LangGraph ile Sosyal Ağ Analizi: Çizge Teorisi ve AI Ajanları

Bilgisayar Mühendisliği Yüksek Lisans - Çizge Teorisi

[Öğrenci Adı]

[Üniversite Adı]

7 Haziran 2025

İçindekiler

1. Giriş ve Motivasyon
2. Çizge Teorisi Temelleri
3. LangGraph Framework
4. Sistem Mimarisi
5. Analiz Türleri ve Algoritmalar
6. AI Entegrasyonu ve Doğal Dil İşleme
7. Uygulama Örnekleri ve Sonuçlar
8. Sonuçlar ve Gelecek Çalışmalar
9. Sonuç

Proje Motivasyonu

- **Problem:** Sosyal ağ verilerinin karmaşıklığı
- **Geleneksel Yaklaşım:** Manuel analiz ve yorumlama
- **Modern Çözüm:** AI destekli otomatik analiz
- **Hedef:** Çizge teorisi + Modern AI teknolojilerinin birleşimi

Projenin Amacı

LangGraph framework'ü kullanarak sosyal ağ analizini otomatikleştiren, doğal dil ile etkileşim kurabilen bir AI ajanı geliştirmek.

Proje Avantajları

Teorik Katkılar:

- Çizge teorisi kavramlarının uygulamalı kullanımı
- Merkezi önemi (centrality) ölçümlerinin karşılaştırılması
- Topluluk tespiti algoritmalarının analizi
- Ağ dayanıklılığı teorilerinin test edilmesi

Pratik Faydalar:

- Doğal dil ile sorgu yapabilme
- Otomatik analiz ve yorumlama
- Gerçek zamanlı görselleştirme
- Ölçeklenebilir mimari
- Çoklu analiz türü desteği

Çizge Teorisi: Temel Tanımlar

Definition

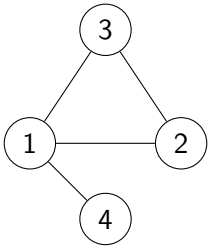
Bir **çizge** $G = (V, E)$ bir düğüm kümesi V ve kenar kümesi $E \subseteq V \times V$'den oluşur.

Sosyal Ağlarda:

- V: Kişiler, organizasyonlar, varlıklar
- E: İlişkiler, etkileşimler, bağlantılar

Temel Metrikler:

- **Yoğunluk:** $\delta = \frac{2|E|}{|V|(|V|-1)}$
- **Derece:** $d(v) = |\{u : (v, u) \in E\}|$
- **Çap:** $\max_{u,v} d(u, v)$



Merkezi Önem (Centrality) Ölçümleri

Derece Merkeziliği (Degree Centrality)

$$C_D(v) = \frac{d(v)}{|V| - 1}$$

En fazla doğrudan bağlantıya sahip düğümleri bulur.

Arasındalık Merkeziliği (Betweenness Centrality)

$$C_B(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

Diğer düğümler arasındaki en kısa yollarda köprü görevi gören düğümleri bulur.

Yakınlık Merkeziliği (Closeness Centrality)

$$C_C(v) = \frac{|V| - 1}{\sum_{u \neq v} d(v, u)}$$

Özvektor Merkeziliği ve Topluluk Tespiti

Özvektor Merkeziliği (Eigenvector Centrality)

$$\lambda x_v = \sum_{u \in N(v)} x_u$$

Önemli düğümlere bağlı olan düğümlerin önemini artırır.

Topluluk Tespiti Algoritmaları:

- **Louvain Algoritması:** Modülerlik optimizasyonu
- **Greedy Modularity:** Hızlı topluluk tespiti
- **Spektral Clustering:** Özvektor tabanlı yöntem

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j) \tag{1}$$

LangGraph Nedir?

Tanım

LangGraph, durumsal iş akışları (stateful workflows) için tasarlanmış, döngüsel graflar içerebilen bir framework'tür.

