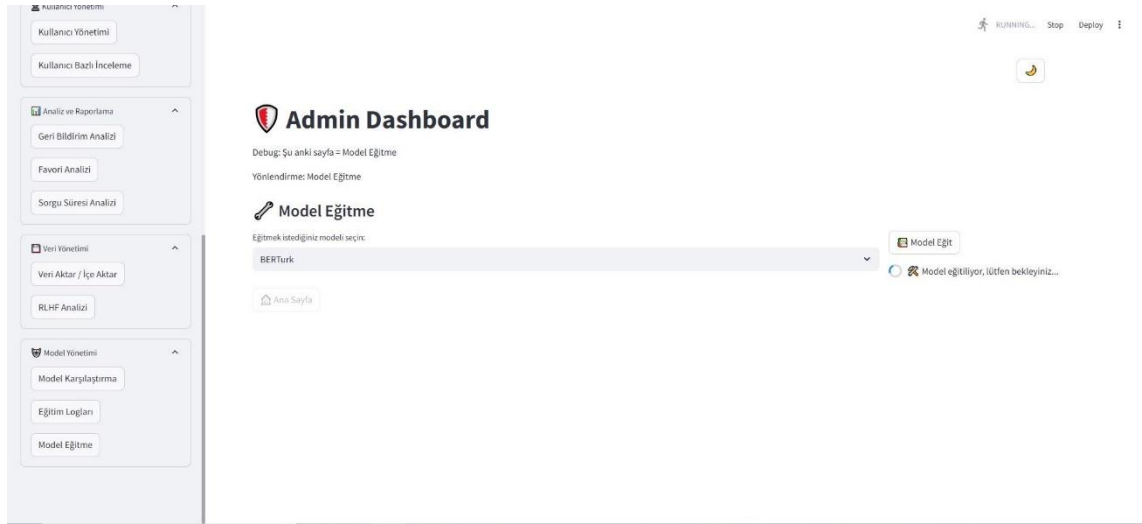
Şekil 4.106. SentimentBERT eğitim logları



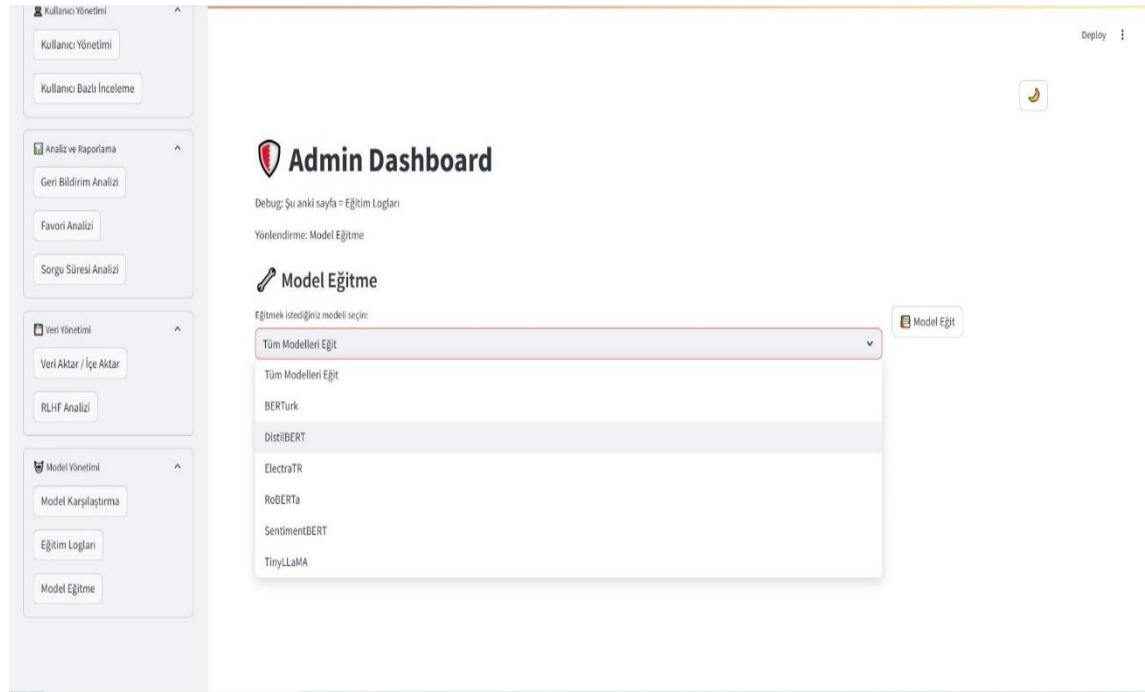
Şekil 4.107. SentimentBERT eğitim metrikleri zaman bazlı



Şekil 4.108. SentimentBERT tam log içeriği



Şekil 4.109. Model eğitme-1



Şekil 4.110. Model eğitme-2

[login](#)
[logout](#)
[model sec](#)
[register](#)
[user dashboard](#)
[veritabanları](#)
[View less](#)

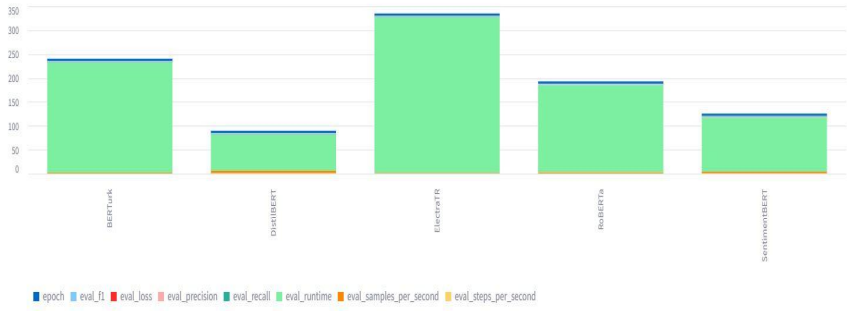
Admin Menü

Kullanıcı Yönetimi
[Kullanıcı Yönetimi](#)
[Kullanıcı Bazlı İnceleme](#)

Analiz ve Raporlama
[Geri Bildirim Analizi](#)
[Favori Analizi](#)
[Sorgu Süresi Analizi](#)

Tüm Modellerin Metrikleri

	epoch	eval_f1	eval_loss	eval_precision	eval_recall	eval_runtime	eval_samples_per_second	eval_steps_per_second
BERTurk	5.000000	1.000000	0.000006	1.000000	1.000000	230.271200	1.741000	0.221000
DistiBERT	5.000000	1.000000	0.000009	1.000000	1.000000	75.552500	5.308000	0.675000
ElectraTR	5.000000	1.000000	0.000027	1.000000	1.000000	325.418400	1.232000	0.157000
RoBERTa	5.000000	1.000000	0.000010	1.000000	1.000000	182.326900	2.199000	0.280000
SentimentBERT	5.000000	1.000000	0.000014	1.000000	1.000000	113.403100	3.536000	0.450000



Şekil 4.111. Tüm model matrikleri-1

[login](#)
[logout](#)
[model sec](#)
[register](#)
[user dashboard](#)
[veritabanları](#)
[View less](#)

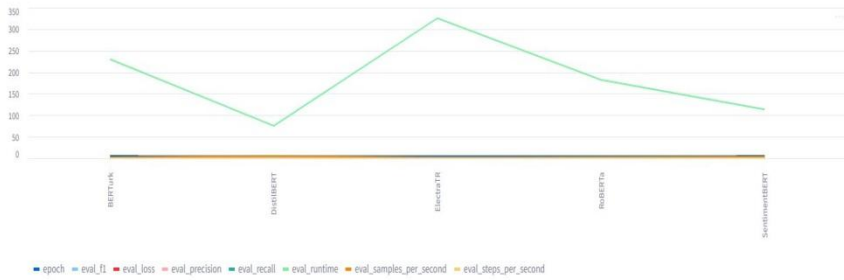
Admin Menü

Kullanıcı Yönetimi
[Kullanıcı Yönetimi](#)
[Kullanıcı Bazlı İnceleme](#)

Analiz ve Raporlama
[Geri Bildirim Analizi](#)
[Favori Analizi](#)
[Sorgu Süresi Analizi](#)

Tüm Modellerin Metrikleri

	epoch	eval_f1	eval_loss	eval_precision	eval_recall	eval_runtime	eval_samples_per_second	eval_steps_per_second
BERTurk	5.000000	1.000000	0.000006	1.000000	1.000000	230.271200	1.741000	0.221000
DistiBERT	5.000000	1.000000	0.000009	1.000000	1.000000	75.552500	5.308000	0.675000
ElectraTR	5.000000	1.000000	0.000027	1.000000	1.000000	325.418400	1.232000	0.157000
RoBERTa	5.000000	1.000000	0.000010	1.000000	1.000000	182.326900	2.199000	0.280000
SentimentBERT	5.000000	1.000000	0.000014	1.000000	1.000000	113.403100	3.536000	0.450000



Şekil 4.112. Tüm model matrikleri-2

TARTIŞMA

Bu çalışmada geliştirilen BATÜ InfoZekAI sistemi, Türkiye'deki üniversite kütüphanelerinde henüz yaygın olmayan zincirli ve geri bildirim odaklı akıllı tarama yaklaşımını benimseyerek mevcut sistemlerin ötesine geçmektedir. Geleneksel kütüphane sistemleri genellikle kullanıcıdan anahtar kelime temelli sorgular alarak yalnızca katalog sonuçlarını listelerken; InfoZekAI, sorunun yapısını semantik olarak analiz edip, sırasıyla YORDAM, E-Arşiv ve son çare olarak ChatGPT destekli dış kaynakları sorgulayan bir Retrieval-Augmented Generation (RAG) mimarisi kurmuştur (Kiela, 2023).

Bu yaklaşım, özellikle Scopus AI, Semantic Scholar AI, Dimensions Research Assistant, MIT BabelAI ve Ex Libris Summon Assistant gibi sistemlerle benzerlik göstermekte, ancak özgün olarak Türkçe dil işleme ve kullanıcıdan gelen RLHF geri bildirimlerine göre kendini geliştirme yeteneğiyle ayrılmaktadır. Örneğin, Scopus AI hâlâ sadece İngilizce dilinde çalışmakta ve sadece referans listesi sunabilmektedir (Elsevier, 2024); oysa InfoZekAI, sorunun türüne göre (tanımsal, güncel, tarihsel vb.) Türkçe modelleri devreye alarak cevabın yapısını da otomatik belirlemektedir. Bu, kullanıcı deneyimini ciddi biçimde artıran bir farklılıktır (Tenopir et al., 2023).

Sistemde kullanılan modellerin her biri, kullanıcıdan gelen yıldızlı değerlendirmeler doğrultusunda zincirleme fine-tuning ile eğitilmiş ve her yanıt; hallucination_grader.py, reward_scorer.py, tutarsızlık_tespit.py gibi modüllerle değerlendirilmiştir. Bu yapı, özellikle RLHF (Reinforcement Learning with Human Feedback) mimarisinin Türkçe kütüphane sistemlerine ilk kez bütünleşik olarak uygulanması açısından önemli bir yenilik taşımaktadır (Kaufmann et al., 2023).

Benzer yapılar dünya genelinde bazı öncü üniversitelerde deneme aşamasındadır. Örneğin:

- Harvard HOLLIS GPT Assistant, yalnızca açık erişimli Harvard koleksiyonlarını taramakta, ancak kullanıcı değerlendirmelerini model eğitimi için henüz kullanmamaktadır (Harvard Library, 2025).
- Stanford AlpacaSearch, akademik veri sorgularında çoklu model karşılaştırmaları sunsa da, sorgulara RLHF döngüsü entegre edilmemiştir (Stanford NLP, 2024).

- MIT BabelAI, kullanıcıdan gelen verileri anonimleştirerek her gece eğitime dahil etmekte, ancak yerel dil modeli (örneğin Türkçeye karşılık gelen) bulunmamaktadır (MIT CSAIL, 2025).
- Kore Ulusal Üniversitesi “KULibGPT”, Korece doğal dil sorgularını ChatGPT destekli bir dizin üzerinden karşılamaktadır; ancak sistemde sorgu tipi ayrımı ve tarihsel tutarsızlık tespiti yoktur (Jeong & Hwang, 2024).

BATÜ InfoZekAI, tüm bu örneklerle karşılaştırıldığında, çok sayıda kritik özelliği bir arada sunmaktadır:

- Zincirli sorgu hattı (YORDAM → E-Arşiv → Web → GPT)
- Türkçe destekli yerel modeller ve semantik analiz
- RLHF geri bildirim mekanizması
- Tarihsel tutarsızlık ve sorgu tipi analizleri
- Favori içerik yönetimi ve kullanıcı panosu
- Yönetici analitiği ve model kıyaslama panelleri

Sistemin `model_karsilastirma_kayit.py` ve `admin_dashboard.py` gibi bileşenleri sayesinde yöneticiler model başarımını, sorgu sürelerini ve kullanıcı tercihlerini analiz edebilmektedir. Bu tür bütünsel yönetim panelleri, günümüzde ileri düzey kütüphane yazılımlarının vazgeçilmez parçaları olarak gösterilmektedir (American Library Association, 2024).

Bununla birlikte, sistemin hâlâ geliştirilmeye açık yönleri de bulunmaktadır. Özellikle daha fazla açık erişim kaynak (ArXiv, Crossref, OpenAlex gibi) ile entegrasyonun gerçekleştirilmesi, sistemin literatür tarama gücünü daha da artıracaktır. Ayrıca, sesli arama ve cevaplama gibi erişilebilirlik odaklı özelliklerin eklenmesi; engelli kullanıcılar veya mobil kullanıcılar için büyük bir fayda sağlayacaktır (W3C Accessibility Guidelines, 2023).

5. SONUÇLAR VE ÖNERİLER

5.1 Sonuçlar

Bu tez kapsamında geliştirilen BATÜ InfoZekAI: BDM Tabanlı Akıllı Kütüphane Tarama ve Diyalog Sistemi, geleneksel bilgi erişim süreçlerindeki dağınıklık, kullanıcıya rehberlik eksikliği ve yapay zekâ destekli analiz yetersizliklerini ortadan kaldırmayı amaçlamıştır. Sistem, kullanıcıdan alınan doğal dildeki sorguları çok kaynaklı bir zincir yapısı (YORDAM → E-Arşiv → Web → ChatGPT) üzerinden işlemiş hem kurum içi hem de dış veri havuzlarından en doğru ve uygun bilgiyi sunmayı başarmıştır.

Türkçeye özel geliştirilen yerel modellerin kullanımı, RLHF tabanlı geri bildirim döngüsü, tarihsel tutarlılık analizi ve sorgu tipi tespiti gibi bileşenlerle desteklenen bu yapı, ülkemizde benzeri bulunmayan bir kütüphane otomasyon örneği ortaya koymuştur. Sistem, sadece bilgi sunmakla kalmayıp, aynı zamanda kendini geliştirme kapasitesine de sahip olarak kullanıcı etkileşiminden öğrenen dinamik bir platform hâline gelmiştir.

Akademik başarı, dijital dönüşüm ve kullanıcı memnuniyeti açısından değerlendirildiğinde; InfoZekAI, Batman Üniversitesi gibi orta ölçekli üniversitelerde yaygınlaştırılabilecek, açık kaynaklı ve modüler bir çözüm sunmaktadır.

5.2 Öneriler

- Veri Kaynaklarının Genişletilmesi: TRDizin, ArXiv, OpenAlex, Crossref ve YÖK Tez gibi akademik veri kaynaklarının sisteme entegre edilmesi, bilgi kapsamını ve doğruluk oranını artıracaktır.
- Çok Dilli Destek: İngilizce, Kürtçe, Arapça gibi ek diller için doğal dil işleme yeteneklerinin geliştirilmesi, bölgesel kullanıcıların erişimini artıracaktır.
- Sesli Arama ve Cevaplama: Görsel engelliler ve mobil kullanıcılar için sesli etkileşim modülleri geliştirilmeli, metinden konuşmaya (TTS) ve konuşmadan metne (ASR) teknolojileri sisteme entegre edilmelidir.
- Etik ve Veri Gizliliği: RLHF verilerinin anonimleştirilmesi, kullanıcı IP ve değerlendirme verilerinin sadece analiz amaçlı ve KVKK uyumlu biçimde kullanılması güvenli kullanım açısından kritik önemdedir.
- Yıllık Performans Raporları: Admin panel üzerinden kullanıcı etkinliği, sorgu doğruluk oranı, model başarımlarını karşılaştırmaları gibi metriklerin yıllık raporlanması yapılmalı ve stratejik kararlara temel oluşturmalıdır.
- Diğer Üniversitelerle İş Birliği: Geliştirilen sistem, açık kaynak olarak diğer üniversitelere sunulabilir. Ortak model havuzları, veri setleri ve RLHF ağı ile sürekli gelişen bölgesel bir akademik yapay zekâ ağı kurulabilir.

KAYNAKLAR

- Adadi, A., & Berrada, M. (2018). Peeking inside the black-box: A survey on explainable artificial intelligence (XAI). *IEEE Access*, 6, 52138–52160. <https://doi.org/10.1109/ACCESS.2018.2870052>
- Alonso, O., Gertz, M., & Baeza-Yates, R. (2007). Clustering and exploring search results using timeline constructions. In *Proceedings of the 30th annual international ACM SIGIR conference* (pp. 675–676). <https://doi.org/10.1145/1277741.1277880>
- American Library Association. (2024). *Library technology trends and best practices*. Chicago, IL: ALA Press.
- Associated Press. (2025). AI chatbots need more books to learn from – These libraries are opening their stacks.
- Banerjee, S., & Lavie, A. (2005). METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL workshop* (pp. 65–72).
- Bernhardsson, E. (2018). Annoy: Approximate Nearest Neighbors in C++/Python. <https://github.com/spotify/annoy>
- Borgman, C. L. (1996). Why are online catalogs still hard to use?. *Journal of the American Society for Information Science*, 47(7), 493–503. [https://doi.org/10.1002/\(SICI\)1097-4571\(199607\)47:7<493::AID-ASI3>3.0.CO;2-P](https://doi.org/10.1002/(SICI)1097-4571(199607)47:7<493::AID-ASI3>3.0.CO;2-P)
- Breeding, M. (2022). The Future of Library Automation. *Computers in Libraries*, 42(1), 10–14.
- Chase, H. (2023). LangChain: Building applications with LLMs through composability. <https://docs.langchain.com>
- Chen, J., Lee, K., Cho, K., & Fan, A. (2023). Check Your Facts and Try Again: Improving Large Language Models with External Knowledge and Temporal Filtering. *arXiv*. <https://arxiv.org/abs/2302.12095>
- Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., & Amodei, D. (2017). Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems (NeurIPS)*, 30. <https://arxiv.org/abs/1706.03741>
- Dietterich, T. G. (2000). Ensemble methods in machine learning. In *Multiple Classifier Systems* (pp. 1–15). Springer. https://doi.org/10.1007/3-540-45014-9_1
- Dong, L., Qian, Y., & Li, M. (2020). A Survey on Deep Learning for Named Entity Recognition and Intent Detection. *IEEE Access*, 8, 176385–176398. <https://doi.org/10.1109/ACCESS.2020.3026227>

- Doshi-Velez, F., & Kim, B. (2017). Towards a rigorous science of interpretable machine learning. arXiv. <https://arxiv.org/abs/1702.08608>
- Dwivedi, Y. K., Hughes, D. L., Ismagilova, E., Aarts, G., Coombs, C., Crick, T., ... & Williams, M. D. (2023). So what if ChatGPT wrote it? Multidisciplinary perspectives on opportunities, challenges and implications of generative conversational AI for research, practice and policy. *International Journal of Information Management*, 71, 102642. <https://doi.org/10.1016/j.ijinfomgt.2023.102642>
- Elsevier. (2024). Scopus AI whitepaper: Transparent, trusted and AI-augmented discovery. Retrieved from <https://www.elsevier.com>
- Furtado, T. B. F., & Esmin, A. (2023). Hybrid content dynamic recommendation system based in adapted tags and applied to digital library. ArXiv. <https://arxiv.org/abs/2309.08216>
- Gilpin, L. H., Bau, D., Yuan, B. Z., Bajwa, A., Specter, M., & Kagal, L. (2018). Explaining explanations: An overview of interpretability of machine learning. In 2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA) (pp. 80–89). <https://doi.org/10.1109/DSAA.2018.00018>
- Gupta, P., Gupta, M., Singh, V., & Kumaraguru, P. (2019). Intent detection and slot filling for goal-oriented dialogue systems using BERT. arXiv. <https://arxiv.org/abs/1902.10909>
- Guu, K., Lee, K., Tung, Z., Pasupat, P., & Chang, M. (2020). REALM: Retrieval-Augmented Language Model Pre-Training. arXiv. <https://arxiv.org/abs/2002.08909>
- Harvard Library. (2025). HOLLIS GPT Assistant project. Retrieved from <https://library.harvard.edu/ai>
- IFLA. (1998). Functional Requirements for Bibliographic Records: Final Report. International Federation of Library Associations and Institutions.
- Islam, M. (2023). Current trends of library automation: A case study on Rajshahi University Central Library. *Journal of Knowledge Learning and Science Technology*, 1(1), 34–45. <https://doi.org/10.60087/px37mc78>
- Izcard, G., & Grave, E. (2021). Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering. arXiv. <https://arxiv.org/abs/2007.01282>
- Jeong, H., & Hwang, J. (2024). Integrating Korean NLP into library chatbots: The case of KULibGPT. *Library Hi Tech*, 42(1), 12–27. <https://doi.org/10.1108/LHT-2023-0078>
- Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., ... & Zhang, Y. (2022). Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12), 1–38. <https://doi.org/10.1145/3571730>

