Akademik Makale Analiz ve Özet Aracı

Proje Genel Bakış

Proje Adı: PaperMate (veya başka bir isim seçebilirsiniz)

Hedef: Öğrenciler ve araştırmacılar için akademik makaleleri hızlı analiz eden, özetleyen ve soru-cevap yapılabilen bir asistan.

Tahmini Süre: 8-10 hafta

Ekip Büyüklüğü: 8 kişi

Teknoloji Stack'i

Backend

- **Python 3.10+**
- **LangChain**: LLM orkestrasyon
- **Ollama/Llama.cpp**: Local model çalıştırma (Llama 3.1 8B veya Mistral 7B)
- **ChromaDB**: Vector database
- **FastAPI**: REST API
- **PyPDF2/PyMuPDF**: PDF parsing
- **Sentence Transformers**: Embedding modeli

Frontend

- **Streamlit** veya **Gradio**: Hızlı prototipleme için
- Alternatif: **React + Tailwind** (daha profesyonel görünüm)

DevOps & Tools

- **Docker**: Container'lar için
- **GitHub Actions**: CI/CD
- **pytest**: Test framework
- **Pre-commit hooks**: Code quality

Hosting (Ücretsiz)

- **Hugging Face Spaces**: Demo hosting
- **GitHub Pages**: Dokümantasyon
- **Cloudflare Pages**: Eğer React kullanırsanız

Ekip Dağılımı ve Görevler

Team 1: PDF Processing & Text Extraction (2 kişi)

Kişi 1A: PDF Parser Lead

Görevler:

- [ ] PDF okuma modülü (PyMuPDF entegrasyonu)
- [ ] Metin çıkarma ve temizleme
- [ ] Tablo ve şekil tespiti
- [ ] Metadata extraction (başlık, yazar, yıl, abstract)
- [ ] Referans listesi parsing
- [ ] Hata yönetimi (bozuk PDF'ler, şifreli dosyalar)

Deliverables:

- `pdf_parser.py` modülü
- Unit testler
- 10+ farklı formattaki PDF'le test edilmiş sistem

Teknik Detaylar:

python

```python
# Beklenen modül yapısı
class PDFParser:
    def extract_text(self, pdf_path: str) -> str
    def extract_metadata(self, pdf_path: str) -> dict
    def extract_sections(self, pdf_path: str) -> dict
    def extract_citations(self, pdf_path: str) -> list
```

Kişi 1B: Text Processing Specialist

Görevler:

- [ ] Text cleaning ve preprocessing
- [ ] Section detection (Abstract, Introduction, Methods, etc.)
- [ ] Citation normalization
- [ ] LaTeX formül handling
- [ ] Text chunking stratejisi (RAG için optimum boyut)
- [ ] Multi-language detection (Türkçe/İngilizce)

Deliverables:

- `text_processor.py` modülü
- Chunking algoritması
- Section classifier
- Unit testler

Teknik Detaylar:

python

```
class TextProcessor:
    def clean_text(self, raw_text: str) -> str
    def detect_sections(self, text: str) -> dict
    def chunk_text(self, text: str, chunk_size: int) -> list
    def normalize_citations(self, citations: list) -> list
```

Team 2: LLM Integration & Prompt Engineering (2 kişi)

Kişi 2A: LLM Integration Lead

Görevler:

- [ ] Ollama kurulumu ve model yönetimi
- [ ] LangChain entegrasyonu
- [ ] Model selection ve karşılaştırma (Llama vs Mistral)
- [ ] Prompt template'leri oluşturma
- [ ] Response parsing ve formatting
- [ ] Hız ve kalite optimizasyonu
- [ ] Fallback stratejileri (model erişilemezse)

Deliverables:

- `llm_manager.py` modülü
- Model benchmark sonuçları
- Prompt library
- Docker compose file (Ollama için)

Teknik Detaylar:

python

```
class LLMManager:
    def initialize_model(self, model_name: str)
    def generate_summary(self, text: str, style: str) -> str
    def answer_question(self, question: str, context: str) -> str
    def extract_keywords(self, text: str) -> list
```

Kişi 2B: Prompt Engineer

Görevler:

- [ ] Özet çıkarma prompt'ları (farklı stiller: kısa, detaylı, bullet point)
- [ ] Soru-cevap prompt'ları

- [ ] Anahtar kelime extraction prompt'ları
- [ ] Citation analysis prompt'ları
- [ ] Few-shot örnekleri hazırlama
- [ ] Prompt versiyonlama ve A/B testing
- [ ] Türkçe/İngilizce prompt optimize

Deliverables:

- `prompts.py` - Tüm prompt template'leri
- Prompt testing sonuçları
- Prompt versiyonlama sistemi
- Best practices dokümanı