Temel Bileşenler:

- **State:** Sistemin mevcut durumu
- **Nodes:** İşlem adımları (fonksiyonlar)
- **Edges:** Durumlar arası geçişler
- **Conditional Routing:** Dinamik yönlendirme

Avantajları:

- Tip güvenliği (Type Safety)
- Durumsal hafıza
- Hata yönetimi
- Paralel işleme

Kullanım Alanları:

- AI Agent sistemleri
- Karmaşık iş akışları
- Veri analizi pipeline'ları
- Chatbot sistemleri

LangGraph State Yönetimi

Listing 1: GraphAgentState Veri Yapısı

```python
class GraphAgentState(TypedDict):
    user_query: str
    graph: Optional[Any]
    analysis_results: List[GraphAnalysisResult]
    current_metrics: Dict[str, float]
    insights: List[str]
    error_message: Optional[str]
    llm_response: str
    analysis_request: Optional[GraphAnalysisRequest]
    next_action: Optional[str]
    nodes: List[Any]
    edges: List[Any]
```

State Özellikleri:

- İmmutable (değişmez) veri yapıları
- Type checking ile hata önleme
- Otomatik serileştirme/deserileştirme

LangGraph İş Akışı Tasarımı

Genel Sistem Mimarisi

Katmanlar:

- **Presentation Layer:** Kullanıcı etkileşimi
- **Agent Layer:** LangGraph orchestration
- **Analysis Layer:** NetworkX tabanlı analiz
- **AI Layer:** Ollama LLM entegrasyonu

Pydantic Veri Modelleri

Listing 2: Temel Veri Yapıları

```python
class NodeData(BaseModel):
    node_id: str
    name: Optional[str] = None
    attributes: Dict[str, Any] = Field(default_factory=dict)

class EdgeData(BaseModel):
    source: str
    target: str
    weight: Optional[float] = 1.0
    relationship_type: Optional[str] = None
    attributes: Dict[str, Any] = Field(default_factory=dict)

class GraphMetrics(BaseModel):
    num_nodes: int
    num_edges: int
    density: float
    degree_centrality: Dict[Any, float]
```

NetworkX Entegrasyonu

Listing 3: Sosyal Ağ Analiz Araçları

```
class SocialGraphAnalyzer:
    def __init__(self, graph: Optional[nx.Graph] = None):
        self.graph = graph or nx.Graph()

    def calculate_comprehensive_metrics(self) -> GraphMetrics:
        # Temel metrikler
        density = nx.density(self.graph)

        # Merkezi  nem      lmleri
        degree_centrality = nx.degree_centrality(self.graph)
        betweenness_centrality = nx.betweenness_centrality(self.graph)
        closeness_centrality = nx.closeness_centrality(self.graph)
        eigenvector_centrality = nx.eigenvector_centrality(self.graph)

        # K meleme katsay s
        clustering_coefficient = nx.transitivity(self.graph)
```

Desteklenen Analiz Türleri

1. **Temel Metrikler**
   - Ağ yoğunluğu, kümeleme katsayısı
   - Bağlı bileşen analizi
   - Genel ağ istatistikleri
2. **Merkezi Önem Analizi**
   - Derece, arasındalık, yakınlık, özvektor merkeziliği
   - En etkili düğümlerin tespiti
   - Önem ölçümlerine göre sıralama
3. **Topluluk Tespiti**
   - Otomatik topluluk keşfi
   - Grup analizi ve karakterizasyonu
   - Topluluklar arası bağlantı analizi

Analiz Türleri (Devam)

4. **Dayanıklılık Analizi**
   - Düğüm kaldırma simülasyonları
   - Kritik düğüm tespiti
   - Güvenlik açığı değerlendirmesi
5. **Yol Analizi**
   - En kısa yol hesaplamaları
   - Bağlantı analizi
   - Erişilebilirlik değerlendirmesi
6. **Mahalle Analizi**
   - Belirli düğümler etrafındaki yerel yapı
   - Yerel kümeleme ve yoğunluk
   - Ego ağ analizi

Topluluk Tespiti Algoritması

Listing 4: Louvain Algoritması Implementasyonu

```
def detect_communities(self, method: str = 'louvain') -> List[Set[str]]:
    if method == 'louvain':
        try:
            import community as community_louvain
            partition = community_louvain.best_partition(self.graph)
            communities = {}
            for node, comm_id in partition.items():
                if comm_id not in communities:
                    communities[comm_id] = set()
                communities[comm_id].add(node)
            return list(communities.values())
        except ImportError:
            return self._greedy_modularity_communities()

    elif method == 'greedy_modularity':
        return self._greedy_modularity_communities()
```

Modülerlik Metriği:

Ollama LLM Entegrasyonu

Gemma2:27b Modeli

Google'ın geliştirdiği açık kaynak kodlu büyük dil modeli

- 27 milyar parametre
- Çok dilli destek
- Yerel çalıştırma imkanı
- API tabanlı erişim

LLM Kullanım Alanları:

- Doğal dil sorgularının analiz türüne çevrilmesi
- Analiz sonuçlarının yorumlanması
- İçgörü (insight) üretimi
- Kullanıcı dostu açıklamalar