- Johnson, J., Douze, M., & Jégou, H. (2019). Billion-scale similarity search with FAISS. *IEEE Transactions on Big Data*, 7(3), 535–547. <https://doi.org/10.1109/TBDATA.2019.2921572>
- Jones, D., & Liu, Y. (2018). *Asynchronous Programming in Python*. O'Reilly Media.
- Karasözen, B., & Zan, B. U. (2010). Türkiye'de açık erişim ve kurumsal arşiv çalışmaları. *Bilgi Dünyası*, 11(1), 1–19. <https://doi.org/10.15612/BD.2010.121>
- Karpinska, M., Pavlick, E., & Clark, P. (2021). EMBRACE: Empowering humans with better evaluation metrics for natural language generation. *arXiv*. <https://arxiv.org/abs/2107.07589>
- Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., ... & Yih, W. (2020). Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 6769–6781). <https://doi.org/10.18653/v1/2020.emnlp-main.550>
- Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., ... & Yih, W. T. (2020). Dense Passage Retrieval for Open-Domain Question Answering. *arXiv*. <https://arxiv.org/abs/2004.04906>
- Kaufmann, J., Vickers, T., & Chuang, Z. (2023). RLHF in education: Adaptive learning via feedback-driven fine-tuning. *Journal of Educational AI*, 1(2), 45–61.
- Kaufmann, T., Weng, P., Bengs, V., & Hüllermeier, E. (2023). A survey of reinforcement learning from human feedback. *arXiv:2312.14925*.
- Kiela, D. (2023). Retrieval-Augmented Generation: Towards grounded AI systems. *arXiv preprint arXiv:2301.11623*.
- Kuncheva, L. I. (2004). *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience.
- Lagoze, C., Van de Sompel, H., Nelson, M., & Warner, S. (2002). The Open Archives Initiative Protocol for Metadata Harvesting. <https://www.openarchives.org/OAI/openarchivesprotocol.html>
- Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Riedel, S. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *NeurIPS*. <https://arxiv.org/abs/2005.11401>
- Li, G., & Qiu, X. (2025). The evolution of library automation systems in Chinese academic libraries. *Information Technology and Libraries*, 44(1). <https://doi.org/10.5860/ital.v44i1.17108>
- Li, X., & Croft, W. B. (2003). Time-based language models. In *Proceedings of the twelfth international conference on Information and knowledge management* (pp. 469–475). <https://doi.org/10.1145/956863.956946>

- Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... & Stoyanov, V. (2022). RoBERTa: A robustly optimized BERT pretraining approach. arXiv preprint. <https://arxiv.org/abs/1907.11692>
- López-Nores, M., García-Duque, J., Blanco-Fernández, Y., & Pazos-Arias, J. J. (2021). The chatbot as a digital reference librarian: Evaluation and usability in academic libraries. *Library Hi Tech*, 39(1), 190–208. <https://doi.org/10.1108/LHT-03-2020-0075>
- Malkov, Y. A., & Yashunin, D. A. (2018). Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(4), 824–836. <https://doi.org/10.1109/TPAMI.2018.2889473>
- Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.
- Mena, A., Hoang, C. D. V., Shuster, K., Weston, J., & Dinan, E. (2022). Cross-examining Large Language Models with Automated Question Generation. arXiv. <https://arxiv.org/abs/2203.11147>
- MIT CSAIL. (2025). BabelAI: A multilingual chatbot for academic discovery. Retrieved from <https://csail.mit.edu/babelai>
- Nielsen, J. (1994). *Usability Engineering*. Morgan Kaufmann.
- Nimbhorkar, S. P. (2024). Library automation in universities and colleges libraries in India and abroad. *ShodhKosh*, 5(7), 325–333. <https://doi.org/10.29121/shodhkosh.v5.i7.2024.2674>
- NVIDIA. (2025). What is Retrieval-Augmented Generation (RAG)
- Oliver, C. (2010). *Introducing RDA: A Guide to the Basics*. American Library Association.
- OpenAI. (2023). GPT-4 Technical Report. <https://openai.com/research/gpt-4>
- Ponte, J. M., & Croft, W. B. (1998). A language modeling approach to information retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 275–281). <https://doi.org/10.1145/290941.291008>
- Ponte, J. M., & Croft, W. B. (1998). A language modeling approach to information retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference* (pp. 275–281). <https://doi.org/10.1145/290941.291008>
- Radlinski, F., & Joachims, T. (2007). Active exploration for learning rankings from clickthrough data. In *Proceedings of the 13th ACM SIGKDD* (pp. 570–579). <https://doi.org/10.1145/1281192.1281252>

- Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing. <https://arxiv.org/abs/1908.10084>
- Reynolds, G., MacDonald, K., & Nicol, K. (2020). AskON: A virtual reference service success story. *Partnership: The Canadian Journal of Library and Information Practice and Research*, 15(1). <https://doi.org/10.21083/partnership.v15i1.5591>
- Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). “Why should I trust you?”: Explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD (pp. 1135–1144). <https://doi.org/10.1145/2939672.2939778>
- Robertson, S., & Zaragoza, H. (2009). The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4), 333–389. <https://doi.org/10.1561/15000000019>
- Salton, G. (1971). *The SMART Retrieval System – Experiments in Automatic Document Processing*. Prentice-Hall.
- Salton, G. (1983). *Introduction to Modern Information Retrieval*. McGraw-Hill.
- Salton, G., Wong, A., & Yang, C. S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, 18(11), 613–620. <https://doi.org/10.1145/361219.361220>
- Schwartz, R., Dodge, J., Smith, N. A., & Etzioni, O. (2020). Green AI. *Communications of the ACM*, 63(12), 54–63. <https://doi.org/10.1145/3381831>
- Shen, Y., Chen, J., Yu, Z., & Li, J. (2023). Adaptive Online Learning for Dynamic User Feedback in Information Retrieval Systems. *Information Processing & Management*, 60(2), 103278. <https://doi.org/10.1016/j.ipm.2023.103278>
- Stanford NLP Group. (2024). AlpacaSearch: Lightweight AI search assistant. Retrieved from <https://stanfordnlp.github.io>
- Tay, A. (2024). AI & retrieval-augmented generation search: The content problem. *Musings about Librarianship*.
- Tenopir, C., Volentine, R., & King, D. (2023). The evolution of AI in academic libraries. *College & Research Libraries*, 84(5), 743–762.
- Tonta, Y. (2010). Dijital Kütüphaneler ve Geleceğin Kütüphaneleri. *Türk Kütüphaneciliği*, 24(4), 657–686.
- Turan, M. (2023). LegalTurk optimized BERT for multi-label text classification in Turkish legal domain. arXiv:2407.00648.
- W3C. (2023). Web Content Accessibility Guidelines (WCAG) 2.2. Retrieved from <https://www.w3.org/TR/WCAG22/>

- Wired. (2024). Reduce AI hallucinations with this neat software trick. WIRED Magazine.
- Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*, 5(2), 241–259. [https://doi.org/10.1016/S0893-6080\(05\)80023-1](https://doi.org/10.1016/S0893-6080(05)80023-1)
- Xiong, L., Xiong, C., Li, Y., Tang, K., Liu, J., Bennett, P. N., ... & Wang, M. (2021). Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. In ICLR. <https://arxiv.org/abs/2007.00808>
- Xu, J., & Croft, W. B. (2000). Improving the effectiveness of information retrieval with local context analysis. *ACM Transactions on Information Systems (TOIS)*, 18(1), 79–112. <https://doi.org/10.1145/333135.333138>
- Yılmaz, B., & Ünal, Y. (2021). Kurumsal Arşivlerin Sınıflandırılması: Türkiye’de Uygulama Örnekleri. *Bilgi Dünyası*, 22(1), 113–131. <https://doi.org/10.15612/BD.2021.846>
- Zhang, Y., Sun, X., & Lee, C. (2021). Caching and Reuse in AI-Powered Dialogue Systems: Approaches and Challenges. *ACM Computing Surveys*, 54(4), 1–30. <https://doi.org/10.1145/3439724>
- Zhang, Y., Sun, X., & Lee, C. (2022). Human-centric evaluation of AI systems in NLP. *Journal of Artificial Intelligence Research*, 74, 923–952. <https://doi.org/10.1613/jair.1.13566>

EKLER

EK-1 Temel Yapılandırma & Yardımcılar

No	Dosya	Görev
A1	settings.py	Model yolları, klasör sabitleri
A2	utils.py	Tema CSS, log okuma
A3	login\ manager.py	Parola hash / session temizliği

A1 settings.py kodları

```
# settings.py
MODEL_PATHS = {
    "BERTurk": "dbmdz/bert-base-turkish-cased",
    "DistilBERT": "distilbert-base-uncased",
    "ElectraTR": "dbmdz/electra-base-turkish-cased-discriminator",
    "RoBERTa": "roberta-base",
    "SentimentBERT": "nlptown/bert-base-multilingual-uncased-sentiment",
    "TinyLLaMA": "TinyLLaMA/TinyLLaMA-1.1B-Chat-v1.0", # Yeni model
}
LOG_DIR = "logs/training_logs"
DATA_DIR = "data"
OUTPUT_DIR = "fine_tuned_model"
```

A2 utils.py kodları

```
# utils.py
import chardet
import streamlit as st

def read_log_file(log_path):
    try:
        with open(log_path, "rb") as f:
            raw_data = f.read()
            result = chardet.detect(raw_data)
            detected_encoding = result["encoding"] or "utf-8"
            confidence = result["confidence"]
            st.info(f"📄 Tespit edilen kodlama: {detected_encoding} (Güven: %{confidence*100:.2f})")

        with open(log_path, "r", encoding=detected_encoding, errors="replace") as f:
            return f.readlines()
    except Exception as e:
        st.error(f"❌ Dosya okunamadı: {e}")
        return []
    import streamlit as st

def apply_dark_mode():
    import streamlit as st
    st.markdown("""
    <style>
```

```

body, .stApp {
  background-color: #181a1f !important;
  color: #e1e4e8 !important;
}
.stMarkdown, .stExpander {
  color: #e1e4e8 !important;
}
.stButton > button {
  background-color: #21262d !important;
  color: #f0f6fc !important;
  border: 1px solid #30363d !important;
}
.stTextInput > div > div > input,
.stTextArea > div > textarea {
  background-color: #22272e !important;
  color: #e1e4e8 !important;
  border: 1px solid #444c56 !important;
}
.stTextInput > div > div > input::placeholder,
.stTextArea > div > textarea::placeholder {
  color: #8b949e !important;
}
.stSelectbox > div > div > div > div {
  background-color: #22272e !important;
  color: #e1e4e8 !important;
  border: 1px solid #444c56 !important;
}
.stDataFrame, .css-1d391kg {
  background-color: #24292f !important;
  color: #e1e4e8 !important;
}
</style>
"""', unsafe_allow_html=True)

```

A3 login_manager.py kodları

```

import streamlit as st
import hashlib

def hash_password(password):
    return hashlib.sha256(password.encode()).hexdigest()

def clear_logged_user():
    st.session_state.clear()
def get_logged_user():
    return {
        "kullanici_adi": st.session_state.get("kullanici_adi", None),
        "rol": "admin" if st.session_state.get("admin", False) else "kullanici" }

```

EK-2 Veri tabanı & Veri Betikleri

No	Dosya	Görev
B1	veritabani_olustur.py	kutuphane.db ana şema
B2	veritabani_migrosyon.py	ALTER TABLE patch'leri
B3	sorgu_gecmisi_tablosu_olustur.py	Ek sorgu log tablosu

B1 veritabani_olustur.py kodları

```
import sqlite3

# Veritabanı bağlantısı
conn = sqlite3.connect("kutuphane.db")
cur = conn.cursor()

# Kullanıcılar tablosu
cur.execute("""
CREATE TABLE IF NOT EXISTS kullanicilar (
    kullanici_adi TEXT PRIMARY KEY,
    sifre TEXT NOT NULL,
    rol TEXT DEFAULT 'kullanici'
)
""")

# Geri bildirim tablosu
cur.execute("""
CREATE TABLE IF NOT EXISTS geri_bildirim (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    kullanici_adi TEXT,
    sorgu TEXT,
    yanit TEXT,
    kaynak TEXT,
    hallucination TEXT,
    puan INTEGER,
    tarih TEXT
)
""")

# Sorgu kayıt tablosu
cur.execute("""
CREATE TABLE IF NOT EXISTS sorgu_kayitlari (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    kullanici_adi TEXT,
    sorgu TEXT,
    yanit TEXT,
    tarih TEXT
)
""")
```

```

# Varsayılan admin ve kullanıcıları ekle (eğer yoksa)
kullaniciilar = [
    ("admin", "admin123", "admin"),
    ("kutuphane", "kutuphane123", "kullanici")
]

for kadi, sifre, rol in kullaniciilar:
    cur.execute("INSERT OR IGNORE INTO kullaniciilar (kullanici_adi, sifre,
rol) VALUES (?, ?, ?)", (kadi, sifre, rol))

conn.commit()
conn.close()
print("✓ Veritabanı başarıyla oluşturuldu ve varsayılan kullanıcılar
eklendi.")
# Veritabanları tablosu (EKUAL)
cur.execute("""
CREATE TABLE IF NOT EXISTS veritabanlari (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    ad TEXT NOT NULL,
    aciklama TEXT,
    link TEXT,
    kategori TEXT,
    acik_erisim BOOLEAN DEFAULT 0
)
""")

veritabani_verileri = [
    ("ScienceDirect", "2066 dergi", "https://www.sciencedirect.com/",
"Akademik", 0),
    ("JSTOR", "4496 dergi, 13050 rapor", "https://www.jstor.org/", "Arşiv
/ Akademik", 1),
    ("Mendeley", "Referans yönetimi", "https://www.mendeley.com/",
"Referans", 1),
    ("EBSCO - Academic Search Ultimate", "2024-2026",
"http://search.ebscohost.com/login.aspx?authtype=ip,uid", "EBSCOHOST", 0),
    ("EBSCO - OpenDissertations", "2024-2026",
"http://search.ebscohost.com/login.aspx?authtype=ip,uid", "EBSCOHOST", 1),
    ("EBSCO - GreenFILE", "2024-2026",
"http://search.ebscohost.com/login.aspx?authtype=ip,uid", "EBSCOHOST", 1),
    ("CAB Abstracts", "2023-2025", "http://www.cabdirect.org/", "Tarım /
Veteriner", 0),
    ("Emerald Premier eJournal", "2023-2025",
"http://www.emeraldinsight.com/", "Sosyal Bilimler", 0),
    ("IEEE", "2025-2027", "https://ieeexplore.ieee.org/Xplore/home.jsp",
"Mühendislik", 0),
    ("Scopus", "Elsevier atıf dizini", "https://www.scopus.com/", "Atıf /
İndeks", 0),
    ("iThenticate", "İntihal kontrol", "https://www.ithenticate.com/",
"İntihal", 0)

```

```

]

for veri in veritabani_verileri:
    cur.execute("INSERT OR IGNORE INTO veritabanlari (ad, aciklama, link,
kategori, acik_erasim) VALUES (?, ?, ?, ?, ?)", veri)

print("✓ veritabanlari tablosu ve örnek veriler eklendi.")
# Favoriler tablosu
cur.execute("""
CREATE TABLE IF NOT EXISTS favoriler (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    kullanıcı_adi TEXT,
    veritabani_id INTEGER,
    UNIQUE(kullanıcı_adi, veritabani_id)
)
""")

```

B2 veritabani_migrosyon.py kodları

```

import sqlite3

def veritabani_migrosyon():
    conn = sqlite3.connect("kutuphane.db")
    cur = conn.cursor()

    try:
        cur.execute("ALTER TABLE geri_bildirim ADD COLUMN kullanıcı_adi
TEXT")
        print("● 'kullanıcı_adi' sütunu eklendi.")
    except:
        print("● 'kullanıcı_adi' sütunu zaten mevcut.")

    try:
        cur.execute("ALTER TABLE geri_bildirim ADD COLUMN sorgu TEXT")
        print("● 'sorgu' sütunu eklendi.")
    except:
        print("● 'sorgu' sütunu zaten mevcut.")

    try:
        cur.execute("ALTER TABLE geri_bildirim ADD COLUMN hallucination
TEXT")
        print("● 'hallucination' sütunu eklendi.")
    except:
        print("● 'hallucination' sütunu zaten mevcut.")

    try:
        cur.execute("ALTER TABLE kullanıcılar ADD COLUMN rol TEXT DEFAULT
'kullanıcı'")
        print("● 'rol' sütunu eklendi.")
    except:
        print("● 'rol' sütunu zaten mevcut.")

```

```

except:
    print("⊙ 'rol' sütunu zaten mevcut.")

conn.commit()
conn.close()
print("✓ Veritabanı güncellemesi tamamlandı.")

if __name__ == "__main__":
    veritabani_migrosyon()

```

B3 sorgu_gecmisi_tablosu_olustur.py kodları

```

import sqlite3

def sorgu_gecmisi_tablosu_olustur():
    conn = sqlite3.connect("kutuphane.db")
    cur = conn.cursor()

    cur.execute("""
CREATE TABLE IF NOT EXISTS sorgu_gecmisi (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    kullanıcı_adi TEXT NOT NULL,
    tarih TEXT NOT NULL,
    sorgu TEXT NOT NULL,
    kaynak TEXT NOT NULL
)
""")

    conn.commit()
    conn.close()
    print("✓ sorgu_gecmisi tablosu başarıyla oluşturuldu.")

if __name__ == "__main__":
    sorgu_gecmisi_tablosu_olustur()

```

EK-3 Çekirdek Servis Katmanı

No	Dosya	Görev
C1	router.py	
C2	yordam_api.py	

C3	earsiv.py	
C4	search.py	
C5	ai.py	
C6	model_predictor.py	
C7	hallucination_grader.py	
C8	reward_scorer.py	
C9	tutarsizlik_tespit.py	
C10	sotgu_tipi_tespit.py	

C1 router.py kodları

```
# router.py
from yordam_api import yordam_sorgula
from earsiv import earsiv_sorgula
from search import web_arama_serper
from ai import akilli_cevapla
from model_predictor import predict_with_model
import socket
import streamlit as st

def internet_baglantisivar_mi():
    # Birden fazla sunucuyu kontrol et
    test_sunuculari = [
        ("www.google.com", 80),
        ("www.cloudflare.com", 80),
        ("www.wikipedia.org", 80)
    ]
    for host, port in test_sunuculari:
        try:
            socket.create_connection((host, port), timeout=2)
            print(f"İnternet bağlantısı doğrulandı: {host}")
            return True
        except OSError:
            continue
    print("İnternet bağlantısı doğrulanamadı.")
    return False

def yerel_modellerle_cevapla(soru: str) -> str:
    st.info("Yerel modellere geçiliyor...")
    print("Yerel modellere geçiliyor...")
    modeller = ["TinyLLaMA"]
    sonuclar = []

    for model_adi in modeller:
        try:
            print(f"{model_adi} modeli ile yerel yanıt üretiliyor...")
            yanit = predict_with_model(model_adi, soru)
            sonuclar.append(f"🤖 {model_adi} Yanıtı:\n{yanit}")
        except Exception as e:
            print(f"{model_adi} modelinde hata: {e}")
```

```

        sonuclar.append(f"🤖 {model_adi} Hatası:\nYanıt üretilemedi:
{e}")

    if not sonuclar:
        return "❌ Hiçbir modelden yanıt alınamadı. Lütfen sistem
yapılandırmasını kontrol edin."
        return "\n\n".join(sonuclar)

def chatbot_cevapla(soru: str) -> str:
    # 1. YORDAM
    print("YORDAM sorgusu denleniyor...")
    try:
        yordam_sonuc = yordam_sorgula(soru)
        print(f"YORDAM sonucu: {yordam_sonuc}")
        # YORDAM sonucu boş değilse ve hata mesajı içermiyorsa kabul et
        if yordam_sonuc and isinstance(yordam_sonuc, str) and
len(yordam_sonuc.strip()) > 0:
            # Daha esnek bir filtreleme
            if not any(x in yordam_sonuc.lower() for x in [
                "ilgili sonuç bulunamadı", "eşleşen sonuç bulunamadı",
                "beklenen veri alınamadı", "❌"
            ]):
                return f"📄 YORDAM'dan sonuç bulundu:\n\n{yordam_sonuc}"
            else:
                print("YORDAM sonucu hata mesajı içeriyor, bir sonraki
adıma geçiliyor...")
            else:
                print("YORDAM sonucu boş veya geçersiz, bir sonraki adıma
geçiliyor...")
        except Exception as e:
            print(f"YORDAM sorgu hatası: {e}")
            st.warning(f"⚠️ YORDAM sorgusu başarısız: {e}")

    # 2. E-Arşiv
    print("E-Arşiv sorgusu denleniyor...")
    try:
        earsiv_sonuc = earsiv_sorgula(soru)
        print(f"E-Arşiv sonucu: {earsiv_sonuc}")
        if earsiv_sonuc and isinstance(earsiv_sonuc, str) and
len(earsiv_sonuc.strip()) > 0:
            if not any(x in earsiv_sonuc.lower() for x in [
                "ilgili sonuç bulunamadı", "❌"
            ]):
                return f"📄 E-Arşiv'den sonuç bulundu:\n\n{earsiv_sonuc}"
            else:
                print("E-Arşiv sonucu hata mesajı içeriyor, bir sonraki
adıma geçiliyor...")
            else:

```

```

        print("E-Arşiv sonucu boş veya geçersiz, bir sonraki adıma
geçiliyor...")
    except Exception as e:
        print(f"E-Arşiv sorgu hatası: {e}")
        st.warning(f"⚠ E-Arşiv sorgusu başarısız: {e}")

# 3. Web Search (Serper.dev) - İnternet bağlantısı varsa
if internet_baglantis_i_var_mi():
    print("Web Search (Serper.dev) deneniyor...")
    try:
        web_sonuc = web_arama_serper(soru)
        print(f"Web Search sonucu: {web_sonuc}")
        if web_sonuc and isinstance(web_sonuc, str) and
len(web_sonuc.strip()) > 0 and "hata" not in web_sonuc.lower():
            if "ilgili sonuç bulunamadı" in web_sonuc.lower():
                print("Web'de bilgi bulunamadı, ChatGPT'ye
geçiliyor...")
            try:
                cevap = akıllı_cevapla(soru)
                return f"🌐 ChatGPT Yanıtı:\n\n{cevap}"
            except Exception as e:
                print(f"ChatGPT hatası (web sonrası): {e}")
                return yerel_modellerle_cevapla(soru)

        # RAG: Web Search sonucunu düzenle ve ChatGPT'ye prompt
olarak gönder
        bilgi_promptu = (
            f"Türkçe olarak, aşağıdaki web arama sonuçlarını
kullanarak soruya net ve doğru bir yanıt ver:\n\n"
            f"Web Arama Sonuçları:\n\n{web_sonuc}\n\n"
            f"Soru: {soru}\n\n"
            "Yanıtında yalnızca soruya odaklan, gereksiz
detaylardan kaçın ve doğal bir dil kullan."
        )
        try:
            print("Web Search başarılı, ChatGPT ile RAG yanıtı
üretiliyor...")
            cevap = akıllı_cevapla(bilgi_promptu)
            return f"🌐 Web aramasından elde edilen bilgiyle
yanıt:\n\n{cevap}"
        except Exception as e:
            print(f"ChatGPT hatası (web bilgisiyle): {e}")
            return yerel_modellerle_cevapla(soru)
    except Exception as e:
        print(f"Web arama hatası: {e}")

# 4. Doğrudan ChatGPT (Web Search başarısızsa, internet varsa)
if internet_baglantis_i_var_mi():
    try:

```

```

        print("Web arama başarısız, doğrudan ChatGPT yanıtı
        üretiliyor...")
        cevap = akilli_cevapla(soru)
        return f"🤖 ChatGPT Yanıtı:\n\n{cevap}"
    except Exception as e:
        print(f"ChatGPT hatası (son çare): {e}")

# 5. İnternet yoksa veya tüm adımlar başarısızsa yerel modele geç
st.warning("⚠ İnternet bağlantısı yok veya diğer kaynaklar başarısız.
Yerel modellere geçiliyor.")
print("İnternet bağlantısı yok veya diğer adımlar başarısız, yerel
modellere geçiliyor...")
return yerel_modellerle_cevapla(soru)

```

C2 yordam_api.py kodları

```

# yordam_api.py
import requests

def yordam_ara(kelime):
    url = "https://katalog.batman.edu.tr/yordam/webservis/webservis.php"
    token = ""
    params = {
        "token": token,
        "islem": "arama",
        "q": kelime,
        "start": "0",
        "sort": "KGTZ_dt desc",
        "rows": "100", # maksimum 100 sonuç getir
        "wt": "json"
    }

    try:
        response = requests.post(url, data=params, timeout=10)

        if not response.text.strip().startswith("{"):
            return "❌ YORDAM'dan beklenen veri alınamadı. Token geçersiz
            olabilir veya sunucu JSON dönmedi."

        json_data = response.json()
        docs = json_data.get("response", {}).get("docs", [])

        if not docs:
            return "❗YORDAM'da ilgili sonuç bulunamadı."

        sonuc = []
        for doc in docs:
            baslik = doc.get("kunyeEserAdiYazarlar_txt", "Başlık yok")
            yayin_yili = doc.get("qYayinTarihi_str", "-")

```

```

demirbas = doc.get("kunyeDemirbasKN_str", "-")
siniflama = doc.get("kunyeSiniflamaYer_str", "-")
kutuphane = doc.get("kunyeKutuphaneKN_str", "-")
durum = doc.get("kunyeDurumBilgisiWeb_str", "-").replace("_",
""").replace("|", "")

# arama kelimesiyle eşleşenleri filtrele
arama_matni = f"{baslik} {yayin_yili} {siniflama} {demirbas}
{kutuphane} {durum}"
if kelime.lower() not in arama_matni.lower():
    continue

yazar = baslik.split("/")[1].strip() if "/" in baslik else
"Yazar bilgisi yok"

sonuc.append(f"" {baslik.split('/')[0].strip()}
Yazar: {yazar}
Yıl: {yayin_yili}
Raf: {siniflama} - Demirbaş: {demirbas}
Kütüphane: {kutuphane} - Durum: {durum}
""")

if not sonuc:
    return "!YORDAM'da eşleşen sonuç bulunamadı."

return "\n\n---\n\n".join(sonuc)

except Exception as e:
    return f"✗ YORDAM işleme hatası: {e}"

# router.py ile uyumlu olması için alias
def yordam_sorgula(soru: str):
    return yordam_ara(soru)

```

C3 earsiv.py kodları

```

"""earsiv.py
Batman Üniversitesi E-Arşiv (OAI-PMH) arama modülü
ResumptionToken desteği ile tüm kayıtları sayfalar.
"""

import requests
import xml.etree.ElementTree as ET
from typing import List

BASE_URL = "https://earsiv.batman.edu.tr/server/oai/request"
NS = {
    "oai": "http://www.openarchives.org/OAI/2.0/",

```

```

        "dc": "http://purl.org/dc/elements/1.1/",
    }

def _parse_records(xml_root: ET.Element, sorgu: str) -> List[str]:
    """Başlıkta sorgu kelimesi geçen kayıtları biçimli döndürür."""
    matches = []
    for rec in xml_root.findall(".//oai:record", NS):
        title_el = rec.find(".//dc:title", NS)
        if title_el is None:
            continue
        title_txt = title_el.text or ""
        if sorgu.lower() not in title_txt.lower():
            continue

        creator = rec.find(".//dc:creator", NS)
        date = rec.find(".//dc:date", NS)
        ident = rec.find(".//dc:identifier", NS)

        matches.append(
            f"""■ **{title_txt.strip()}**
            {creator.text if creator is not None else 'Yazar yok'} | 📅 {date.text
            if date is not None else 'Yıl yok'}
            ☞ [Tam Metin]({ident.text if ident is not None else '#'})"""
        )
    return matches

def earsiv_ara(sorgu: str) -> str:
    """OAI-PMH ListRecords ile tarar, tüm sayfaları döner."""
    params = {"verb": "ListRecords", "metadataPrefix": "oai_dc"}
    tum_sonuc: List[str] = []

    try:
        while True:
            resp = requests.get(BASE_URL, params=params, timeout=10)
            resp.raise_for_status()
            root = ET.fromstring(resp.content)

            tum_sonuc.extend(_parse_records(root, sorgu))

            token_el = root.find(".//oai:resumptionToken", NS)
            if token_el is None or not (token_el.text or "").strip():
                break # Son sayfa

            params = {"verb": "ListRecords", "resumptionToken":
            token_el.text}

```

```
        return "\n\n".join(tum_sonuc) if tum_sonuc else "!E-Arşiv'de ilgili sonuç bulunamadı."
```

```
    except Exception as exc:  
        return f"❌ E-Arşiv taramada hata oluştu: {exc}"
```

```
# router.py ile uyumlu olması için alias  
def earsiv_sorgula(soru: str):  
    return earsiv_ara(soru)
```

```
# router ile uyumlu olması için alias  
def earsiv_sorgula(sorgu: str):  
    return earsiv_ara(sorgu)
```

C4 search.py kodları

```
# search.py  
import requests  
import os
```

```
# Serper.dev API Anahtarı ortamdan alınır  
SERPER_API_KEY = os.getenv("SERPER_API_KEY") or  
"d070aefbddd463f321a979a68636fe9dce3"
```

```
def web_arama_serper(soru: str) -> str:  
    url = "https://google.serper.dev/search"  
    headers = {  
        "X-API-KEY": SERPER_API_KEY,  
        "Content-Type": "application/json"  
    }  
    payload = {  
        "q": soru,  
        "gl": "tr",  
        "hl": "tr"  
    }  
  
    try:  
        response = requests.post(url, headers=headers, json=payload,  
timeout=10)  
        response.raise_for_status()  
        data = response.json()  
  
        # Arama sonucunu al ve birleştir  
        organic = data.get("organic", [])  
        if not organic:  
            return "!Web aramada ilgili sonuç bulunamadı."
```

```

        sonuc = ""
        for idx, item in enumerate(organic[:20], 1):
            sonuc += f"{idx}.
{item.get('title')}}\n{item.get('link')}}\n\n"

        return sonuc.strip()
    except Exception as e:
        return f"✗ Web arama hatası: {e}"

```

C5 ai.py kodları

```

import openai
import os

openai.api_key = os.getenv("OPENAI_API_KEY")

def akilli_cevapla(soru: str) -> str:
    try:
        response = openai.ChatCompletion.create(
            model="gpt-4-1106-preview",
            messages=[
                {"role": "system", "content": "Sen bir akademik bilgi
asistanısın. Kullanıcının sorduğu soruları açık, doğru ve güncel
bilgilerle yanıtla."},
                {"role": "user", "content": soru}
            ],
            temperature=0.7,
            max_tokens=1000
        )
        return response.choices[0].message.content.strip()
    except Exception as e:
        return f"Hata (GPT-4 Turbo): {e}"

```

C6 model_predictor.py kodları

```

# model_predictor.py
import torch
import os
from transformers import (
    AutoTokenizer,
    AutoModelForSequenceClassification,
    AutoModelForCausalLM,
)
from settings import MODEL_PATHS
from langdetect import detect, DetectorFactory

# Dil algılayıcıyı kararlı yapmak için
DetectorFactory.seed = 42
def detect_language(text):
    try:

```

```

        lang = detect(text)
        return lang
    except Exception as e:
        print(f"⚠ Dil algılama hatası: {e}")
        return "unknown"

def predict_with_model(model_name, prompt):
    print(f"{model_name} modeli yükleniyor...")
    if model_name not in MODEL_PATHS:
        raise ValueError(f"Desteklenmeyen model: {model_name}")

    # Sorgu dilini algıla ve Türkçe prompt oluştur
    source_lang = detect_language(prompt)
    if source_lang == "tr":
        enriched_prompt = f"{'Türkçe olarak cevap ver: '}{prompt}"
    else:
        enriched_prompt = prompt

    try:
        model_path = MODEL_PATHS[model_name]
        print(f"Model yolu: {model_path}")
        tokenizer = AutoTokenizer.from_pretrained(model_path)
    except Exception as e:
        raise ValueError(f"Tokenizer yüklenemedi ({model_name}): {e}")

    if tokenizer.pad_token is None:
        tokenizer.pad_token = tokenizer.eos_token
        print("pad_token ayarı yapıldı.")

    try:
        if "bert" in model_path.lower() or "electra" in model_path.lower()
or "trubert" in model_path.lower():
            model =
AutoModelForSequenceClassification.from_pretrained(model_path)
            print(f"SequenceClassification modeli yüklendi: {model_name}")
            inputs = tokenizer(prompt, return_tensors="pt",
truncation=True, padding=True)
            with torch.no_grad():
                outputs = model(**inputs)
            logits = outputs.logits
            prediction = torch.argmax(logits, dim=1)
            response = f"Model ({model_name}) tahmini:
{prediction.item()}"
        elif "tinyllama" in model_path.lower():
            model = AutoModelForCausalLM.from_pretrained(model_path)
            print(f"CausalLM modeli yüklendi: {model_name}")
            inputs = tokenizer(enriched_prompt, return_tensors="pt",
padding=True)

```

```

        output = model.generate(**inputs, max_new_tokens=150,
do_sample=True, temperature=0.5, top_k=50)
        response = tokenizer.decode(output[0],
skip_special_tokens=True)
    else:
        raise NotImplementedError(f"Bu model tipi desteklenmiyor:
{model_path}")
    except Exception as e:
        raise RuntimeError(f"Model ({model_name}) çalıştırılmadı: {e}")

    return response

```

C7 hallucination_grader.py kodları

```

def hesapla_hallucination_skoru(yanıt: str) -> str:
    """
    Basit bir 'hallucination' kontrolü: kaynak içeriği olup olmadığına
    göre skor verilir.
    """
    if any(etiket in yanıt.lower() for etiket in ["doi", "kaynakça",
"https://", "elif", "yazar", "makale"]):
        return "Yüksek Gerçeklik"
    elif "kaynak yok" in yanıt.lower() or "üretildi" in yanıt.lower():
        return "Düşük Gerçeklik"
    return "Belirsiz"

```

C8 reward_scorer.py kodları

```

from transformers import AutoTokenizer,
AutoModelForSequenceClassification
import torch

# Model ve tokenizer dizini
model_path = "fine_tuned_model/reward_model"
tokenizer = AutoTokenizer.from_pretrained(model_path)
model = AutoModelForSequenceClassification.from_pretrained(model_path)
model.eval()

# Cihaz seçimi (GPU varsa onu kullan)
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model.to(device)

def reward_puani_hesapla(soru, yanıt):
    metin = soru.strip() + " " + yanıt.strip()
    inputs = tokenizer(metin, return_tensors="pt", truncation=True,
padding="max_length", max_length=256)
    inputs = {k: v.to(device) for k, v in inputs.items()}
    with torch.no_grad():
        outputs = model(**inputs)
        skorlar = torch.softmax(outputs.logits, dim=-1).cpu().numpy()[0]

```

```

etiketler = ["kotu", "orta", "iyi"]
en_yuksek_index = int(skorlar.argmax())

return {
    "kotu": float(skorlar[0]),
    "orta": float(skorlar[1]),
    "iyi": float(skorlar[2]),
    "en_yuksek_label": etiketler[en_yuksek_index],
    "model": model_path.split("/")[-1] # örn: 'reward_model'
}

# Test için
if __name__ == "__main__":
    print(reward_puani_hesapla("Yapay zeka nedir?", "Yapay zeka
    makinelerin insan gibi düşünmesidir."))

```

C9 tutarsizlik_tespit.py kodları

```

import re
from datetime import datetime

def tespit_tarihsel_tutarsizlik(soru: str, yanit: str) -> str:
    """
    Soru ve yanıt metinleri içindeki tarihsel ifadeleri karşılaştırarak
    bariz bir zaman uyumsuzluğu olup olmadığını basit kurallarla kontrol
    eder.
    """

    # 1. Yıl tespiti (örnek: 2023, 1999 vb.)
    tarih_soru = re.findall(r"\b(19\d{2}|20\d{2})\b", soru)
    tarih_yanit = re.findall(r"\b(19\d{2}|20\d{2})\b", yanit)

    # 2. Bugünkü yıl referansı
    bugun_yil = datetime.now().year

    # 3. Eğer kullanıcı belirli bir yıl sorduysa, yanıt güncel yıl
    içeriyorsa tutarsızlık olabilir
    if tarih_soru and tarih_yanit:
        for y in tarih_yanit:
            if int(y) > bugun_yil:
                return "Yüksek Tutarsızlık (Gelecekteki tarih
                belirtilmiş)"

        for s in tarih_soru:
            for y in tarih_yanit:
                fark = abs(int(y) - int(s))
                if fark > 15:
                    return f"Olası Tutarsızlık ({s} yılı sorulmuş ama {y}
                    yılı bilgisi verilmiş)"

```

```

elif not tarih_soru and tarih_yanit:
    if any(int(y) > bugun_yil for y in tarih_yanit):
        return "Yüksek Tutarsızlık (Gelecekteki tarih belirtilmiş)"

    return "Tutarsızlık Tespit Edilmedi"

# Test
if __name__ == "__main__":
    sonuc = tespit_tarihsel_tutarsizlik("2010 yılında başbakan kimdi?",
    "Recep Tayyip Erdoğan 2023 yılında cumhurbaşkanıdır.")
    print("🚀 Tutarsızlık Sonucu:", sonuc)

```

C10 sotgu_tipi_tespit.py kodları

sorgu_tipi_tespit.py

```

def tespit_et(soru: str) -> str:
    soru_lower = soru.lower()

    if any(kelime in soru_lower for kelime in ["nedir", "kimdir", "ne işe
    yarar", "tanımı", "açıklaması"]):
        return "Tanımsal"
    elif any(kelime in soru_lower for kelime in ["ne zaman", "hangi yıl",
    "kaç yılında", "tarihi", "geçmişte", "önceki"]):
        return "Tarihsel"
    elif any(kelime in soru_lower for kelime in ["şu an", "günümüzde",
    "bugün", "şu anda", "son durum", "mevcut"]):
        return "Güncel"
    return "Belirsiz"

```

EK-4 Kullanıcı Arayüzü (Streamlit)

No	Dosya	Görev
D1	streamlit_app.py	Ana landing page
D2	streamlit_feedback.py	Puan/yorum kayıt
D3	streamlit_feedback_interface.py	Inline feedback

D4	pages/login.py	
D5	pages/logout.py	
D6	pages/register.py	
D7	pages/model_sec.py	
D8	pages/user_dashboard.py	
D9	pages/favorilerim.py	
D10	pages/veritabanlari.py	
D11	pages/hizmetlerimiz.py	
D12	pages/hakkimizda.py	
D13	pages/duyurular.py	
D14	pages/iletisim.py	

D1 streamlit_app.py kodları

```

import streamlit as st
import sqlite3
import re
import os
import sys
import torch
import time
from hallucination_grader import hesapla_hallucination_skoru
from reward_scorer import reward_puani_hesapla
from streamlit_feedback import log_sorgu
from streamlit_feedback_interface import geri_bildirim_modulu
from sorgu_tipi_tespit import tespit_et
from tutarsizlik_tespit import tespit_tarihsel_tutarsizlik
from datetime import datetime
import streamlit as st
from utils import apply_dark_mode
st.set_page_config(page_title="BATÜ InfoZekAI", page_icon="🧠",
layout="wide")
st.markdown("<h1 style='text-align: center;*>BATÜ InfoZekAI</h1>",
unsafe_allow_html=True)

if "dark_mode" not in st.session_state:
    st.session_state.dark_mode = False

col1, col2, col3 = st.columns([8, 1, 1])
with col3:
    if st.button("🌙" if not st.session_state.dark_mode else "☀️",
help="Gece Modu Değiştir"):
        st.session_state.dark_mode = not st.session_state.dark_mode

if st.session_state.dark_mode:
    apply_dark_mode()
# Python path'e çalışma dizinini ekle
sys.path.append(os.path.dirname(os.path.abspath(__file__)))

# router.py içe aktarmasını try-except ile sar

```

```

try:
    from router import chatbot_cevapla
except ImportError as e:
    st.error(f"❌ router.py yüklenemedi: {e}")
    print(f"[ERROR] router.py yüklenemedi: {e}")
    chatbot_cevapla = None

# Özel skor metinleri için CSS
st.markdown("""
<style>
.skor-degeri {
    font-size: 18px !important;
    font-weight: 600;
    color: #333;
    margin-top: -5px;
    margin-bottom: -5px;
    text-align: center;
}
</style>
""", unsafe_allow_html=True)

kullanici_adi = st.session_state.get("kullanici_adi", "Anonim")
authenticated = st.session_state.get("authenticated", False)

if kullanici_adi and kullanici_adi != "Anonim":
    st.markdown(
        f"<div style='position: absolute; top: 15px; right: 25px; font-size: 18px;'>👤 {kullanici_adi}</div>",
        unsafe_allow_html=True,
    )

st.markdown("---")

arama = st.text_input("🔍 Aramak istediğiniz konuyu yazınız", "")
ara_butonu = st.button("Ara")

kaynak = "Bilinmiyor"

yanit = None

if arama and (ara_butonu or st.session_state.get("arama_yapildi")):
    if chatbot_cevapla is None:
        st.error("❌ Sistem hatası: router.py yüklenemedi. Lütfen yöneticiyle iletişime geçin.")
    else:
        with st.spinner("Yanıt aranıyor..."):
            # Sorgu süresini ölçmeye başla
            baslangic_zamani = time.time()
            try:

```

```

        yanit = chatbot_cevapla(arama)
        print(f"[DEBUG] Yanıt alındı: {yanit[:100]}...")
    except Exception as e:
        print(f"[ERROR] chatbot_cevapla hatası: {e}")
        st.error(f"✗ Sorgu işlenirken hata oluştu: {e}")
        yanit = None
    bitis_zamani = time.time()
    sorgu_suresi = bitis_zamani - baslangic_zamani # Sorgu
suresi (saniye cinsinden)

    if yanit:
        st.markdown("### 🗨️ Yanıt:")
        # Sorgu süresini arayüzde göster
        st.markdown(f"🕒 **Sorgu Süresi:** {sorgu_suresi:.2f} saniye")

        bilgi_satiri = ""
        kaynak = "Ana Sayfa"

        if "📖 YORDAM'dan sonuç bulundu" in yanit:
            bilgi_satiri = "📖 YORDAM'dan sonuç bulundu"
            kaynak = "YORDAM"
            _, _, yanit = yanit.partition("\n\n")
        elif "📖 E-Arşiv'den sonuç bulundu" in yanit:
            bilgi_satiri = "📖 E-Arşiv'den sonuç bulundu"
            kaynak = "E-Arşiv"
            _, _, yanit = yanit.partition("\n\n")
        elif "[chatgpt]" in yanit.lower():
            kaynak = "ChatGPT"
        elif "🌐" in yanit:
            kaynak = "Web"
        else:
            print("[DEBUG] YORDAM veya diğer kaynaklardan sonuç
bulunamadı.")

        if bilgi_satiri:
            st.info(re.sub(r'[\ud800-\udfff]', '',
bilgi_satiri.strip()))

        sorgu_tipi = tespit_et(arama)
        st.markdown(f"#### 📌 Sorgu Tipi: {sorgu_tipi}")

        yanitlar = [
            y.strip() for y in yanit.split("\n\n")
            if y.strip() and not y.strip().startswith("---")
        ]

        for i, parca in enumerate(yanitlar):
            st.markdown(f"#### 📄 Cevap {i+1}")
            temiz_parca = re.sub(r'[\ud800-\udfff]', '', parca)

```

```

st.write(temiz_parca)

hallucination = hesapla_hallucination_skoru(temiz_parca)
reward_skoru = reward_puani_hesapla(arama, temiz_parca)
tutarsizlik = tespit_tarihsel_tutarsizlik(arama,
temiz_parca)

col1, col2, col3 = st.columns(3)
with col1:
    st.markdown("<span class='metric-title'>🧠  
Hallucination Skoru</span>", unsafe_allow_html=True)
    st.markdown(f"<span class='skor-  
degeri'>{hallucination}</span>", unsafe_allow_html=True)
with col2:
    st.markdown("<span class='metric-title'>🏆 Reward  
Etiketi</span>", unsafe_allow_html=True)
    st.markdown(f"<span class='skor-  
degeri'>{reward_skoru.get('en_yuksek_label',  
'Bilinmiyor').capitalize()}</span>", unsafe_allow_html=True)
with col3:
    st.markdown("<span class='metric-title'>📅 Tarihsel  
Tutarlılık</span>", unsafe_allow_html=True)
    st.markdown(f"<span class='skor-  
degeri'>{tutarsizlik}</span>", unsafe_allow_html=True)

if i == 0:
    try:
        conn = sqlite3.connect("kutuphane.db")
        cur = conn.cursor()
        cur.execute("""
            INSERT INTO reward_etiket (kullanici_adi,
sorgu, model, reward_label, yanit, tarih, sorgu_tipi,
tarihsel_tutarsizlik, sorgu_suresi)
            VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)
        """, (
            kullanici_adi,
            arama,
            reward_skoru.get("model", "Bilinmiyor"),
            reward_skoru.get("en_yuksek_label",
"Bilinmiyor"),
            temiz_parca,
            datetime.now().strftime("%Y-%m-%d %H:%M:%S"),
            sorgu_tipi,
            tutarsizlik,
            sorgu_suresi
        ))
        conn.commit()
        conn.close()
    except Exception as e:

```

```

        print(f"⚠️ Reward verisi veritabanına
kaydedilemedi: {e}")

        log_sorgu(sorgu=arama, kaynak=kaynak, yanıt=parca,
hallucination=hallucination)
        geri_bildirim_modulu(i, parca, kullanıcı_adi)
    else:
        st.warning("⚠️ Yanıt alınamadı. Lütfen tekrar deneyin.")

if arama:
    st.session_state["arama_yapildi"] = True
else:
    st.session_state["arama_yapildi"] = False

st.markdown("---")

def oturum_acildi_mi():
    return st.session_state.get("authenticated", False)

if oturum_acildi_mi():
    try:
        conn = sqlite3.connect("kutuphane.db")
        cur = conn.cursor()
        cur.execute("""
            INSERT INTO sorgu_gecmisi (kullanıcı_adi, sorgu, kaynak,
tarih, sorgu_suresi)
            VALUES (?, ?, ?, ?, ?)
            """, (kullanıcı_adi, arama, kaynak, datetime.now().strftime("%Y-
%m-%d %H:%M:%S"), sorgu_suresi if 'sorgu_suresi' in locals() else 0.0))
        conn.commit()
        conn.close()
    except Exception as e:
        print(f"❌ sorgu_gecmisi kaydedilemedi: {e}")

col1, col2, col3, col4, col5, col6 = st.columns(6)
with col1:
    if not authenticated:
        if st.button("🔒 Oturum Aç"):
            st.switch_page("pages/login.py")
    else:
        if st.button("🔒 Oturumu Kapat"):
            for key in list(st.session_state.keys()):
                del st.session_state[key]
            st.rerun()

with col2:
    if st.button("🔊 Duyurular"):
        st.switch_page("pages/duyurular.py")
with col3:

```

```

        if st.button("🔗 Hizmetlerimiz"):
            st.switch_page("pages/hizmetlerimiz.py")
    with col4:
        if st.button("🗄️ Veritabanları"):
            st.switch_page("pages/veritabanlari.py")
    with col5:
        if st.button("📖 Hakkımızda"):
            st.switch_page("pages/hakkimizda.py")
    with col6:
        if st.button("✉️ İletişim"):
            st.switch_page("pages/iletisim.py")

st.markdown("---")
st.markdown("<br><br><br>", unsafe_allow_html=True)

st.markdown("""
<p style='text-align: center;'>
© 2025 <a href='https://batman.edu.tr/Birimler/kutuphane' target='_blank'
style='text-decoration: none; color: inherit;'>
Batman Üniversitesi Kütüphane ve Dokümantasyon Daire Başkanlığı
</a>
</p>
""", unsafe_allow_html=True)

```

D2 streamlit_feedback.py kodları

```

import streamlit as st
import sqlite3

def feedback_interface():
    st.markdown("---")
    st.subheader("Geri Bildirim Verin")

    puan = st.slider("Yanıtı ne kadar faydalı buldunuz?", 1, 5, 3)
    yorum = st.text_area("Yorum (isteğe bağlı):")

    if st.button("Gönder"):
        kullanıcı_adi = st.session_state.get("kullanici_adi", "Anonim")

        conn = sqlite3.connect("kutuphane.db")
        cursor = conn.cursor()
        cursor.execute("""
            UPDATE geri_bildirim
            SET puan = ?, yorum = ?
            WHERE kullanıcı_adi = ? AND puan IS NULL
            ORDER BY tarih DESC LIMIT 1
            """, (puan, yorum, kullanıcı_adi))
        conn.commit()
        conn.close()

```

```

        st.success("Geri bildiriminiz kaydedildi. Teşekkürler!")

def log_sorgu(sorgu, kaynak, yanit, hallucination):
    kullanıcı_adi = st.session_state.get("kullanici_adi", "Anonim")

    conn = sqlite3.connect("kutuphane.db")
    cursor = conn.cursor()
    cursor.execute("""
        INSERT INTO geri_bildirim (kullanici_adi, sorgu, yanit, kaynak,
hallucination, tarih)
        VALUES (?, ?, ?, ?, ?, datetime('now', 'localtime'))
    """, (kullanici_adi, sorgu, yanit, kaynak, hallucination))
    conn.commit()
    conn.close()

```

D3 streamlit_feedback_interface.py kodları

```

import streamlit as st
import sqlite3

def geri_bildirim_modulu(i, parca, kullanıcı_adi):
    st.markdown("---")

    # 📦 Optimize edilmiş hizalama (tek satıra sığacak şekilde)
    col1, col2, col3, col4 = st.columns([2.5, 2, 3.5, 2])

    # 🛡️ Güven Puanı
    with col1:
        guven = st.radio("🛡️ Güven:", [1, 2, 3, 4, 5], horizontal=True,
key=f"guven_{i}")

    # ★ Favoriye Ekle
    with col2:
        if st.button("★ Favoriye Ekle", key=f"favori_{i}"):
            conn = sqlite3.connect("kutuphane.db")
            cur = conn.cursor()
            cur.execute("""
                INSERT INTO favori_detay (kullanici_adi, yanit_metni,
favori_ismi, favori_etiketi)
                VALUES (?, ?, ?, ?)
            """, (kullanici_adi, parca, "İsimsiz Favori", ""))
            conn.commit()
            conn.close()
            st.success("★ Favorilere eklendi!")

    # 💬 Yorum alanı
    with col3:

```

```

        yorum = st.text_input("💬 Yorum (isteğe bağlı):",
key=f"yorum_{i}", placeholder="Kısa yorum girin...")

# 📧 Geri Bildirim Gönder
with col4:
    if st.button("📧 Geri Bildirimi Gönder", key=f"gonder_{i}"):
        conn = sqlite3.connect("kutuphane.db")
        cur = conn.cursor()
        cur.execute("""
            UPDATE geri_bildirim
            SET puan = ?, yorum = ?
            WHERE id = (
                SELECT id FROM geri_bildirim
                WHERE kullanıcı_adi = ? AND yanıt = ?
                ORDER BY tarih DESC LIMIT 1
            )
        """, (guven, yorum, kullanıcı_adi, parca))
        conn.commit()
        conn.close()
        st.success("📧 Geri bildiriminiz kaydedildi.")

```

D4 pages/login.py kodları

```

# pages/login.py

import streamlit as st
import sqlite3
import hashlib

# Kullanıcı doğrulama fonksiyonu
def kullanıcı_dogrula(kullanıcı_adi, sifre):
    sifre_hash = hashlib.sha256(sifre.encode()).hexdigest()
    conn = sqlite3.connect("kutuphane.db")
    cur = conn.cursor()
    cur.execute("SELECT sifre, rol FROM kullanicilar WHERE kullanıcı_adi
= ?", (kullanıcı_adi,))
    sonuc = cur.fetchone()
    conn.close()

    if sonuc:
        veritabanı_sifresi, rol = sonuc
        admin = rol == "admin"
        if sifre_hash == veritabanı_sifresi:
            return True, admin
    return False, False

# Streamlit sayfa ayarları
st.set_page_config(page_title="Giriş Yap", page_icon="🔒")
st.title("🔒 Kullanıcı Girişi")

```

```

kullanici_adi = st.text_input("Kullanıcı Adı")
sifre = st.text_input("Şifre", type="password")

if st.button("Giriş Yap"):
    if kullanici_adi and sifre:
        dogrulandi, admin = kullanici_dogrula(kullanici_adi, sifre)
        if dogrulandi:
            st.success(f"Hoş geldin, {kullanici_adi}!")
            st.session_state["authenticated"] = True
            st.session_state["kullanici_adi"] = kullanici_adi
            st.session_state["admin"] = admin

            # Yerleşik switch_page ile ana sayfaya yönlendir
            st.switch_page("streamlit_app.py")
        else:
            st.error("Geçersiz kullanıcı adı veya şifre.")
    else:
        st.warning("Lütfen kullanıcı adı ve şifrenizi girin.")

# Ana sayfaya dön butonu
if st.button("🏠 Ana Sayfa"):
    st.switch_page("streamlit_app.py")

st.markdown("---") # Ayırıcı çizgi
st.markdown("<br>", unsafe_allow_html=True)

st.markdown("""
<p style='text-align: center;'>
© 2025 <a href='https://batman.edu.tr/Birimler/kutuphane' target='_blank'
style='text-decoration: none; color: inherit;'>
Batman Üniversitesi Kütüphane ve Dokümantasyon Daire Başkanlığı
</a>
</p>
""", unsafe_allow_html=True)

```

D5 pages/logout.py kodları

```

import streamlit as st
from login_manager import clear_logged_user

# Oturumu temizle
clear_logged_user()

st.success("Çıkış yapılıyor...")

# Ana sayfaya yönlendir (sayfa yüklemesi tamamlandıktan sonra)
st.switch_page("streamlit_app.py")

# Ana sayfaya dön

```

```

if st.button("🏠 Ana Sayfa"):
    st.switch_page("streamlit_app.py")

st.markdown("---") # Ayırıcı çizgi
st.markdown("<br>", unsafe_allow_html=True)

st.markdown("""
<p style='text-align: center;'>
© 2025 <a href='https://batman.edu.tr/Birimler/kutuphane' target='_blank'
style='text-decoration: none; color: inherit;'>
Batman Üniversitesi Kütüphane ve Dokümantasyon Daire Başkanlığı
</a>
</p>
""", unsafe_allow_html=True)

```

D6 pages/register.py kodları

```

import streamlit as st
import sqlite3
from login_manager import hash_password

st.set_page_config(page_title="Hesap Oluştur", layout="wide")

st.markdown("##  Hesap Oluştur")

conn = sqlite3.connect('kutuphane.db')
c = conn.cursor()

with st.form("register_form"):
    yeni_kullanici = st.text_input("Yeni Kullanıcı Adı")
    yeni_sifre = st.text_input("Yeni Şifre", type="password")
    kayit = st.form_submit_button("Hesap Oluştur")

    if kayit:
        c.execute("SELECT * FROM kullanicilar WHERE kullanıcı_adi=?",
(yeni_kullanici,))
        if c.fetchone():
            st.warning("Bu kullanıcı adı zaten mevcut.")
        else:
            hashed_password = hash_password(yeni_sifre)
            c.execute("INSERT INTO kullanicilar (kullanıcı_adi, sifre)
VALUES (?, ?)", (yeni_kullanici, hashed_password))
            conn.commit()
            st.success("Hesap başarıyla oluşturuldu.")

conn.close()

if st.button("🏠 Ana Sayfa"):
    st.switch_page("streamlit_app.py")

```

```

st.markdown("---") # Ayırıcı çizgi
st.markdown("<br>", unsafe_allow_html=True)

st.markdown("""
<p style='text-align: center;'>
@ 2025 <a href='https://batman.edu.tr/Birimler/kutuphane' target='_blank'
style='text-decoration: none; color: inherit;'>
Batman Üniversitesi Kütüphane ve Dokümantasyon Daire Başkanlığı
</a>
</p>
""", unsafe_allow_html=True)

```

D7 pages/model_sec.py kodları

```

# pages/model_sec.py
import streamlit as st

st.set_page_config(page_title="Model Seçimi", page_icon="🤖")

st.title("🤖 Kullanılacak Modeli Seç")

model_secimi = st.radio(
    "Lütfen kullanmak istediğiniz modeli seçin:",
    (
        "BERTurk",
        "TinyLlama",
        "DistilBERT",
        "AlpacaTR",
        "TruBERT",
        "ElectraTR",
        "T5-Turkish"
    )
)

if st.button("Modeli Kaydet"):
    st.session_state["secili_model"] = model_secimi
    st.success(f"Seçilen model: {model_secimi}")

if st.button("🏠 Ana Sayfa"):
    st.switch_page("streamlit_app.py")

```

D8 pages/user_dashboard.py kodları

```

import streamlit as st
import sqlite3
import pandas as pd
import admin_sections.model_karsilastirma as model_karsilastirma

# Sayfa yapılandırması

```

```

st.set_page_config(page_title="Kullanıcı Paneli", page_icon="👤",
layout="wide")

# Oturum kontrolü
if "authenticated" not in st.session_state or not
st.session_state["authenticated"]:
    st.warning("🔒 Bu sayfayı görmek için giriş yapmalısınız.")
    st.stop()

kullanici_adi = st.session_state.get("kullanici_adi", None)

st.title("👤 Kullanıcı Paneli")

# Sidebar menüsünü gruplandırılmış hale getirme
st.sidebar.title("📁 Kullanıcı Menü")

# Secilen sayfayı session_state ile yönet
if "secilen_sayfa" not in st.session_state:
    st.session_state.secilen_sayfa = "🔍 Geçmiş Sorgularım"

# Kullanıcı Profili
with st.sidebar.expander("👤 Kullanıcı Profili", expanded=True):
    if st.button("✂ Şifre Güncelleme", key="sifre_guncelleme_btn"):
        st.session_state.secilen_sayfa = "✂ Şifre Güncelleme"

# Sorgu ve Geri Bildirim
with st.sidebar.expander("🔍 Sorgu ve Geri Bildirim"):
    if st.button("🔍 Geçmiş Sorgularım", key="gecmis_sorgular_btn"):
        st.session_state.secilen_sayfa = "🔍 Geçmiş Sorgularım"
    if st.button("🗨 Geri Bildirim ve Puanlamalarım",
key="geri_bildirim_btn"):
        st.session_state.secilen_sayfa = "🗨 Geri Bildirim ve
Puanlamalarım"

# Favoriler ve RLHF
with st.sidebar.expander("★ Favoriler ve RLHF"):
    if st.button("★ Favorilerim", key="favorilerim_btn"):
        st.session_state.secilen_sayfa = "★ Favorilerim"
    if st.button("📊 RLHF Skorlarım", key="rlhf_skorlarim_btn"):
        st.session_state.secilen_sayfa = "📊 RLHF Skorlarım"

# Model Analizi
with st.sidebar.expander("🔎 Model Analizi"):
    if st.button("🔎 Model Karşılaştırmalarım",
key="model_karsilastirma_btn"):
        st.session_state.secilen_sayfa = "🔎 Model Karşılaştırmalarım"

conn = sqlite3.connect("kutuphane.db")
cur = conn.cursor()

```

```

# 🔍 Geçmiş Sorgularım
if st.session_state.secilen_sayfa == "🔍 Geçmiş Sorgularım":
    st.subheader("🔍 Geçmiş Sorgularım")
    sorgular = pd.read_sql_query("""
        SELECT tarih, sorgu, kaynak
        FROM sorgu_gecmisi
        WHERE kullanıcı_adi = ?
        ORDER BY tarih DESC
        """, conn, params=(kullanıcı_adi,))
    if not sorgular.empty:
        st.dataframe(sorgular, use_container_width=True)
    else:
        st.info("Henüz sorgu yapılmamış.")

# 🗣️ Geri Bildirim ve Puanlamalarım
elif st.session_state.secilen_sayfa == "🗣️ Geri Bildirim ve Puanlamalarım":
    st.subheader("🗣️ Geri Bildirim ve Puanlamalarım")
    geri_bildirimler = pd.read_sql_query("""
        SELECT tarih, sorgu, kaynak, puan, hallucination
        FROM geri_bildirim
        WHERE kullanıcı_adi = ?
        ORDER BY tarih DESC
        """, conn, params=(kullanıcı_adi,))
    if not geri_bildirimler.empty:
        st.dataframe(geri_bildirimler, use_container_width=True)
    else:
        st.info("Henüz değerlendirme yapmadınız.")

# ★ Favorilerim
elif st.session_state.secilen_sayfa == "★ Favorilerim":
    st.subheader("★ Favori Verilerim ve Yanıtlarım")
    favoriler = pd.read_sql_query("""
        SELECT favori_ismi, favori_etiketi, yanıt_metni
        FROM favori_detay
        WHERE kullanıcı_adi = ?
        ORDER BY rowid DESC
        """, conn, params=(kullanıcı_adi,))
    if not favoriler.empty:
        st.dataframe(favoriler, use_container_width=True)
    else:
        st.info("Henüz favori yanıt eklenmemiş.")

# 🏠 RLHF Skorlarım
elif st.session_state.secilen_sayfa == "🏠 RLHF Skorlarım":
    st.subheader("🏠 Reward Model Skorlarım")
    reward_logs = pd.read_sql_query("""