Prompt Örnekleri:


python

```
SUMMARY_PROMPTS = {
    "short": "Summarize this academic paper in 3-5 sentences...",
    "detailed": "Provide a comprehensive summary including...",
    "bullet": "Create a bullet-point summary covering..."
}
```

Team 3: Vector DB & RAG System (2 kişi)

Kişi 3A: Vector Database Architect

Görevler:

- [ ] ChromaDB setup ve konfigürasyon
- [ ] Embedding model seçimi ve entegrasyonu
- [ ] Collection yönetimi (her paper bir collection)
- [ ] Metadata indexing
- [ ] Vector search optimizasyonu
- [ ] Persistence stratejisi
- [ ] Bulk insert operations

Deliverables:

- `vector_store.py` modülü
- Database schema
- Index stratejisi dokümanı
- Performance benchmark

Teknik Detaylar:

python

```python
class VectorStore:
    def create_collection(self, paper_id: str)
    def add_chunks(self, paper_id: str, chunks: list, metadata: dict)
    def similarity_search(self, query: str, top_k: int) -> list
    def delete_collection(self, paper_id: str)
```

Kişi 3B: RAG System Developer

Görevler:

- [ ] RAG pipeline oluşturma
- [ ] Query expansion ve rewriting
- [ ] Context retrieval stratejisi
- [ ] Re-ranking algoritması
- [ ] Context window optimization
- [ ] Multi-query RAG
- [ ] Response citation (kaynak gösterme)

Deliverables:

- `rag_engine.py` modülü
- RAG pipeline
- Evaluation metrics
- Context relevance scorer

Teknik Detaylar:

python

```python
class RAGEngine:
    def ask_question(self, paper_id: str, question: str) -> dict
    def retrieve_context(self, query: str, top_k: int) -> list
    def rerank_results(self, query: str, results: list) -> list
    def generate_answer_with_citation(self, question: str, context: list) -> dict
```

Team 4: Frontend Developer (1 kişi)

Kişi 4: Full-Stack Frontend Lead

Görevler:

- [ ] UI/UX tasarımı (Figma wireframe)
- [ ] Streamlit/Gradio app geliştirme

- [ ] PDF upload interface
- [ ] Progress indicators
- [ ] Results visualization (özet, keywords, Q&A)
- [ ] Chat interface (soru-cevap için)
- [ ] Export fonksiyonları (PDF, Markdown)
- [ ] Responsive design
- [ ] Error handling ve user feedback

Deliverables:

- `app.py` - Ana uygulama
- UI mockup'lar
- User guide
- Demo video

Özellikler:

Sayfalar:
1. Ana sayfa: PDF upload
2. Paper Dashboard: Özet, metadata, keywords
3. Chat Interface: Soru-cevap
4. Library: Yüklenen tüm paperlar
5. Settings: Model seçimi, dil seçimi

Streamlit Örnek Yapı:

python

```python
# Tabs
tab1, tab2, tab3 = st.tabs(["Upload", "Summary", "Ask Questions"])

with tab1:
    uploaded_file = st.file_uploader("Choose PDF")

with tab2:
    st.markdown(summary)
    st.write("Keywords:", keywords)

with tab3:
    question = st.text_input("Ask a question")
    if st.button("Ask"):
        answer = get_answer(question)
```

Team 5: Test & Dataset Engineer (1 kişi)

Kişi 5: QA & Data Engineer

Görevler:

- [ ] Test dataset hazırlama (50+ farklı paper, farklı alanlardan)
- [ ] Unit test yazma (tüm modüller için)
- [ ] Integration test
- [ ] End-to-end test scenarios
- [ ] Performance testing (hız, memory kullanımı)
- [ ] Ground truth dataset (özet, keywords için)
- [ ] Evaluation metrics (ROUGE, BLEU, BERTScore)
- [ ] Automated testing pipeline
- [ ] Bug tracking ve raporlama

Deliverables:

- `tests/` klasörü (pytest testleri)
- Test dataset (papers + ground truth)
- Test coverage raporu (%80+)
- Performance benchmark raporu
- Bug tracking dokümanı

Test Kategorileri:


python

```
# Unit Tests
tests/test_pdf_parser.py
tests/test_text_processor.py
tests/test_llm_manager.py
tests/test_vector_store.py
tests/test_rag_engine.py

# Integration Tests
tests/test_full_pipeline.py
tests/test_api_endpoints.py

# Performance Tests
tests/test_performance.py
```

Proje Yöneticisi + DevOps (1 kişi - ekstra veya roller paylaşılabilir)

Proje Yöneticisi

Görevler:

- [ ] Sprint planning (2 haftalık sprint'ler)
- [ ] Daily standup notları
- [ ] Task tracking (GitHub Projects)
- [ ] Blocker'ları çözme
- [ ] Demo hazırlama
- [ ] README.md yazma
- [ ] Dokümantasyon koordinasyonu
- [ ] GitHub repo yönetimi
- [ ] Code review koordinasyonu