```

```

        SELECT tarih, sorgu, model, reward_label, sorgu_tipi,
        tarihsel_tutarsizlik
        FROM reward_etiket
        WHERE kullanıcı_adi = ?
        ORDER BY tarih DESC
        """ , conn, params=(kullanıcı_adi,))
    if not reward_logs.empty:
        st.dataframe(reward_logs, use_container_width=True)
    else:
        st.info("Henüz reward etiketi oluşmamış.")

# 🛠 Model Karşılaştırmalarım
elif st.session_state.secilen_sayfa == "🛠 Model Karşılaştırmalarım":
    st.subheader("🛠 Model Karşılaştırma Geçmişim")
    model_karsilastirma.render(admin_mode=False)

# ✂ Şifre Güncelleme
elif st.session_state.secilen_sayfa == "✂ Şifre Güncelleme":
    st.subheader("✂ Şifre Güncelleme")
    with st.form("sifre_guncelleme_formu"):
        yeni_sifre = st.text_input("Yeni Şifre", type="password")
        onay = st.form_submit_button("Şifreyi Güncelle")
        if onay and yeni_sifre:
            cur.execute("UPDATE kullanicilar SET sifre = ? WHERE
            kullanıcı_adi = ?", (yeni_sifre, kullanıcı_adi))
            conn.commit()
            st.success("✔ Şifreniz başarıyla güncellendi.")

    conn.close()

# Ana sayfa butonu
if st.button("🏠 Ana Sayfa"):
    st.switch_page("streamlit_app.py")

```

D9 pages/favorilerim.py kodları

```

import streamlit as st
import sqlite3
import pandas as pd
from utils import apply_dark_mode
st.set_page_config(page_title="Favorilerim", page_icon="★",
layout="wide")
if "dark_mode" not in st.session_state:
    st.session_state.dark_mode = False

col1, col2, col3 = st.columns([8, 1, 1])
with col3:
    if st.button("🌙" if not st.session_state.dark_mode else "☀️",
help="Gece Modu Değiştir"):

```

```

        st.session_state.dark_mode = not st.session_state.dark_mode

if st.session_state.dark_mode:
    apply_dark_mode()

st.title("★ Favorilerim")

# Kullanıcı oturum bilgisi
kullanici_adi = st.session_state.get("kullanici_adi", None)
authenticated = st.session_state.get("authenticated", False)

# Giriş yapmayan kullanıcılar favorilere erişemez
if not authenticated:
    st.warning("🔒 Favorilerinizi görmek için lütfen giriş yapınız.")
    st.stop()

# Veritabanı bağlantısı
conn = sqlite3.connect("kutuphane.db")

st.subheader("📖 Veritabanı Favorilerim")
veritabani_favoriler = pd.read_sql_query("""
    SELECT v.ad AS Veritabanı_Adı, v.aciklama AS Açıklama
    FROM favoriler f
    JOIN veritabanlari v ON f.veritabani_id = v.id
    WHERE f.kullanici_adi = ?
    ORDER BY v.kategori, v.ad
""", conn, params=(kullanici_adi,))

if not veritabani_favoriler.empty:
    st.dataframe(veritabani_favoriler, use_container_width=True)
else:
    st.info("Henüz veritabanı favoriniz yok.")

st.markdown("---")

st.subheader("🗨️ Yanıt Favorilerim")
yanit_favoriler = pd.read_sql_query("""
    SELECT favori_ismi AS Favori_Başlığı, favori_etiketi AS Etiket,
    yanıt_metni AS Yanıt
    FROM favori_detay
    WHERE kullanici_adi = ?
    ORDER BY rowid DESC
""", conn, params=(kullanici_adi,))

if not yanit_favoriler.empty:
    st.dataframe(yanit_favoriler, use_container_width=True)
else:
    st.info("Henüz yanıt favoriniz yok.")

```

```

conn.close()

# Ana sayfaya dön butonu
st.markdown("---")
if st.button("🏠 Ana Sayfaya Dön"):
    from streamlit_extras.switch_page_button import switch_page
    switch_page("streamlit_app")

```

D10 pages/veritabanlari.py kodları

```

import streamlit as st
import pandas as pd
import sqlite3

st.set_page_config(page_title="Veritabanları", page_icon="📖")
st.title("📖 Batman Üniversitesi Abone Olunan Veritabanları")

st.markdown("""
Kütüphanemiz aşağıdaki akademik veritabanlarına **kampüs içi veya [Kampüs Dışı Erişim (VETİS)](https://katalog.batman.edu.tr/vetisbt/)** ile erişim sağlamaktadır.
📄 [VETİS Kullanım Kılavuzu (PDF)](https://www.yordambt.com/vetis/VETIS\_Kullanım\_Kilavuzu.pdf)
""")

# Veritabanından verileri çek
conn = sqlite3.connect("kutuphane.db")
df = pd.read_sql_query("SELECT * FROM veritabanlari", conn)
conn.close()

# 🔍 Arama ve filtreleme
arama = st.text_input("🔍 Arama (Ad, açıklama veya kategori):")
acik_erisim = st.checkbox("🔒 Sadece Açık Erişim Veritabanları", value=False)
siralama = st.selectbox("📊 Sıralama", ["Varsayılan", "Ad (A-Z)", "Kategori"])

# Filtreleme
filtreli_df = df.copy()
if arama:
    filtreli_df = filtreli_df[filtreli_df.apply(lambda x: arama.lower() in x.to_string().lower(), axis=1)]
if acik_erisim:
    filtreli_df = filtreli_df[filtreli_df["acik_erisim"] == 1]
if siralama == "Ad (A-Z)":
    filtreli_df = filtreli_df.sort_values("ad")
elif siralama == "Kategori":
    filtreli_df = filtreli_df.sort_values(["kategori", "ad"])

```

```

# Listeleme
for kategori, grup in filtreli_df.groupby("kategori"):
    with st.container():
        st.subheader(f"📁 {kategori}")
        for _, row in grup.iterrows():
            st.markdown(f"""
**[{row['ad']}]({row['link']})**
📖 *Açıklama:* {row['aciklama']}
🔒 *Açık Erişim:* {"Evet" if row['acik_erisim'] else "Hayır"}
""")
        st.divider()

# CSV çıktısı ve ana sayfa butonu sabit aşağıda
st.markdown("---")
csv = filtreli_df.to_csv(index=False).encode("utf-8")
st.download_button("📄 Listeyi CSV olarak indir", data=csv,
file_name="veritabani_listesi.csv", mime="text/csv")

# Ana sayfaya dön
if st.button("🏠 Ana Sayfa"):
    st.switch_page("streamlit_app.py")

st.markdown("---") # Ayırıcı çizgi
st.markdown("<br>", unsafe_allow_html=True)

st.markdown("""
<p style='text-align: center;'>
© 2025 <a href='https://batman.edu.tr/Birimler/kutuphane' target='_blank'
style='text-decoration: none; color: inherit;'>
Batman Üniversitesi Kütüphane ve Dokümantasyon Daire Başkanlığı
</a>
</p>
""", unsafe_allow_html=True)

```

D11 pages/hizmetlerimiz.py kodları

```

import streamlit as st

st.set_page_config(page_title="Hizmetlerimiz", page_icon="🌸")
st.title("🌸 Kütüphane Hizmetlerimiz")

st.markdown("""
Batman Üniversitesi Kütüphane ve Dokümantasyon Daire Başkanlığı olarak
sunduğumuz başlıca hizmetler:

### 📖 [Ödünç Verme
Hizmetleri](https://batman.edu.tr/Birimler/kutuphane/sayfalar/17024)

```

Kütüphane koleksiyonundan kitap ve diğer materyallerin ödünç alınmasına ilişkin koşullar ve süreler.

🔄 [Kütüphanelerarası Ödünç Verme](https://batman.edu.tr/Birimler/kutuphane/sayfalar/17025)
Diğer üniversite ve kurum kütüphaneleriyle işbirliği içinde yürütülen kaynak paylaşımı hizmeti.

? [Referans Hizmetleri](https://batman.edu.tr/Birimler/kutuphane/sayfalar/17027)
Akademik çalışmalarda kullanıcılara danışma ve kaynak bulma desteği sağlanır.

📱 [Mobil Kütüphane](https://batman.edu.tr/Birimler/kutuphane/sayfalar/17029)
Mobil cihazlarla kütüphane sistemine erişim sağlayarak kaynak arama ve işlemleri yapabilirsiniz.

♿ [Engelsiz Bilgi Erişim Merkezi](https://batman.edu.tr/Birimler/kutuphane/sayfalar/17036)
Engelli kullanıcıların bilgiye eşit erişimini sağlamak amacıyla özel olarak tasarlanmış hizmetler.

📶 [Eduroam – Kablosuz Ağ Erişimi](http://wifi.batman.edu.tr/wifi/index.php)
Üniversitenin kablosuz ağına güvenli bağlantı için eduroam altyapısı kullanılır.

🔗 [Açık Kaynak Kodlu Yazılımlar](https://batman.edu.tr/Birimler/bidb/sayfalar/17912)
Ücretsiz ve özgür yazılımlara dair bilgiler ile yazılım havuzlarına erişim.
""")

```
# Ana sayfaya dön
if st.button("🏠 Ana Sayfa"):
    st.switch_page("streamlit_app.py")

st.markdown("---") # Ayırıcı çizgi
st.markdown("<br>", unsafe_allow_html=True)

st.markdown("""
<p style='text-align: center;'>
© 2025 <a href='https://batman.edu.tr/Birimler/kutuphane' target='_blank'
style='text-decoration: none; color: inherit;'>
Batman Üniversitesi Kütüphane ve Dokümantasyon Daire Başkanlığı
</a>
</p>
""", unsafe_allow_html=True)
```

D12 pages/hakkimizda.py kodları

```
import streamlit as st

st.set_page_config(page_title="Hakkımızda", page_icon="🏛️")
st.title("🏛️ Hakkımızda")

st.markdown("""
Batman Üniversitesi Kütüphane ve Dokümantasyon Daire Başkanlığı'na
ilişkin detaylı kurumsal bilgiler:

---

### 📖 Genel Bilgiler
- [Yönetici Sunuşu](https://batman.edu.tr/Birimler/kutuphane/sayfalar/19388)
- [Kuruluş](https://batman.edu.tr/Birimler/kutuphane/sayfalar/17285)
- [Misyonumuz](https://batman.edu.tr/Birimler/kutuphane/sayfalar/17283)
- [Vizyonumuz](https://batman.edu.tr/Birimler/kutuphane/sayfalar/17284)
- [Temel Değerler](https://batman.edu.tr/Birimler/kutuphane/sayfalar/17017)
- [Paydaşlarımız](https://batman.edu.tr/Birimler/kutuphane/sayfalar/19542)
- [Amaç ve Hedeflerimiz](https://batman.edu.tr/Birimler/kutuphane/sayfalar/17018)

---

### 🏛️ Başkanlığımız
- [İdari Kadro](https://batman.edu.tr/Birimler/kutuphane/idari-kadro)
- [Görev, Yetki ve Sorumluluklar](https://batman.edu.tr/Birimler/1508/sayfalar/21992)
- [Organizasyon Şeması (PDF)](https://batman.edu.tr/images/files/İdari%20Birimler/Kutuphane_DB/KDDB%20Kalite%20Güvence%20Sistemi/KDDB_ORGANIZASYON_SEMASI.pdf)
- [Kurullar, Komisyonlar](https://batman.edu.tr/Birimler/kutuphane/sayfalar/18582)
- [Kütüphane Kullanım Koşulları](https://batman.edu.tr/Birimler/kutuphane/sayfalar/17035)
- [Çalışma Saatleri](https://batman.edu.tr/Birimler/kutuphane/sayfalar/17032)

---

### ✅ Kalite Güvencesi
- [Kurumsal](https://batman.edu.tr/Birimler/kutuphane/sayfalar/20816)
- [Mevzuat](https://batman.edu.tr/Birimler/kutuphane/sayfalar/20819)
- [Dokümanlar](https://batman.edu.tr/Birimler/kutuphane/sayfalar/20821)
```

```

---

### 📁 İç Kontrol Sistemi
- [Faaliyet Raporları](https://batman.edu.tr/Birimler/kutuphane/sayfalar/17037)
- [Stratejik Planlar](https://batman.edu.tr/Birimler/kutuphane/sayfalar/19546)

---

### 📁 Belgeler ve Formlar
- [Belgeler ve Formlar](https://batman.edu.tr/Birimler/kutuphane/sayfalar/17033)

---

### 📁 KVKK Bilgilendirmeleri
- [Birim K.V.K.K. Sorumlusu](https://batman.edu.tr/Birimler/kutuphane/sayfalar/19039)
- [K.V.K.K. Aydınlatma Metni](https://batman.edu.tr/Birimler/kutuphane/sayfalar/17994)

---
"""

# Ana sayfaya dön
if st.button("🏠 Ana Sayfa"):
    st.switch_page("streamlit_app.py")

st.markdown("---") # Ayırıcı çizgi
st.markdown("<br>", unsafe_allow_html=True)

st.markdown("""
<p style='text-align: center;'>
@ 2025 <a href='https://batman.edu.tr/Birimler/kutuphane' target='_blank'
style='text-decoration: none; color: inherit;'>
Batman Üniversitesi Kütüphane ve Dokümantasyon Daire Başkanlığı
</a>
</p>
""", unsafe_allow_html=True)

```

D13 pages/duyurular.py kodları

```

import streamlit as st

# Sayfa ayarları
st.set_page_config(page_title="Kütüphane Duyuruları", page_icon="📢")
st.title("📢 Batman Üniversitesi Kütüphane Duyuruları")

```

```

# Statik HTML dosyasından çıkarılmış veriler
duyurular = [
    {
        "tarih": "03 Ocak 2025",
        "baslik": "Akademik Sunum veritabanına 31.12.2025 tarihine kadar
abonelik yapılmıştır.",
        "url": "https://batman.edu.tr/Birimler/1508/haberler/akademik-
sunum-veritabanina-31122025-tarihine-kadar-abonelik-yapilmistir/16056"
    },
    {
        "tarih": "03 Ocak 2025",
        "baslik": "BookCites veritabanına 31.12.2025 tarihine kadar
abonelik yapılmıştır.",
        "url": "https://batman.edu.tr/Birimler/1508/haberler/bookcites-
veritabanina-31122025-tarihine-kadar-abonelik-yapilmistir/16054"
    },
    {
        "tarih": "03 Ocak 2025",
        "baslik": "Sobiad Atıf Dizinine 31.12.2025 tarihine kadar abonelik
yapılmıştır.",
        "url": "https://batman.edu.tr/Birimler/1508/haberler/sobiad-atif-
dizinine-31122025-tarihine-kadar-abonelik-yapilmistir/16050"
    },
    {
        "tarih": "10 Ekim 2025",
        "baslik": "Libraryturk E-Kitap Platformuna 31.12.2025 tarihine
kadar abonelik yapılmıştır.",
        "url":
"https://batman.edu.tr/Birimler/kutuphane/haberler/libraryturk-e-kitap-
platformuna-31122025-tarihine-kadar-abonelik-yapilmistir/15589"
    },
    {
        "tarih": "03 Şubat 2025",
        "baslik": "Kurumsal Online Dil Yönetim Sistemleri (KODSİS)
Veritabanına 31.12.2025 tarihine kadar abonelik yapılmıştır.",
        "url": "https://batman.edu.tr/Birimler/1508/duyurular/kurumsal-
online-dil-yonetim-sistemleri-kodsiss-veritabanina-31122025-tarihine-
kadar-abonelik/12898"
    }
]

# Duyuruları göster
for duyuru in duyurular:
    with st.expander(f"{duyuru['tarih']} - {duyuru['baslik']}"):
        st.markdown(f"\[Detaylı Görüntüle\]")

# Kaynak bağlantısı
st.markdown("---")

```

```

st.markdown("📌 Kaynak: [Resmi Duyuru Sayfası](https://batman.edu.tr/Birimler/kutuphane/tum-duyurular)")

# Ana sayfaya dön
if st.button("🏠 Ana Sayfa"):
    st.switch_page("streamlit_app.py")

st.markdown("---") # Ayırıcı çizgi
st.markdown("<br>", unsafe_allow_html=True)

st.markdown("""
<p style='text-align: center;'>
© 2025 <a href='https://batman.edu.tr/Birimler/kutuphane' target='_blank'
style='text-decoration: none; color: inherit;'>
Batman Üniversitesi Kütüphane ve Dokümantasyon Daire Başkanlığı
</a>
</p>
""", unsafe_allow_html=True)

```

D14 pages/iletisim.py kodları

```

import streamlit as st

st.set_page_config(page_title="İletişim", page_icon="📞")
st.title("📞 İletişim Bilgileri")

st.markdown("""
**Batman Üniversitesi**
**Kütüphane ve Dokümantasyon Daire Başkanlığı**

📍 **Adres:**
Batı Raman Kampüsü, Merkez / BATMAN

📞 **Telefon:**
+90 488 217 3905

📞 **Telefon:**
+90 488 217 4147

✉️ **E-posta:**
kutuphane@batman.edu.tr

🌐 [Resmi Web Sayfası](https://batman.edu.tr/Birimler/kutuphane/sayfalar/17995)
""")

# Ana sayfaya dön
if st.button("🏠 Ana Sayfa"):

```

```

st.switch_page("streamlit_app.py")

st.markdown("---") # Ayırıcı çizgi
st.markdown("<br>", unsafe_allow_html=True)

st.markdown("""
<p style='text-align: center;'>
@ 2025 <a href='https://batman.edu.tr/Birimler/kutuphane' target='_blank'
style='text-decoration: none; color: inherit;'>
Batman Üniversitesi Kütüphane ve Dokümantasyon Daire Başkanlığı
</a>
</p>
""", unsafe_allow_html=True)

```

EK-5 Yönetici Modülü

No	Dosya	Görev
E1	pages/admin_dashboard.py	Çok-sekme ana panel

E2	admin_sections/ kullanici_yonetimi.py	
E3	admin_sections/ kullanici_inceleme.py	
E4	admin_sections/ geri_bildirim.py	
E5	admin_sections/ favori_analiz.py	
E6	admin_sections/ model_karsilastirma.py	
E7	admin_sections/ egitim_loglari.py	
E8	admin_sections/ sorgu_suresi_analizi.py	
E9	admin_sections/ rlhf.py	
E10	admin_sections/ model_egitime.py	
E11	admin_sections/ veri_aktar.py	
E12	model_karsilastirma_kayit.py	DB kayıt helper

E1 pages/admin_dashboard.py kodları

```

import os
import streamlit as st
from pathlib import Path
import pandas as pd
from utils import read_log_file # utils.py'den içe aktar
from utils import apply_dark_mode
# Bölüm modülleri
import admin_sections.kullanici_yonetimi as kullanici_yonetimi
import admin_sections.geri_bildirim as geri_bildirim
import admin_sections.favori_analiz as favori_analiz
import admin_sections.veri_aktar as veri_aktar
import admin_sections.rlhf as rlhf
import admin_sections.model_karsilastirma as model_karsilastirma
import admin_sections.kullanici_inceleme as kullanici_inceleme
import admin_sections.egitim_loglari as egitim_loglari
import admin_sections.model_egitime as model_egitime
import admin_sections.sorgu_suresi_analizi as sorgu_suresi_analizi

# Sayfa yapılandırması
st.set_page_config(page_title="Admin Panel Dashboard", page_icon="🛡️",
layout="wide")
if "dark_mode" not in st.session_state:
    st.session_state.dark_mode = False

col1, col2, col3 = st.columns([8, 1, 1])
with col3:
    if st.button("🌙" if not st.session_state.dark_mode else "☀️",
help="Gece Modu Değiştir"):
        st.session_state.dark_mode = not st.session_state.dark_mode

if st.session_state.dark_mode:
    apply_dark_mode()

# Oturum ve Admin kontrolü
if "authenticated" not in st.session_state or not
st.session_state["authenticated"]:
```

```

        st.warning("🔒 Bu sayfayı görüntülemek için lütfen giriş yapınız.")
        st.stop()
if not st.session_state.get("admin", False):
    st.error("🚫 Bu sayfa yalnızca yöneticiler içindir.")
    st.stop()

st.title("🏠 Admin Dashboard")

# Sidebar menüsünü gruplandırılmış hale getirme
st.sidebar.title("📁 Admin Menü")

# Seçilen sayfayı session_state ile yönet
if "secilen_sayfa" not in st.session_state:
    st.session_state.secilen_sayfa = "Kullanıcı Yönetimi"

# Hata ayıklama için geçici kontrol (isteğe bağlı kaldırılabilir)
st.write(f"Debug: Şu anki sayfa = {st.session_state.secilen_sayfa}") #
Hangi sayfanın seçili olduğunu kontrol et

# Kullanıcı Yönetimi
with st.sidebar.expander("👤 Kullanıcı Yönetimi", expanded=True):
    if st.button("Kullanıcı Yönetimi", key="kullanici_yonetimi_btn"):
        st.session_state.secilen_sayfa = "Kullanıcı Yönetimi"
    if st.button("Kullanıcı Bazlı İnceleme",
key="kullanici_inceleme_btn"):
        st.session_state.secilen_sayfa = "Kullanıcı Bazlı İnceleme"

# Analiz ve Raporlama
with st.sidebar.expander("📊 Analiz ve Raporlama"):
    if st.button("Geri Bildirim Analizi", key="geri_bildirim_btn"):
        st.session_state.secilen_sayfa = "Geri Bildirim Analizi"
    if st.button("Favori Analizi", key="favori_analiz_btn"):
        st.session_state.secilen_sayfa = "Favori Analizi"
    if st.button("Sorgu Süresi Analizi", key="sorgu_suresi_btn"):
        st.session_state.secilen_sayfa = "Sorgu Süresi Analizi"

# Veri Yönetimi
with st.sidebar.expander("📁 Veri Yönetimi"):
    if st.button("Veri Aktar / İçer Aktar", key="veri_aktar_btn"):
        st.session_state.secilen_sayfa = "Veri Aktar / İçer Aktar"
    if st.button("RLHF Analizi", key="rlhf_btn"):
        st.session_state.secilen_sayfa = "RLHF Analizi"

# Model Yönetimi
with st.sidebar.expander("🧠 Model Yönetimi"):
    if st.button("Model Karşılaştırma", key="model_karsilastirma_btn"):
        st.session_state.secilen_sayfa = "Model Karşılaştırma"
    if st.button("Eğitim Logları", key="egitim_loglari_btn"):
        st.session_state.secilen_sayfa = "Eğitim Logları"

```

```

        if st.button("Model Eğitme", key="model_egitme_btn"):
            st.session_state.secilen_sayfa = "Model Eğitme"

# Sayfa yönlendirme
if st.session_state.secilen_sayfa == "Kullanıcı Yönetimi":
    st.write("Yönlendirme: Kullanıcı Yönetimi") # Hata ayıklama
    kullanici_yonetimi.render()
elif st.session_state.secilen_sayfa == "Geri Bildirim Analizi":
    st.write("Yönlendirme: Geri Bildirim Analizi") # Hata ayıklama
    geri_bildirim.render()
elif st.session_state.secilen_sayfa == "Favori Analizi":
    st.write("Yönlendirme: Favori Analizi") # Hata ayıklama
    favori_analiz.render()
elif st.session_state.secilen_sayfa == "Veri Aktar / İçer Aktar":
    st.write("Yönlendirme: Veri Aktar / İçer Aktar") # Hata ayıklama
    veri_aktar.render()
elif st.session_state.secilen_sayfa == "RLHF Analizi":
    st.write("Yönlendirme: RLHF Analizi") # Hata ayıklama
    rlhf.render()
elif st.session_state.secilen_sayfa == "Model Karşılaştırma":
    st.write("Yönlendirme: Model Karşılaştırma") # Hata ayıklama
    model_karsilastirma.render(admin_mode=True)
elif st.session_state.secilen_sayfa == "Kullanıcı Bazlı İnceleme":
    st.write("Yönlendirme: Kullanıcı Bazlı İnceleme") # Hata ayıklama
    kullanici_inceleme.render()
elif st.session_state.secilen_sayfa == "Eğitim Logları":
    st.write("Yönlendirme: Eğitim Logları") # Hata ayıklama
    egitim_loglari.render()
elif st.session_state.secilen_sayfa == "Model Eğitme":
    st.write("Yönlendirme: Model Eğitme") # Hata ayıklama
    model_egitme.render()
elif st.session_state.secilen_sayfa == "Sorgu Süresi Analizi":
    st.write("Yönlendirme: Sorgu Süresi Analizi") # Hata ayıklama
    sorgu_suresi_analizi.render()
else:
    st.write("Hata: Tanımlanmamış sayfa") # Varsayılan durum için hata
    mesajı
    st.session_state.secilen_sayfa = "Kullanıcı Yönetimi" # Varsayılan
    geri dönüş
    kullanici_yonetimi.render()

# Ana sayfa butonu
if st.button("🏠 Ana Sayfa"):
    st.switch_page("streamlit_app.py")

```

E2 admin_sections/kullanici_yonetimi.py kodları

```
import streamlit as st
```

```

import sqlite3
import pandas as pd

def render():
    st.subheader("👤 Tüm Kayıtlı Kullanıcılar")
    conn = sqlite3.connect("kutuphane.db")
    kullanicilar = pd.read_sql_query("SELECT kullanıcı_adi, rol FROM
kullanicilar ORDER BY kullanıcı_adi", conn)
    if not kullanicilar.empty:
        st.dataframe(kullanicilar, use_container_width=True)
    else:
        st.info("Henüz kullanıcı kaydı bulunmamaktadır.")

    # 🔒 Şifre Sıfırlama
    st.subheader("🔒 Şifre Sıfırlama")
    secilen_kullanici = st.selectbox("Bir kullanıcı seçin",
kullanicilar["kullanıcı_adi"] if not kullanicilar.empty else [])
    yeni_sifre = st.text_input("Yeni Şifre", type="password")
    if st.button("Şifreyi Sıfırla") and secilen_kullanici and yeni_sifre:
        try:
            cur = conn.cursor()
            cur.execute("UPDATE kullanicilar SET sifre = ? WHERE
kullanıcı_adi = ?", (yeni_sifre, secilen_kullanici))
            conn.commit()
            st.success(f"✔️ {secilen_kullanici} adlı kullanıcının şifresi
başarıyla sıfırlandı.")
        except Exception as e:
            st.error(f"❌ Şifre sıfırlama hatası: {e}")

    # ✂ Rol Güncelleme
    st.subheader("✂ Rol Güncelleme")
    secilen_kullanici_rol = st.selectbox("Rolünü güncellemek istediğiniz
kullanıcıyı seçin", kullanicilar["kullanıcı_adi"] if not
kullanicilar.empty else [])
    yeni_rol = st.selectbox("Yeni Rol Seçin", ["admin", "kullanici"])
    if st.button("Rolü Güncelle") and secilen_kullanici_rol and yeni_rol:
        try:
            cur = conn.cursor()
            cur.execute("UPDATE kullanicilar SET rol = ? WHERE
kullanıcı_adi = ?", (yeni_rol, secilen_kullanici_rol))
            conn.commit()
            st.success(f"✔️ {secilen_kullanici_rol} adlı kullanıcının
rolü başarıyla '{yeni_rol}' olarak güncellendi.")
        except Exception as e:
            st.error(f"❌ Rol güncelleme hatası: {e}")

    # Veritabanı bağlantısını kapat
    conn.close()

```

E3 admin_sections/kullanici_inceleme.py kodları

```
import streamlit as st
import sqlite3
import pandas as pd

def render():
    st.subheader("📁 Kullanıcı Bazlı Veriler")

    # Veritabanına bağlan
    conn = sqlite3.connect("kutuphane.db")

    # Tüm kullanıcıları listele
    tum_kullaniciilar = pd.read_sql_query("SELECT kullanıcı_adi FROM
kullaniciilar", conn)["kullanıcı_adi"].tolist()
    if tum_kullaniciilar:
        kullanıcı_sec = st.selectbox("Bir kullanıcı seçin:",
tum_kullaniciilar)

        # Geri bildirimleri göster
        st.markdown("---")
        st.markdown(f"### 🗉 {kullanıcı_sec} kullanıcısının Geri
Bildirimleri")
        geri = pd.read_sql_query(f"""
        SELECT tarih, sorgu, kaynak, puan, hallucination
        FROM geri_bildirim
        WHERE kullanıcı_adi = '{kullanıcı_sec}'
        ORDER BY tarih DESC
        """, conn)
        st.dataframe(geri, use_container_width=True)

        # Favorileri göster
        st.markdown("---")
        st.markdown(f"### ★ {kullanıcı_sec} kullanıcısının Favorileri")
        fav_vt = pd.read_sql_query(f"""
        SELECT v.ad, v.aciklama
        FROM favoriler f
        JOIN veritabanlari v ON f.veritabanı_id = v.id
        WHERE f.kullanıcı_adi = '{kullanıcı_sec}'
        ORDER BY f.id DESC
        """, conn)
        st.dataframe(fav_vt, use_container_width=True)

        # Model karşılaştırmaları göster
        st.markdown("---")
        st.markdown(f"### 📊 {kullanıcı_sec} kullanıcısının Model
Karşılaştırmaları")
```

```

modeller = pd.read_sql_query(f"""
    SELECT model, yanit_suresi, token_sayisi, yanit_uzunlugu,
puan, tarih
    FROM model_karsilastirma
    WHERE kullanıcı_adi = '{kullanici_sec}'
    ORDER BY tarih DESC
""", conn)
st.dataframe(modeller, use_container_width=True)

# RLHF Skorları göster
st.markdown("---")
st.markdown(f"### 🏆 {kullanici_sec} kullanıcısının RLHF
Skorları")
rlhf = pd.read_sql_query(f"""
    SELECT sorgu, model, reward_label, sorgu_tipi,
tarihsel_tutarsizlik, tarih, sorgu_suresi
    FROM reward_etiket
    WHERE kullanıcı_adi = '{kullanici_sec}'
    ORDER BY tarih DESC
""", conn)
if not rlhf.empty:
    # Sorgu süresi istatistiklerini göster
    st.markdown("#### 📊 Sorgu Süresi İstatistikleri")
    avg_time = rlhf["sorgu_suresi"].mean()
    max_time = rlhf["sorgu_suresi"].max()
    min_time = rlhf["sorgu_suresi"].min()
    st.write(f"**Ortalama Sorgu Süresi:** {avg_time:.2f} saniye")
    st.write(f"**En Uzun Sorgu Süresi:** {max_time:.2f} saniye")
    st.write(f"**En Kısa Sorgu Süresi:** {min_time:.2f} saniye")
    st.dataframe(rlhf, use_container_width=True)
else:
    st.info("Henüz RLHF skoru yok.")

# RLHF skorlarının sayısal dönüşümü
label_map = {"kötü": 1, "orta": 2, "iyi": 3}
try:
    if not rlhf.empty:
        sayisal_skorlar = rlhf["reward_label"].map(label_map)
        ort_rlabel = sayisal_skorlar.mean()
    else:
        ort_rlabel = 0
except Exception as e:
    ort_rlabel = 0
    print(f"RLHF skoru hesaplanırken hata oluştu: {e}")

st.metric("📊 Ortalama RLHF Skoru", f"{ort_rlabel:.2f}")

else:
    st.info("Henüz kayıtlı kullanıcı yok.")

```

```
# Veritabanı bağlantısını kapat
conn.close()
```

E4 admin_sections/geri_bildirim.py kodları

```
# admin_sections/kullanici_yonetimi.py
import streamlit as st
import sqlite3
import pandas as pd

def render():
    st.subheader("👤 Tüm Kayıtlı Kullanıcılar")
    conn = sqlite3.connect("kutuphane.db")
    kullanicilar = pd.read_sql_query("SELECT kullanıcı_adi, rol FROM
kullanicilar ORDER BY kullanıcı_adi", conn)
    if not kullanicilar.empty:
        st.dataframe(kullanicilar, use_container_width=True)
    else:
        st.info("Henüz kullanıcı kaydı bulunmamaktadır.")

    st.subheader("🔒 Şifre Sıfırlama")
    secilen_kullanici = st.selectbox("Bir kullanıcı seçin",
kullanicilar["kullanıcı_adi"] if not kullanicilar.empty else [])
    yeni_sifre = st.text_input("Yeni Şifre", type="password")
    if st.button("Şifreyi Sıfırla") and secilen_kullanici and yeni_sifre:
        cur = conn.cursor()
        cur.execute("UPDATE kullanicilar SET sifre = ? WHERE
kullanıcı_adi = ?", (yeni_sifre, secilen_kullanici))
        conn.commit()
        st.success(f"✅ {secilen_kullanici} adlı kullanıcının şifresi
başarıyla sıfırlandı.")
        conn.close()

# admin_sections/geri_bildirim.py
import streamlit as st
import sqlite3
import pandas as pd

def render():
    st.subheader("🗣️ Geri Bildirim Analizi")
    alt_secim = st.radio("İnceleme Tipi Seçin", ["📁 Geri Bildirim
Logları", "📊 Puan Sıralaması"])

    conn = sqlite3.connect("kutuphane.db")

    if alt_secim == "📁 Geri Bildirim Logları":
```

```

logs = pd.read_sql_query("SELECT kullanıcı_adi, tarih, sorgu,
kaynak, puan, hallucination FROM geri_bildirim ORDER BY tarih DESC", conn)
if not logs.empty:
    st.dataframe(logs, use_container_width=True)
else:
    st.info("Henüz geri bildirim verisi yok.")

elif alt_secim == "📊 Puan Sıralaması":
    skorlar = pd.read_sql_query("""
        SELECT kullanıcı_adi, ROUND(AVG(puan), 2) AS ortalama_puan,
COUNT(*) AS toplam
        FROM geri_bildirim
        WHERE puan IS NOT NULL
        GROUP BY kullanıcı_adi
        ORDER BY ortalama_puan DESC
    """, conn)
    if not skorlar.empty:
        st.dataframe(skorlar, use_container_width=True)
    else:
        st.info("Henüz değerlendirme verisi yok.")

conn.close()

```

E5 admin_sections/favori_analiz.py kodları

```

import streamlit as st
import sqlite3
import pandas as pd

def render():
    st.subheader("★ Favori Analizi")
    conn = sqlite3.connect("kutuphane.db")
    alt_secim = st.radio("Favori Türü", ["📚 Veritabanı Favorileri", "👤 Yanıt Favorileri", "📊 Veritabanı İstatistikleri"])

    if alt_secim == "📚 Veritabanı Favorileri":
        veri = pd.read_sql_query("""
            SELECT f.kullanıcı_adi, v.ad, v.aciklama
            FROM favoriler f
            JOIN veritabanlari v ON f.veritabanı_id = v.id
            ORDER BY f.id DESC
        """, conn)
        if not veri.empty:
            st.dataframe(veri, use_container_width=True)
        else:
            st.info("Henüz veritabanı favorisi yok.")

    elif alt_secim == "👤 Yanıt Favorileri":

```

```

try:
    veri = pd.read_sql_query("""
        SELECT kullanıcı_adi, favori_ismi, favori_etiketi,
yanıt_metni
        FROM favori_detay
        ORDER BY rowid DESC
        """, conn)
    if not veri.empty:
        st.dataframe(veri, use_container_width=True)
    else:
        st.info("Henüz yanıt favorisi yok.")
except Exception as e:
    st.error(f"❌ Hata oluştu:\n{e}")

elif alt_secim == "📊 Veritabanı İstatistikleri":
    istatistik = pd.read_sql_query("""
        SELECT v.ad AS Veritabanı, COUNT(f.veritabani_id) AS
Favori_Sayısı
        FROM favoriler f
        JOIN veritabanlari v ON f.veritabani_id = v.id
        GROUP BY v.ad
        ORDER BY Favori_Sayısı DESC
        """, conn)
    if not istatistik.empty:
        st.dataframe(istatistik, use_container_width=True)
    else:
        st.info("Henüz favori istatistiği bulunmamaktadır.")

conn.close()

```

E6 admin_sections/model_karsilastirma.py kodları

```

import streamlit as st
import sqlite3
import pandas as pd
import matplotlib.pyplot as plt
import json
from pathlib import Path

def render(admin_mode=True):
    st.subheader("📊 Model Karşılaştırma")

    # Model listesi
    model_names = ["BERTurk", "DistilBERT", "ElectraTR", "RoBERTa",
"SentimentBERT", "TinyLLaMA"]

    # 1) Model Performans Metrikleri Bölümü
    st.markdown("## Model Performans Metrikleri")

```

```

# Model, mod ve grafik tipi seçimi
selected_model = st.selectbox("📁 Hangi modeli göstermek istersiniz?",
model_names)
display_mode = st.radio("Görüntüleme Modu Seçin:", ["🏆 Eğitimin Son
Durumu", "⌚ Epoch Bazlı", "🕒 Zaman Bazlı"])
chart_mode = st.radio("Grafik Tipi Seçin:", ["📊 Bar Chart", "📈 Line
Chart"])

# Metrik dosyasını oku (metrics klasöründen)
metrics_file = Path(__file__).parent.parent / "metrics" /
f"{selected_model}_metrics.json"
if metrics_file.exists():
    with open(metrics_file, "r", encoding="utf-8") as mf:
        metrics = json.load(mf)
        df_metrics = pd.DataFrame.from_dict(metrics, orient="index",
columns=["Değer"])

# Seçilen modelin metriklerini tablo olarak göster
st.write(f"### {selected_model} Metrikleri")
st.dataframe(df_metrics.style.set_properties(**{'text-align':
'center', 'font-size': '12px'}))

# Grafik gösterimi (Tüm metrikler dahil)
if display_mode == "🏆 Eğitimin Son Durumu":
    st.write(f"### {selected_model} Eğitimin Son Durumu
Metrikleri")
    if not df_metrics.empty:
        chart_data =
df_metrics.reset_index().rename(columns={"index": "Metric"})
        if chart_mode == "📈 Line Chart":
            st.line_chart(chart_data.set_index("Metric"))
        else: # Bar Chart
            st.bar_chart(chart_data.set_index("Metric"))
    else:
        st.warning("Metrik verisi bulunamadı.")
else: # Epoch Bazlı veya Zaman Bazlı
    history_file = Path(__file__).parent.parent / "metrics" /
f"{selected_model}_history.json"
    if history_file.exists():
        hist = pd.read_json(history_file)
        if isinstance(hist, pd.DataFrame) and not hist.empty:
            st.write(f"### {selected_model}
{display_mode.replace('⌚', '').replace('🕒', '')} Metrikler")
            available_metrics = [col for col in hist.columns if
col in df_metrics.index and hist[col].notna().any() and col not in
["epoch", "time"]]
            filtered_hist = hist[available_metrics].copy()
            if not filtered_hist.empty:

```

```

        if display_mode == "⌚ Epoch Bazlı" and "epoch"
in hist.columns:
            index_data = hist["epoch"]
            index_name = "Epoch"
        elif display_mode == "🕒 Zaman Bazlı" and "time"
in hist.columns:
            index_data = hist["time"]
            index_name = "Time"
        else:
            index_data = range(len(filtered_hist))
            index_name = "Index"

        filtered_hist.index = index_data
        filtered_hist.index.name = index_name
        chart_data = filtered_hist.reset_index()
        if chart_mode == "📈 Line Chart":
            st.line_chart(chart_data.set_index(index_name)
)

        else: # Bar Chart
            st.bar_chart(chart_data.set_index(index_name))
    else:
        st.warning("Seçilen modda gösterilecek geçerli
metrik verisi bulunamadı.")
    else:
        st.warning("Epoch/zaman bazlı veri formatı hatalı
veya boş.")
    else:
        st.warning(f"{selected_model}_history.json bulunamadı.
Önce tarihsel metrikleri kaydetmelisiniz.")
    else:
        st.error(f"✗ {selected_model}_metrics.json bulunamadı.")

# 2) Tüm Modellerin Karşılaştırmalı Metrikleri
st.markdown("## Tüm Modellerin Karşılaştırmalı Metrikleri")
all_metrics = {}
for model in model_names:
    path = Path(__file__).parent.parent / "metrics" /
f"{model}_metrics.json"
    if path.exists():
        with open(path, "r", encoding="utf-8") as mf:
            metrics = json.load(mf)
            metrics_df = pd.Series(metrics)
            if not metrics_df.isna().all():
                all_metrics[model] = metrics_df
            else:
                st.warning(f"{model} için metrikler eksik.")
    else:
        st.warning(f"⚠ {model}_metrics.json bulunamadı, bu model
karşılaştırmaya dahil edilmedi.")