Deliverables:

- Sprint planları
- README.md
- CONTRIBUTING.md
- Architecture diagram
- API dokümantasyonu
- Demo script

DevOps Engineer (aynı kişi veya farklı)

Görevler:

- [ ] Docker containerization
- [ ] Docker Compose setup
- [ ] GitHub Actions CI/CD
- [ ] Pre-commit hooks

- [ ] Linting ve formatting (Black, Flake8)
- [ ] Dependency management (Poetry veya requirements.txt)
- [ ] Hugging Face Spaces deployment
- [ ] Environment management (.env)

Deliverables:

- `Dockerfile`
- `docker-compose.yml`
- `.github/workflows/ci.yml`
- `.pre-commit-config.yaml`
- Deployment guide

Proje Mimarisi

```
papermate/
├── src/
│   ├── pdf_processing/
│   │   ├── pdf_parser.py
│   │   └── text_processor.py
│   ├── llm/
│   │   ├── llm_manager.py
│   │   └── prompts.py
│   ├── rag/
│   │   ├── vector_store.py
│   │   └── rag_engine.py
│   ├── api/
│   │   └── main.py (FastAPI)
│   └── utils/
│       └── helpers.py
├── frontend/
│   └── app.py (Streamlit)
├── tests/
│   ├── unit/
│   ├── integration/
│   └── data/
├── notebooks/
│   └── experiments.ipynb
├── docs/
│   ├── architecture.md
│   ├── api_reference.md
│   └── user_guide.md
├── scripts/
│   └── setup.sh
├── docker-compose.yml
├── Dockerfile
├── requirements.txt
├── .env.example
└── README.md
```

Sprint Planı (8 Hafta)

Sprint 1 (Hafta 1-2): Setup & Foundation

Hafta 1:

- Herkes için: Environment setup (Python, Git, Ollama)
- Team 1: PDF parsing research, PyMuPDF test
- Team 2: Ollama kurulum, ilk model çalıştırma
- Team 3: ChromaDB setup, embedding modeli test
- Team 4: Streamlit hello world, UI mockup
- Team 5: Test framework setup, ilk dataset toplama
- PM: Repo oluşturma, proje yapısı

Hafta 2:

- Team 1: İlk PDF parser versiyonu, 5 paper test
- Team 2: İlk LangChain entegrasyonu, basit özet
- Team 3: İlk vector collection, basit search
- Team 4: PDF upload sayfası
- Team 5: İlk unit testler
- PM: İlk sprint review, GitHub Projects setup

Sprint 1 Hedefi: PDF yükle → text çıkar → basit özet

Sprint 2 (Hafta 3-4): Core Features

Hafta 3:

- Team 1: Section detection, metadata extraction
- Team 2: Farklı özet stilleri, prompt optimization
- Team 3: RAG pipeline v1, soru-cevap
- Team 4: Summary display sayfası
- Team 5: Integration testler, 20+ paper dataset
- PM: API dokümantasyonu başlangıcı

Hafta 4:

- Team 1: Citation parsing, error handling
- Team 2: Keyword extraction, multi-language
- Team 3: Context retrieval optimization
- Team 4: Chat interface
- Team 5: Performance testing başlangıcı
- PM: Architecture diagram

Sprint 2 Hedefi: Özet + Keywords + Basit Q&A çalışıyor

Sprint 3 (Hafta 5-6): Advanced Features

Hafta 5:

- Team 1: Tablo/şekil extraction
- Team 2: Few-shot prompting, quality improvement
- Team 3: Multi-query RAG, re-ranking
- Team 4: Paper library page, metadata display
- Team 5: Evaluation metrics (ROUGE scores)
- PM: User guide yazımı

Hafta 6:

- Team 1: LaTeX formül handling
- Team 2: Response citation ekle
- Team 3: Similarity search optimization
- Team 4: Export fonksiyonları (PDF, Markdown)
- Team 5: End-to-end test scenarios
- PM: API reference tamamlama

Sprint 3 Hedefi: Tüm temel özellikler tamamlandı

Sprint 4 (Hafta 7-8): Polish & Launch

Hafta 7:

- Tüm ekip: Bug fixing
- Team 1-3: Performance optimization
- Team 4: UI/UX iyileştirme, responsive design
- Team 5: Kapsamlı testing, 50+ paper
- PM: Demo hazırlama, README polish

Hafta 8:

- Herkes: Code review
- DevOps: Dockerization, CI/CD
- Team 4: Final UI touches
- Team 5: Final testing, test coverage %80+
- PM: Hugging Face Spaces deployment
- Herkes: Demo çekimi, blog post yazımı

Sprint 4 Hedefi: Production-ready, deploy edildi

Minimum Viable Product (MVP) - 4 Hafta

Eğer hızlı ilerlersek ilk 4 haftada MVP:

- ✅ PDF upload
- ✅ Text extraction
- ✅ Basit özet (bir stil)
- ✅ Keyword extraction
- ✅ Soru-cevap (RAG ile)
- ✅ Basit UI (Streamlit)

Teknik Gereksinimler

Her Ekip Üyesi İçin:


bash

```
# Python 3.10+
python --version

# Git
git --version

# VS Code veya PyCharm

# Ollama (local LLM için)
curl https://ollama.ai/install.sh | sh
ollama pull llama3.1:8b

# Dependencies
pip install -r requirements.txt
```

requirements.txt

```
langchain==0.1.0
chromadb==0.4.22
ollama-python==0.1.7
pymupdf==1.23.8
sentence-transformers==2.2.2
fastapi==0.109.0
streamlit==1.30.0
pytest==7.4.4
python-dotenv==1.0.0
pydantic==2.5.3
```

Başarı Metrikleri

Teknik Metrikler:

- [ ] Test coverage %80+
- [ ] PDF parsing success rate %95+
- [ ] Summary generation < 10 saniye
- [ ] Q&A response < 5 saniye
- [ ] Memory usage < 4GB

Kullanıcı Metrikleri:

- [ ] 10+ farklı kullanıcı test etti
- [ ] 50+ paper başarıyla işlendi
- [ ] User feedback toplanır (form)

Proje Metrikleri:

- [ ] GitHub stars hedefi: 100+
- [ ] README kalitesi: kapsamlı
- [ ] Dokümantasyon tamamlığı
- [ ] Demo video hazır

İletişim ve İşbirliği

Günlük:

- **Daily Standup:** Her gün 15 dakika (Discord/Teams)
  - Dün ne yaptım?
  - Bugün ne yapacağım?
  - Blocker var mı?

Haftalık:

- **Sprint Planning:** Pazartesi (1 saat)
- **Sprint Review:** Cuma (30 dakika)
- **Code Review:** Sürekli (PR'lar)

Araçlar:

- **Discord/Slack:** Anlık iletişim
- **GitHub Projects:** Task tracking
- **Google Docs:** Dokümantasyon
- **Figma:** UI tasarım (gerekirse)

Bonus Özellikler (Zamanınız Varsa)

- [ ] Birden fazla paper karşılaştırma
- [ ] Otomatik literatür review oluşturma
- [ ] Citation network graph'i
- [ ] Paper recommendation sistemi
- [ ] Collaborative notes (takım notları)
- [ ] Mobile-friendly UI
- [ ] API versiyonu (REST API)
- [ ] Browser extension
- [ ] Batch processing

Kaynaklar

Öğrenme Kaynakları:

- **LangChain Docs:** https://python.langchain.com/docs/
- **ChromaDB Docs:** https://docs.trychroma.com/
- **Ollama Docs:** https://github.com/ollama/ollama
- **RAG Tutorial:** https://www.pinecone.io/learn/retrieval-augmented-generation/

Dataset Kaynakları:

- **arXiv:** https://arxiv.org/ (API'si var)
- **Semantic Scholar:** https://www.semanticscholar.org/
- **PubMed:** https://pubmed.ncbi.nlm.nih.gov/

Örnek Projeler:

- **Paper QA:** https://github.com/whitead/paper-qa
- **LangChain RAG Tutorial:** https://github.com/langchain-ai/rag-from-scratch

---

İlk Adımlar (Haftaya Başlarken)

1. **Herkes:**
   - ☐ Bu dokümanı okuyun
   - ☐ GitHub hesabı oluşturun
   - ☐ Python 3.10+ kurun
   - ☐ Ollama kurun ve test edin
2. **Proje Yöneticisi:**
   - ☐ GitHub repo oluştur (public)
   - ☐ Team'leri ekle
   - ☐ Discord/Slack kanalı oluştur
   - ☐ İlk toplantı planla
3. **Her Team:**
   - ☐ İlk branch'i oluşturun
   - ☐ Modül iskeletini yazın
   - ☐ İlk test'i yazın
   - ☐ İlk PR'ı açın

Notlar

- Her commit anlamlı olmalı (conventional commits)
- Her PR'da code review zorunlu (en az 1 kişi)
- Kod yazarken Türkçe yorum yazabilirsiniz ama değişken/fonksiyon isimleri İngilizce
- Herkes birbirinin kodunu anlayabilmeli (readability önemli)
- Blocker olunca hemen ekibe bildirin
- Her sprint sonunda working demo olmalı

Son Söz: Bu büyük bir proje ama 8 kişiyle tamamen yapılabilir. Önemli olan düzenli iletişim, net görev dağılımı ve sürekli ilerleme. Her gün küçük adımlar atın, 8 hafta sonunda harika bir ürün olacak! 🚀