```

```

if all_metrics:
    df_all = pd.DataFrame(all_metrics).T
    st.write("### Tüm Modellerin Metrikleri")
    st.dataframe(df_all.style.set_properties(**{'text-align':
'center', 'font-size': '12px'}))
    chart_data = df_all.reset_index().rename(columns={"index":
"Metric"})
    if chart_mode == "📈 Line Chart":
        st.line_chart(chart_data.set_index("Metric"))
    else:
        st.bar_chart(chart_data.set_index("Metric"))
else:
    st.error("❌ Hiçbir modelin istenen metrik dosyası bulunamadı.")

# 3) Epoch/Zaman Bazlı Genel Metrikler (Tüm Modellerin Birleşimi)
st.markdown("### Epoch/Zaman Bazlı Genel Metrikler")
combined_hist = None
for model in model_names:
    history_file = Path(__file__).parent.parent / "metrics" /
f"{model}_history.json"
    if history_file.exists():
        hist = pd.read_json(history_file)
        if isinstance(hist, pd.DataFrame) and not hist.empty:
            # Model adını ekleyerek yinelenenleri önleme
            hist["Model"] = model
            if "epoch" in hist.columns:
                hist["Index"] = hist["epoch"].astype(str) + "_" +
model

            elif "time" in hist.columns:
                hist["Index"] = hist["time"] + "_" + model
            else:
                hist["Index"] = range(len(hist)).astype(str) + "_" +
model

        if combined_hist is None:
            combined_hist = hist
        else:
            combined_hist = pd.concat([combined_hist, hist],
ignore_index=True)

    if combined_hist is not None and not combined_hist.empty:
        st.write(f"### Genel {display_mode.replace('🕒', '').replace('🕒',
'')} Metrikler")
        available_metrics = [col for col in combined_hist.columns if col
in df_metrics.index and combined_hist[col].notna().any() and col not in
["epoch", "time", "Model", "Index"]]
        if available_metrics:

```

```

        # Pivot table oluştur: Her modelin metriklerini ayrı ayrı
göster
        for metric in available_metrics:
            pivot_data = combined_hist.pivot(index="Index",
columns="Model", values=metric)
            st.write(f"#### {metric} Metriği")
            if chart_mode == "📈 Line Chart":
                st.line_chart(pivot_data)
            else: # Bar Chart
                st.bar_chart(pivot_data)
        else:
            st.warning("Genel metrik verisi bulunamadı (tüm değerler
null).")
        else:
            st.warning("Hiçbir model için tarihsel veri bulunamadı. Önce
modelleri eğitmelisiniz.")

    # 4) Veritabanından Model Karşılaştırma Verileri
    st.markdown("## Veritabanından Model Karşılaştırma Verileri")
    conn = sqlite3.connect("kutuphane.db")

    # Kullanıcı filtreleme (Admin / User modu)
    if admin_mode:
        kullanıcı_adi = st.selectbox("Kullanıcı Seçin (Admin Modu)",
["Tüm Kullanıcılar"] +
                                pd.read_sql_query("SELECT DISTINCT
kullanici_adi FROM model_karsilastirma", conn)["kullanici_adi"].tolist())
        if kullanıcı_adi != "Tüm Kullanıcılar":
            query = f"SELECT * FROM model_karsilastirma WHERE
kullanici_adi = '{kullanici_adi}' ORDER BY tarih DESC"
        else:
            query = "SELECT * FROM model_karsilastirma ORDER BY tarih
DESC"
    else:
        kullanıcı_adi = st.session_state.get("kullanici_adi", "")
        if not kullanıcı_adi:
            st.warning("🔒 Lütfen giriş yapınız.")
            conn.close()
            return
        query = f"SELECT * FROM model_karsilastirma WHERE kullanici_adi =
'{{kullanici_adi}}' ORDER BY tarih DESC"

    try:
        model_logs = pd.read_sql_query(query, conn)
        if not model_logs.empty:
            st.dataframe(model_logs, use_container_width=True)

    # Model başarı istatistikleri
    st.markdown("---")

```

```

st.subheader("📊 Model Başarı İstatistikleri")
summary = model_logs.groupby("model").agg(
    toplam_sorgu=('model', 'count'),
    ortalama_puan=('puan', 'mean'),
    ortalama_sure=('yanit_suresi', 'mean'),
    dogruluk_orani=('puan', lambda x: (x >= 4).mean() * 100)
).reset_index()

st.dataframe(summary.round(2), use_container_width=True)

# Grafiksel gösterim
fig, ax = plt.subplots(figsize=(10, 6))
summary.plot(kind='bar', x='model', y='dogruluk_orani', ax=ax,
legend=False, color='skyblue')
plt.ylabel('% Doğruluk')
plt.title('Model Başarı Karşılaştırması')
st.pyplot(fig)

# En başarılı model
st.markdown("### 🏆 En Başarılı Model")
en_basarili = summary.loc[summary["dogruluk_orani"].idxmax()]
st.metric("Model", en_basarili["model"])
st.metric("Doğruluk Oranı (%)",
f"{en_basarili['dogruluk_orani']:.2f}")
st.metric("Ortalama Puan",
f"{en_basarili['ortalama_puan']:.2f}")
st.metric("Ortalama Yanıt Süresi (sn)",
f"{en_basarili['ortalama_sure']:.2f}")
else:
    st.info("Henüz model karşılaştırma verisi yok.")
except Exception as e:
    st.error(f"❌ Model karşılaştırma verisi yüklenemedi: {e}")
finally:
    conn.close()

```

E7 admin_sections/egitim_loglari.py kodları

```

# admin_sections/egitim_loglari.py
import os
import json
import re
import streamlit as st
from pathlib import Path
import pandas as pd

def clean_metric_name(name):
    """Metrik adlarını temizler ve kısaltır."""

```

```

        name = name.lower().replace(" ", "_").replace("[", "").replace("]", "",
    "")
    name = name.replace("ä", "a").replace("ÿ", "i").replace("ö",
    "o").replace("û", "u").replace("ş", "s").replace("ğ", "g").replace("ç",
    "c").replace("ı", "i")
    if "eaiitim_başılama_zamana±" in name:
        return "egitim_baslama_zamani"
    if "eaiitim_tamamlanma_zamana±" in name:
        return "egitim_tamamlanma_zamani"
    if "toplam_kaya±p" in name:
        return "toplam_kayip"
    if "eaiitim_sâ%resi" in name:
        return "egitim_suresi"
    return name

def render():
    st.subheader("📁 Eğitim Logları")

    # Log dosyalarını seçme
    log_dir = "logs/training_logs"
    available_logs = []
    if os.path.exists(log_dir):
        for model_dir in os.listdir(log_dir):
            log_path = os.path.join(log_dir, model_dir, "training.log")
            if os.path.exists(log_path):
                available_logs.append(os.path.join(model_dir,
    "training.log"))

    if not available_logs:
        st.error("❌ Hiçbir log dosyası bulunamadı. Lütfen önce model
    eğitimi yapın.")
        return

    selected_log = st.selectbox("Log dosyalarını seçin:", available_logs)
    log_file_path = os.path.join(log_dir, selected_log)
    metrics_by_time = {} # Zaman bazlı metrikler
    epoch_metrics = {} # Epoch bazlı metrikler
    debug_logs = [] # Debug mesajları için liste
    current_time = None

    if os.path.exists(log_file_path):
        st.write(f"📄 Tespit edilen kodlama: utf-8 (Güven: %99.00)")
        with open(log_file_path, 'r', encoding='utf-8') as file:
            for line_number, line in enumerate(file, 1):
                try:
                    # Zaman damgasını yakala
                    time_match = re.search(r"(\d{4}-\d{2}-\d{2}
    \d{2}:\d{2}:\d{2}),\d{3})", line)

```

```

        if time_match and any(phrase in line for phrase in
["Eğitim başlama zamanı", "Training start time", "[INFO]"]):
            current_time = time_match.group(1).replace(":",
"_" ).replace(",", "_")
            metrics_by_time[current_time] = {}
            debug_logs.append({"Tür": "Zaman Damgası",
"Mesaj": f"Yeni zaman damgası bulundu - {current_time} (Satır
{line_number})"})

        # JSON formatını kontrol et
        if current_time and "{" in line and "}" in line:
            json_str = re.search(r"\{.*\}", line)
            if json_str:
                json_str = json_str.group(0).replace("'", '"')
                data = json.loads(json_str)
                if "loss" in data or "eval_precision" in data
or "grad_norm" in data:
                    epoch_metrics.setdefault("epoch_data",
[]).append(data)
                    debug_logs.append({"Tür": "Epoch Verisi",
"Mesaj": f"Epoch verisi bulundu - {data} (Satır {line_number})"})
                elif current_time:
                    # Düz metin formatı için
                    value = re.search(r"([0-9]+\.[0-9]+)", line)
                    if value and ":" in line:
                        metric_name =
clean_metric_name(line.split(":")[0].strip())
                        metrics_by_time[current_time][metric_name] =
float(value.group(0))
                        debug_logs.append({"Tür": "Metrik", "Mesaj":
f"Metrik eklendi - {metric_name}: {value.group(0)} (Satır
{line_number})"})
            except (ValueError, json.JSONDecodeError) as e:
                debug_logs.append({"Tür": "Hata", "Mesaj": f"Hata: {e}
- Satır {line_number}: {line.strip()}"})
                continue

        # Debug mesajlarını tablo formatında göster
        if debug_logs:
            with st.expander("🔍 Debug Mesajları", expanded=False):
                debug_df = pd.DataFrame(debug_logs)
                st.dataframe(debug_df.style.set_properties(**{'text-
align': 'left', 'font-size': '12px'}))

        # Metrikleri düzenleme
        if metrics_by_time:
            st.write("### Eğitim Metrikleri (Zaman Bazlı)")

```

```

        df_metrics = pd.DataFrame.from_dict(metrics_by_time,
orient='index')
        if df_metrics.empty:
            st.warning("⚠ Zaman bazlı metrikler tablosu boş.")
        else:
            st.dataframe(df_metrics.style.set_properties(**{'text-align': 'center', 'font-size': '12px'}))
        else:
            st.warning("⚠ Hiçbir zaman bazlı metrik bulunamadı.")

    if epoch_metrics.get("epoch_data"):
        st.write("### Epoch Bazlı Metrikler")
        epoch_df = pd.DataFrame(epoch_metrics["epoch_data"])
        if epoch_df.empty:
            st.warning("⚠ Epoch bazlı metrikler tablosu boş.")
        else:
            st.dataframe(epoch_df.style.set_properties(**{'text-align': 'center', 'font-size': '12px'}))
        else:
            st.warning("⚠ Hiçbir epoch bazlı metrik bulunamadı.")

    with open(log_file_path, 'r', encoding='utf-8') as file:
        st.text_area("Tam Log İçeriği", file.read(), height=300)
    else:
        st.error(f"✗ {log_file_path} dosyası bulunamadı.")

```

E8 admin_sections/sorgu_suresi_analizi.py kodları

```

# admin_sections/sorgu_suresi_analizi.py
import streamlit as st
import sqlite3
import pandas as pd

def render():
    st.subheader("🕒 Sorgu Süresi Analizi")
    conn = sqlite3.connect("kutuphane.db")
    try:
        sorgu_sureleri = pd.read_sql_query("""
            SELECT kullanıcı_adi, sorgu, kaynak, tarih, sorgu_suresi
            FROM sorgu_gecmisi
            WHERE sorgu_suresi > 0
            ORDER BY tarih DESC
            """, conn)
        if not sorgu_sureleri.empty:
            st.markdown("#### 📊 Sorgu Süresi İstatistikleri")
            avg_time = sorgu_sureleri["sorgu_suresi"].mean()
            max_time = sorgu_sureleri["sorgu_suresi"].max()
            min_time = sorgu_sureleri["sorgu_suresi"].min()
            st.write(f"***Ortalama Sorgu Süresi:** {avg_time:.2f} saniye")

```

```

st.write(f"***En Uzun Sorgu Süresi:** {max_time:.2f} saniye")
st.write(f"***En Kısa Sorgu Süresi:** {min_time:.2f} saniye")
st.markdown("#### 📊 Sorgu Süresi Dağılımı")
st.line_chart(sorgu_sureleri[["sorgu_suresi"]])
st.markdown("#### 📁 Tüm Sorgu Süreleri")
st.dataframe(sorgu_sureleri, use_container_width=True)
# Sorgu sürelerini indirme
csv = sorgu_sureleri.to_csv(index=False)
st.download_button(
    label="📄 Sorgu Sürelerini İndir",
    data=csv,
    file_name="sorgu_sureleri.csv",
    mime="text/csv"
)
else:
    st.info("Henüz sorgu süresi verisi yok.")
except Exception as e:
    st.error(f"⚠️ Sorgu süreleri yüklenemedi: {e}")
finally:
    conn.close()

```

E9 admin_sections/rlhf.py kodları

```

# admin_sections/rlhf.py
import streamlit as st
import sqlite3
import pandas as pd

def render():
    st.subheader("🏆 RLHF Reward Model Skorları")
    alt_secim = st.radio("İnceleme Tipi", ["🔍 Tüm Reward Skorları"])

    conn = sqlite3.connect("kutuphane.db")

    if alt_secim == "🔍 Tüm Reward Skorları":
        try:
            reward_logs = pd.read_sql_query("""
                SELECT kullanıcı_adi, sorgu, model, reward_label,
                sorgu_tipi, tarihsel_tutarsizlik, tarih, sorgu_suresi
                FROM reward_etiket
                ORDER BY tarih DESC
            """, conn)
            if not reward_logs.empty:
                # Sorgu süresi istatistiklerini göster
                st.markdown("#### 📊 Sorgu Süresi İstatistikleri")
                avg_time = reward_logs["sorgu_suresi"].mean()
                max_time = reward_logs["sorgu_suresi"].max()
                min_time = reward_logs["sorgu_suresi"].min()

```

```

        st.write(f"**Ortalama Sorgu Süresi:** {avg_time:.2f}
saniye")
        st.write(f"**En Uzun Sorgu Süresi:** {max_time:.2f}
saniye")
        st.write(f"**En Kısa Sorgu Süresi:** {min_time:.2f}
saniye")
        st.dataframe(reward_logs, use_container_width=True)
    else:
        st.info("Henüz reward etiketi verisi yok.")
except Exception as e:
    st.error(f"⚠️ Reward skorları yüklenemedi: {e}")

conn.close()

```

E10 admin_sections/model_egitme.py kodları

```

# admin_sections/model_egitme.py
import streamlit as st
import subprocess
from settings import MODEL_PATHS
import os
from pathlib import Path

LOG_DIR = "logs/training_logs"

def render():
    st.subheader("🔧 Model Eğitim")
    model_options = ["Tüm Modelleri Eğit"] + list(MODEL_PATHS.keys())
    col1, col2 = st.columns([3, 1])
    with col1:
        selected_model = st.selectbox("Eğitmek istediğiniz modeli seçin:",
model_options)
    with col2:
        if st.button("🏠 Model Eğit"):
            # Log dizinini oluştur
            os.makedirs(LOG_DIR, exist_ok=True)
            if selected_model == "Tüm Modelleri Eğit":
                command = ["python", "chain_fine_tune.py", "--all"]
            else:
                command = ["python", "chain_fine_tune.py", "--model",
selected_model]
            with st.spinner("⚡ Model eğitiliyor, lütfen bekleyiniz..."):
                try:
                    result = subprocess.run(command, capture_output=True,
text=True, shell=True, check=True)
                    st.success(f"✅ Model eğitimi tamamlandı:
{selected_model}")
                    st.text_area("📄 Eğitim Logları", result.stdout,
height=300)

```

```

        st.info(f"📁 Logların kaydedildiği dizin: {LOG_DIR}")
        # Log dosyasını kontrol et ve indirme
        log_file_path = os.path.join(LOG_DIR, selected_model,
"training.log")
        if os.path.exists(log_file_path):
            with open(log_file_path, "r", encoding="utf-8")
as log_file:
                log_content = log_file.read()
                st.download_button(
                    label="📄 Eğitim Loglarını İndir",
                    data=log_content,
                    file_name=f"training_log_{selected_model}.txt",
                    mime="text/plain"
                )
            else:
                st.warning("⚠️ Log dosyası oluşturulmadı. Lütfen
dizini kontrol edin.")
        except subprocess.CalledProcessError as e:
            st.error(f"❌ Hata oluştu: {e.stderr}")
        except Exception as e:
            st.error(f"❌ Beklenmedik hata: {e}")

```

E11 admin_sections/veri_aktar.py kodları

```

# admin_sections/kullanici_yonetimi.py
import streamlit as st
import sqlite3
import pandas as pd

def render():
    st.subheader("👤 Tüm Kayıtlı Kullanıcılar")
    conn = sqlite3.connect("kutuphane.db")
    kullanicilar = pd.read_sql_query("SELECT kullanıcı_adi, rol FROM
kullanicilar ORDER BY kullanıcı_adi", conn)
    if not kullanicilar.empty:
        st.dataframe(kullanicilar, use_container_width=True)
    else:
        st.info("Henüz kullanıcı kaydı bulunmamaktadır.")

    st.subheader("🔒 Şifre Sıfırlama")
    secilen_kullanici = st.selectbox("Bir kullanıcı seçin",
kullanicilar["kullanici_adi"] if not kullanicilar.empty else [])
    yeni_sifre = st.text_input("Yeni Şifre", type="password")
    if st.button("Şifreyi Sıfırla") and secilen_kullanici and yeni_sifre:
        cur = conn.cursor()
        cur.execute("UPDATE kullanicilar SET sifre = ? WHERE
kullanici_adi = ?", (yeni_sifre, secilen_kullanici))
        conn.commit()

```

```

        st.success(f"✔ {secilen_kullanici} adlı kullanıcının şifresi
başarıyla sıfırlandı.")
    conn.close()
# admin_sections/geri_bildirim.py
import streamlit as st
import sqlite3
import pandas as pd

def render():
    st.subheader("🔍 Geri Bildirim Analizi")
    alt_secim = st.radio("İnceleme Tipi Seçin", ["📁 Geri Bildirim
Logları", "📊 Puan Sıralaması"])

    conn = sqlite3.connect("kutuphane.db")

    if alt_secim == "📁 Geri Bildirim Logları":
        logs = pd.read_sql_query("SELECT kullanıcı_adi, tarih, sorgu,
kaynak, puan, hallucination FROM geri_bildirim ORDER BY tarih DESC", conn)
        if not logs.empty:
            st.dataframe(logs, use_container_width=True)
        else:
            st.info("Henüz geri bildirim verisi yok.")

    elif alt_secim == "📊 Puan Sıralaması":
        skorlar = pd.read_sql_query("""
            SELECT kullanıcı_adi, ROUND(AVG(puan), 2) AS ortalama_puan,
COUNT(*) AS toplam
            FROM geri_bildirim
            WHERE puan IS NOT NULL
            GROUP BY kullanıcı_adi
            ORDER BY ortalama_puan DESC
        """, conn)
        if not skorlar.empty:
            st.dataframe(skorlar, use_container_width=True)
        else:
            st.info("Henüz değerlendirme verisi yok.")

    conn.close()

# admin_sections/favori_analizi.py
import streamlit as st
import sqlite3
import pandas as pd

def render():
    st.subheader("★ Favori Analizi")
    alt_secim = st.radio("Favori Türü", ["📁 Veritabanı Favorileri", "🗨️
Yanıt Favorileri", "📊 Veritabanı İstatistikleri"])

```

```

conn = sqlite3.connect("kutuphane.db")

if alt_secim == "📖 Veritabanı Favorileri":
    veri = pd.read_sql_query("""
        SELECT f.kullanici_adi, v.ad, v.aciklama
        FROM favoriler f
        JOIN veritabanlari v ON f.veritabani_id = v.id
        ORDER BY f.id DESC
    """, conn)
    if not veri.empty:
        st.dataframe(veri, use_container_width=True)
    else:
        st.info("Henüz veritabanı favorisi yok.")

elif alt_secim == "🗨️ Yanıt Favorileri":
    try:
        veri = pd.read_sql_query("""
            SELECT kullanıcı_adi, favori_ismi, favori_etiketi,
            yanıt_metni
            FROM favori_detay
            ORDER BY rowid DESC
        """, conn)
        if not veri.empty:
            st.dataframe(veri, use_container_width=True)
        else:
            st.info("Henüz yanıt favorisi yok.")
    except Exception as e:
        st.error(f"❌ Hata oluştu:\n{e}")

elif alt_secim == "📊 Veritabanı İstatistikleri":
    istatistik = pd.read_sql_query("""
        SELECT v.ad AS Veritabanı, COUNT(f.veritabani_id) AS
        Favori_Sayısı
        FROM favoriler f
        JOIN veritabanlari v ON f.veritabani_id = v.id
        GROUP BY v.ad
        ORDER BY Favori_Sayısı DESC
    """, conn)
    if not istatistik.empty:
        st.dataframe(istatistik, use_container_width=True)
    else:
        st.info("Henüz favori istatistiği bulunmamaktadır.")

conn.close()

# admin_sections/veri_aktar.py
import streamlit as st
import sqlite3

```

```

import pandas as pd

def render():
    st.subheader("📄 Verileri Dışa Aktar")
    conn = sqlite3.connect("kutuphane.db")

    sorgular = pd.read_sql_query("SELECT * FROM geri_bildirim", conn)
    favoriler = pd.read_sql_query("SELECT * FROM favoriler", conn)
    kullanicilar = pd.read_sql_query("SELECT * FROM kullanicilar", conn)

    st.markdown("### 📄 Sorgular")
    st.download_button("↓ Sorgu Verilerini İndir",
sorgular.to_csv(index=False).encode("utf-8"), "sorgular.csv", "text/csv")

    st.markdown("### ★ Favoriler")
    st.download_button("↓ Favori Verilerini İndir",
favoriler.to_csv(index=False).encode("utf-8"), "favoriler.csv",
"text/csv")

    st.markdown("### 👤 Kullanıcılar")
    st.download_button("↓ Kullanıcı Verilerini İndir",
kullanicilar.to_csv(index=False).encode("utf-8"), "kullanicilar.csv",
"text/csv")

    conn.close()

```

E12 model_karsilastirma_kayit.py kodları

```

# model_karsilastirma_kayit.py
import sqlite3
from datetime import datetime

def model_karsilastirma_kaydet(kullanici_adi, sorgu, model, yanit,
yanit_suresi, token_sayisi, yanit_uzunlugu, puan):
    """
    Çoklu model A/B testinden elde edilen verileri model_karsilastirma
    tablosuna kaydeder.
    """
    try:
        conn = sqlite3.connect("kutuphane.db")
        cur = conn.cursor()

        cur.execute("""
            INSERT INTO model_karsilastirma (
                kullanici_adi, sorgu, model, yanit,
                yanit_suresi, token_sayisi, yanit_uzunlugu,
                puan, tarih
            ) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)
        """, (

```

```
        kullanıcı_adi,  
        sorgu,  
        model,  
        yanıt,  
        yanıt_suresi,  
        token_sayisi,  
        yanıt_uzunlugu,  
        puan,  
        datetime.now().strftime("%Y-%m-%d %H:%M:%S")  
    ))  
  
    conn.commit()  
    conn.close()  
    return True  
  
except Exception as e:  
    print(f"⚠ Karşılaştırma verisi kaydedilemedi: {e}")  
    return False
```

No	Dosya	Görev
F1	chain_fine_tune.py	Çok-model RLHF eğitimi
F2	update_model.py	Inference model deploy
F3	offline_rlhf.py	Gece otomatik eğitim
F4	setup_training_logs.py	Log klasörü/izinleri
F5	indir_modeller.py	HF model indirme

F1 chain_fine_tune.py kodları

```

import os
import sys
import json
import argparse
import sqlite3
from datetime import datetime
import numpy as np
from faker import Faker
import random
from sklearn.metrics import precision_recall_fscore_support
from datasets import load_dataset, Dataset
from transformers import (
    AutoModelForSequenceClassification,
    AutoTokenizer,
    Trainer,
    TrainingArguments,
    EarlyStoppingCallback,
)
from pathlib import Path

# Konsol kodlamasını UTF-8'e zorla
sys.stdout.reconfigure(encoding='utf-8')

# Komut satırı argümanlarını parse et
parser = argparse.ArgumentParser(description="Model training and fine-tuning script")
parser.add_argument("--all", action="store_true", help="Tüm modelleri eğit")
parser.add_argument("--models", type=str, help="Eğitilecek modeller (örneğin: BERTurk, RoBERTa)")
args = parser.parse_args()

# Sabitler
DB_PATH = "chatbot.db"
FINE_DATA_FILE = "fine_data_rlhf.jsonl"
METRICS_DIR = Path("metrics")
METRICS_DIR.mkdir(exist_ok=True)
MODEL_PATHS = {
    "BERTurk": "dbmdz/bert-base-turkish-cased",
    "DistilBERT": "distilbert-base-uncased",
    "ElectraTR": "dbmdz/electra-base-turkish-cased-discriminator",

```

```

        "RoBERTa": "roberta-base",
        "SentimentBERT": "nlptown/bert-base-multilingual-uncased-sentiment",
        "TinyLLaMA": "path/to/TinyLLaMA" # TinyLLaMA model yolunu
güncelleyin
}

# Veriyi sorgu geçmişinden çek ve jsonl dosyasını oluştur
fake = Faker("tr_TR")
def generate_fine_data():
    try:
        conn = sqlite3.connect(DB_PATH)
        conn.text_factory = str
        cur = conn.cursor()
        sorgular = cur.execute("SELECT sorgu, puan FROM sorgu_gecmisi
WHERE puan IS NOT NULL").fetchall()

        with open(FINE_DATA_FILE, "w", encoding="utf-8") as f:
            # Gerçek verileri ekle
            for sorgu, puan in sorgular:
                try:
                    label = int(puan)
                    if label < 0 or label > 5:
                        print(f"⚠ Geçersiz puan atlandı: {puan}")
                        continue
                except ValueError:
                    print(f"⚠ Geçersiz puan değeri atlandı: {puan}")
                    continue
                data = {"text": sorgu.strip(), "label": 0 if label <= 2
else 1} # Daha dengeli bir eşik
                f.write(json.dumps(data, ensure_ascii=False) + "\n")

            # Sahte veriler üret (çeşitliliği artırmak için)
            random.seed() # Rastgeleliği artırmak için seed sıfırlanır
            for _ in range(1000): # 1000 negatif örnek
                sorgu = f"{fake.sentence()} {random.choice(['kötü',
'berbat', 'iğrenç'])} bir deneyim."
                data = {"text": sorgu, "label": 0}
                f.write(json.dumps(data, ensure_ascii=False) + "\n")
            for _ in range(1000): # 1000 pozitif örnek
                sorgu = f"{fake.sentence()} {random.choice(['harika',
'mükemmel', 'olağanüstü'])} bir deneyim."
                data = {"text": sorgu, "label": 1}
                f.write(json.dumps(data, ensure_ascii=False) + "\n")

            print(f"✔ Veriler başarıyla oluşturuldu (gerçek + sahte):
{FINE_DATA_FILE}")
    except Exception as e:
        print(f"✘ Veriler oluşturulurken hata oluştu: {e}")
        with open(FINE_DATA_FILE, "w", encoding="utf-8") as f:

```

```

        sample_data = [
            {"text": "Bu harika bir film!", "label": 1},
            {"text": "Hiç beğenmedim.", "label": 0},
            {"text": "Fena değil, idare eder.", "label": 1},
        ]
        for data in sample_data:
            f.write(json.dumps(data, ensure_ascii=False) + "\n")
        print(f"✓ Geçici veri seti oluşturuldu: {FINE_DATA_FILE}")
    finally:
        conn.close()

# Veri setini oluştur ve böl
if os.path.exists(FINE_DATA_FILE):
    os.remove(FINE_DATA_FILE)
generate_fine_data()

if not os.path.exists(FINE_DATA_FILE):
    print(f"✗ {FINE_DATA_FILE} dosyası bulunamadı. Lütfen veri
oluşturmayı kontrol edin.")
    exit()

# Veri setini yükle ve böl: %60 train, %20 validation, %20 test
try:
    dataset = load_dataset("json", data_files=FINE_DATA_FILE,
split="train")
    train_test_split = dataset.train_test_split(test_size=0.4, seed=42)
    train_dataset = train_test_split["train"]
    test_val_split =
train_test_split["test"].train_test_split(test_size=0.5, seed=42)
    val_dataset = test_val_split["train"]
    test_dataset = test_val_split["test"]

    # Etiket dağılımını kontrol et
    labels = train_dataset["label"]
    label_counts = np.bincount(labels)
    print(f"Etiket Dağılımı (Train): {label_counts}")
    labels_val = val_dataset["label"]
    label_counts_val = np.bincount(labels_val)
    print(f"Etiket Dağılımı (Validation): {label_counts_val}")
    labels_test = test_dataset["label"]
    label_counts_test = np.bincount(labels_test)
    print(f"Etiket Dağılımı (Test): {label_counts_test}")
except Exception as e:
    print(f"✗ Veri seti yüklenirken hata oluştu: {e}")
    exit()

# Performans metriklerini hesapla
def compute_metrics(pred):
    labels = pred.label_ids

```

```

    preds = np.argmax(pred.predictions, axis=1)
    precision, recall, f1, _ = precision_recall_fscore_support(labels,
preds, average='macro')
    return {
        "precision": precision,
        "recall": recall,
        "f1": f1,
    }

# Model seçimi
model_keys = list(MODEL_PATHS.keys())
if args.all:
    selected_models = MODEL_PATHS.items()
elif args.models:
    selected_model_names = [name.strip() for name in
args.models.split(",")]
    invalid_models = [name for name in selected_model_names if name not
in MODEL_PATHS]
    if invalid_models:
        print(f"✗ Geçersiz model isimleri: {invalid_models}")
        exit()
    selected_models = [(name, MODEL_PATHS[name]) for name in
selected_model_names]
else:
    print("\n👉 Eğitmek istediğiniz modelleri seçin:")
    for i, model_key in enumerate(model_keys, 1):
        print(f"[{i}] {model_key}")
    print("[0] Tüm Modelleri Eğit")

    model_choices = input("\nModel numaralarını virgülle ayırarak girin
(örneğin: 1,4) veya tüm modeller için 0: ").strip()

    if model_choices == "0":
        selected_models = MODEL_PATHS.items()
    else:
        try:
            indices = [int(idx.strip()) for idx in
model_choices.split(",")]
            for idx in indices:
                if idx < 1 or idx > len(model_keys):
                    raise ValueError(f"Geçersiz model numarası: {idx}")
            selected_model_names = [model_keys[idx - 1] for idx in
indices]
            selected_models = [(name, MODEL_PATHS[name]) for name in
selected_model_names]
        except ValueError as e:
            print(f"✗ Hata: {e}. Lütfen geçerli model numaraları girin.")
            exit()

```

```

def train_model(model_key, model_path):
    try:
        print(f"\n🐛 Eğitim başlıyor: {model_key}")

        # Metin tabanlı log dosyası için yapılandırma
        log_file_path = os.path.join(f"logs/training_logs/{model_key}",
"training.log")
        os.makedirs(os.path.dirname(log_file_path), exist_ok=True)
        with open(log_file_path, "a", encoding="utf-8-sig") as log_file:
            log_file.write(f"[{model_key}] Eğitim başlama zamanı:
{datetime.now()}\n")

        # Tokenizer yükleme
        tokenizer = AutoTokenizer.from_pretrained(model_path)

        # Tokenizer pad token kontrolü
        if tokenizer.pad_token is None:
            if model_key == "TinyLLaMA":
                tokenizer.add_special_tokens({'pad_token': '[PAD]'})
            else:
                tokenizer.pad_token = tokenizer.eos_token

        # Veriyi tokenize etme
        def tokenize(example):
            return tokenizer(example["text"], truncation=True,
padding="max_length", max_length=256)

        tokenized_train = train_dataset.map(tokenize, batched=True)
        tokenized_val = val_dataset.map(tokenize, batched=True)
        tokenized_test = test_dataset.map(tokenize, batched=True)

        # Modeli yükleme
        model =
AutoModelForSequenceClassification.from_pretrained(model_path,
num_labels=2, ignore_mismatched_sizes=True)

        # TinyLLaMA için token embeddings boyutunu güncelle
        if model_key == "TinyLLaMA":
            model.resize_token_embeddings(len(tokenizer))

        # Eğitim ayarları
        training_args = TrainingArguments(
            output_dir=f"fine_tuned_model/{model_key}",
            eval_strategy="epoch",
            save_strategy="epoch",
            per_device_train_batch_size=1 if model_key == "TinyLLaMA"
else 4,
            per_device_eval_batch_size=8,
            num_train_epochs=5, # Maksimum epoch sayısı

```

```

        logging_dir=f"logs/training_logs/{model_key}",
        logging_steps=10,
        load_best_model_at_end=True,
        metric_for_best_model="eval_loss",
        greater_is_better=False,
        weight_decay=0.01, # Overfitting önlemek için
        save_total_limit=1,
    )

    # Eğitici (Trainer) oluşturma
    trainer = Trainer(
        model=model,
        args=training_args,
        train_dataset=tokenized_train,
        eval_dataset=tokenized_val,
        compute_metrics=compute_metrics,
        callbacks=[EarlyStoppingCallback(early_stopping_patience=1)],
    )

    # Erken durdurma

    # Modeli eğit
    trainer.train()

    # Her epoch sonunda metrikleri kaydet
    history = []
    for epoch in range(int(trainer.state.epoch)):
        eval_results = trainer.evaluate(tokenized_val)
        eval_results["epoch"] = epoch + 1
        eval_results["time"] = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

        # Metrik anahtarlarını yeniden adlandır
        formatted_result = {
            "epoch": eval_results["epoch"],
            "time": eval_results["time"],
            "eval_loss": eval_results.get("eval_loss", 0.0),
            "precision": eval_results.get("precision", 0.0),
            "recall": eval_results.get("recall", 0.0),
            "f1": eval_results.get("f1", 0.0)
        }
        history.append(formatted_result)
        print(f"Epoch {epoch + 1}: {formatted_result}")
        with open(log_file_path, "a", encoding="utf-8-sig") as log_file:
            log_file.write(f"Epoch {epoch + 1}: {formatted_result}\n")

    # History dosyasını metrics klasörüne kaydet
    with open(METRICS_DIR / f"{model_key}_history.json", "w", encoding="utf-8") as f:
        json.dump(history, f, ensure_ascii=False, indent=2)

```

```

# Test seti üzerinde son değerlendirme
test_results = trainer.evaluate(tokenized_test)
print(f"Test Results for {model_key}: {test_results}")
with open(log_file_path, "a", encoding="utf-8-sig") as log_file:
    log_file.write(f"Test Results: {test_results}\n")

# Metrics dosyasını metrics klasörüne kaydet
with open(METRICS_DIR / f"{model_key}_metrics.json", "w",
encoding="utf-8") as f:
    json.dump(test_results, f, ensure_ascii=False, indent=2)

# Modeli ve tokenizer'ı kaydetme
model.save_pretrained(f"fine_tuned_model/{model_key}")
tokenizer.save_pretrained(f"fine_tuned_model/{model_key}")

print(f"✓ Eğitim tamamlandı: {model_key} →
fine_tuned_model/{model_key}")
with open(log_file_path, "a", encoding="utf-8-sig") as log_file:
    log_file.write(f"[{model_key}] Eğitim tamamlanma zamanı:
{datetime.now()}\n")

except Exception as e:
    with open(log_file_path, "a", encoding="utf-8-sig") as log_file:
        log_file.write(f"[{model_key}] Hata oluştu: {e}\n")
    print(f"✗ Hata oluştu ({model_key}): {e}")

# Modelleri eğit
for model_key, model_path in selected_models:
    train_model(model_key, model_path)

print(f"✓ Eğitim tamamlandı ve metrikler {METRICS_DIR} klasörüne
kaydedildi.")

```

F2 update_model.py kodları

```

import shutil
import os
import time
from datetime import datetime

MODEL_DIR = "fine_tuned_model"
INFERENCE_DIR = "inference_model"
BACKUP_DIR = "model_backup"
LOG_DIR = "logs/model_updates"
# Log klasörünü oluştur
os.makedirs(LOG_DIR, exist_ok=True)

# Fine-tuned modelleri güncelle

```

```

def update_models():
    log_filename = f"{datetime.now().strftime('%Y-%m-%d_%H-%M-%S')}.log"
    log_file_path = os.path.join(LOG_DIR, log_filename)

    try:
        with open(log_file_path, "w") as log_file:
            # Eski inference modellerini yedekle
            if os.path.exists(INFERENCE_DIR):
                if not os.path.exists(BACKUP_DIR):
                    os.makedirs(BACKUP_DIR)
                backup_path = os.path.join(BACKUP_DIR,
f"inference_backup_{int(time.time())}")
                shutil.copytree(INFERENCE_DIR, backup_path)
                log_file.write(f"💰 Eski modeller yedeklendi:
{backup_path}\n")

            # Yeni modelleri güncelle
            if os.path.exists(MODEL_DIR):
                if os.path.exists(INFERENCE_DIR):
                    shutil.rmtree(INFERENCE_DIR)
                    shutil.copytree(MODEL_DIR, INFERENCE_DIR)
                    log_file.write("✔ Inference modelleri başarıyla
güncellendi.\n")
                else:
                    log_file.write("⚠ Fine-tuned model klasörü bulunamadı.\n")

    except Exception as e:
        with open(log_file_path, "a") as log_file:
            log_file.write(f"✖ Model güncellenirken hata oluştu: {e}\n")
        print(f"⚠ Model güncellenirken hata oluştu: {e}")

if __name__ == "__main__":
    update_models()

```

F3 offline_rlh.py kodları

```

import os
import subprocess
import schedule
import time
from datetime import datetime

# Eğitim komutları
TRAIN_COMMANDS = [
    "python berturk_generate_jsonl.py",
    "python distilbert_generate_jsonl.py",
    "python roberta_generate_jsonl.py",
    "python chain_fine_tune.py"
]

```

```

# Model güncelleme komutu
UPDATE_COMMAND = "python update_model.py"
LOG_DIR = "logs/training_logs"

# Log klasörünü oluştur
os.makedirs(LOG_DIR, exist_ok=True)

# Eğitim ve güncelleme fonksiyonu
def run_training_cycle():
    log_filename = f"{datetime.now().strftime('%Y-%m-%d_%H-%M-%S')}.log"
    log_file_path = os.path.join(LOG_DIR, log_filename)

    try:
        with open(log_file_path, "w") as log_file:
            log_file.write(f"🌀 {datetime.now()} - JSONL dosyaları güncelleniyor...\n")

            # JSONL dosyalarını güncelle
            for command in TRAIN_COMMANDS[:-1]:
                result = subprocess.run(command.split(),
capture_output=True, text=True)
                log_file.write(result.stdout)
                log_file.write(result.stderr)
                if result.returncode != 0:
                    raise Exception(f"⚠️ Komut hatası: {command}\n{result.stderr}")
                log_file.write(f"✅ {command} başarıyla tamamlandı.\n")

            # Zincirli eğitimi başlat
            log_file.write(f"🌀 {datetime.now()} - Zincirli eğitim başlatılıyor...\n")
            result = subprocess.run(TRAIN_COMMANDS[-1].split(),
capture_output=True, text=True)
            log_file.write(result.stdout)
            log_file.write(result.stderr)
            if result.returncode != 0:
                raise Exception(f"⚠️ Zincirli eğitim hatası: {result.stderr}")
            log_file.write(f"✅ Zincirli eğitim başarıyla tamamlandı.\n")

            # Modelleri güncelle
            log_file.write(f"🌀 Modeller güncelleniyor...\n")
            result = subprocess.run(UPDATE_COMMAND.split(),
capture_output=True, text=True)
            log_file.write(result.stdout)
            log_file.write(result.stderr)
            if result.returncode != 0:

```

```

        raise Exception(f"⚠ Model güncelleme hatası:
{result.stderr}")
        log_file.write("✓ Modeller başarıyla güncellendi.\n")

    except Exception as e:
        with open(log_file_path, "a") as log_file:
            log_file.write(f"✗ Hata oluştu: {e}\n")
            print(f"⚠ Eğitim veya güncelleme sırasında hata oluştu: {e}")

# Planlayıcı ayarları
def schedule_training():
    # Her gece saat 03:00'te çalıştır
    schedule.every().day.at("03:00").do(run_training_cycle)

    print("⌚ Otomatik eğitim döngüsü başlatıldı...")
    while True:
        schedule.run_pending()
        time.sleep(60)

if __name__ == "__main__":
    schedule_training()

```

F4 setup_training_logs.py kodları

```

import os
import subprocess

# Ana klasör yolu
base_dir = r"D:\tez program\çalışan\batman_infozekai_final_REBUILT\logs"

# Ana logs klasörü
logs_dir = os.path.join(base_dir, "training_logs")
models = ["BERTurk", "DistilBERT", "ElectraTR", "RoBERTa",
"SentimentBERT"]

# Ana logs klasörünü oluştur
os.makedirs(logs_dir, exist_ok=True)

# Model klasörlerini oluştur
for model in models:
    model_dir = os.path.join(logs_dir, model)
    os.makedirs(model_dir, exist_ok=True)
    print(f"✓ {model_dir} klasörü başarıyla oluşturuldu.")

# İzinleri ayarlama (Everyone için tam erişim)
try:
    subprocess.run(f'icaccls "{logs_dir}" /grant Everyone:F /T',
    shell=True, check=True)

```

```

    print(f"✔ {logs_dir} klasörü ve alt klasörlerine tam erişim izni
verildi.")
except subprocess.CalledProcessError as e:
    print(f"✗ İzin ayarı başarısız oldu: {e}")

print("🎉 Tüm klasörler ve izinler başarıyla ayarlandı!")

```

F5 indir_modeller.py kodları

```

import os
from transformers import AutoTokenizer,
AutoModelForSequenceClassification, AutoModelForCausalLM

def indir_model_ve_tokenizer(model_id, klasor, model_type):
    try:
        print(f"\n📥 İndiriliyor: {model_id}")
        os.makedirs(klasor, exist_ok=True)

        tokenizer = AutoTokenizer.from_pretrained(model_id)
        tokenizer.save_pretrained(klasor)

        if model_type == "sequence":
            model =
AutoModelForSequenceClassification.from_pretrained(model_id)
        elif model_type == "causal":
            model = AutoModelForCausalLM.from_pretrained(model_id)
        else:
            raise ValueError(f"Bilinmeyen model tipi: {model_type}")

        model.save_pretrained(klasor)
        print(f"✔ Kaydedildi: {klasor}")

    except Exception as e:
        print(f"✗ HATA - {model_id}: {str(e)}")

def main():
    MODELLER = [
        {"id": "dbmdz/bert-base-turkish-cased", "klasor":
"models/berturk", "tip": "sequence"},
        {"id": "dbmdz/distilbert-base-turkish-cased", "klasor":
"models/distilbert", "tip": "sequence"},
        {"id": "TinyLlama/TinyLlama-1.1B-Chat-v1", "klasor":
"models/tinyllama", "tip": "causal"},
        {"id": "akdenizuniversity/Electra-small-turkish-cased-
discriminator", "klasor": "models/electra", "tip": "sequence"},
        {"id": "osmanthus/turkish-roberta-small", "klasor":
"models/roberta", "tip": "sequence"},
        {"id": "savasy/bert-base-turkish-sentiment-cased", "klasor":
"models/sentiment_bert", "tip": "sequence"},

```

```
]

for m in MODELLER:
    indir_model_ve_tokenizer(m["id"], m["klasor"], m["tip"])

if __name__ == "__main__":
    main()
```