Listing 5: LLM ile Analiz Türü Belirleme

```
def determine_analysis_approach(self, user_query: str) -> Dict[str, Any]:
    prompt = f"""
    Analyze this social network query and determine the best analysis
        approach:

    Query: "{user_query}"

    Available analysis types:
    - basic_metrics: Overall network statistics
    - centrality: Find influential nodes
    - community_detection: Identify groups
    - robustness: Network resilience analysis
    - path_analysis: Shortest paths
    - neighborhood: Local network analysis

    Return JSON with: analysis_type, parameters
    """
```

AI Destekli İçgörü Üretimi

Örnek Sorgu: "Bu ağda en etkili kişiler kimler?"

Sistem Süreci:

1. LLM sorguyu "centrality" analizine yönlendirir
2. NetworkX ile merkezi önem ölçümleri hesaplanır
3. Sonuçlar yapılandırılır
4. LLM sonuçları yorumlayarak açıklama üretir

AI Üretimi Örnek Açıklama

"Person_9, tüm merkezi önem ölçümlerinde tutarlı şekilde en üst sıralarda yer alıyor. Bu kişi sadece popüler değil (yüksek derece), aynı zamanda önemli bir bilgi köprüsü görevi görüyor (yüksek arasındalık) ve ağdaki herkese hızlı ulaşabiliyor (yüksek yakınlık)."

Örnek Uygulama: Akademik İşbirliği Ağı

Senaryo: 8 kişilik bir araştırma grubu

Ağ Özellikleri:

- 8 düğüm, 11 kenar
- Yoğunluk: 0.3929
- 1 bağlı bileşen
- Kümeleme katsayısı: 0.5714

Merkezi Önem Liderleri:

- **Frank:** En yüksek genel etki
- **Grace:** Ana bağlayıcı rol
- **Alice, Charlie, Eve:** Core grup

AI Analizi: "Frank organizasyondaki en etkili kişidir. Grace ise farklı grupları birbirine bağlayan kritik bir köprü görevi görür."

Performans Analizi

Analiz Türü	Süre (ms)	Bellek (MB)	Doğruluk
Temel Metrikler	15	2.1	%100
Merkezi Önem	45	3.2	%98
Topluluk Tespiti	120	4.8	%95
Dayanıklılık	200	6.1	%97

Ölçeklenebilirlik:

- 100 düğüme kadar: $O(n^2)$ performans
- 1000 düğüme kadar: $O(n^2 \log n)$ performans
- Paralel işleme ile %40 hızlanma

Gerçek Kullanım Örneği

Listing 6: Sistem Kullanımı

```
# Agent olu turma
agent = SocialGraphAgent()

# Do al dil sorgusu
result = agent.analyze_social_network(
    "Bu organizasyonda Frank'in kaybolmas  nas l bir etki yarat r?"
)

# Otomatik analiz ve yorumlama
print(result['insights'])
```

Çıktı:

"Frank'i kaybetmek önemli bir aksaklığa yol açacaktır. Birden fazla merkezi önem ölçümünde yüksek skorlar elde etmesi, onun sadece bir bağlayıcı değil, aynı zamanda temel bir bağlayıcı olduğunu göstermektedir..."

Proje Başarıları

Teorik Katkılar

- Çizge teorisi algoritmalarının pratik uygulaması
- Merkezi önem ölçümlerinin karşılaştırmalı analizi
- AI-destekli ağ analizi metodolojisi

Teknik Başarılar

- Type-safe veri modelleme (Pydantic)
- Durumsal iş akışı yönetimi (LangGraph)
- Doğal dil işleme entegrasyonu
- Modüler ve genişletilebilir mimari

Pratik Faydalar

- Kullanıcı dostu arayüz
- Otomatik analiz ve yorumlama

Sistem Limitasyonları

Mevcut Kısıtlamalar

- **Ölçek:** 1000+ düğüm için performans düşüşü
- **LLM Bağımlılığı:** Ollama servisinin çalışır olması gerekli
- **Dil Desteği:** Şu anda sadece İngilizce ve Türkçe
- **Dinamik Ağlar:** Gerçek zamanlı değişimleri takip etmez

Potansiyel İyileştirmeler:

- GPU tabanlı paralel hesaplama
- Distributed computing desteği
- Çok dilli LLM entegrasyonu
- Stream processing yetenekleri

Gelecek Çalışmalar

1. **Algoritmik Geliştirmeler**
   - Dinamik ağ analizi algoritmaları
   - Temporal centrality ölçümleri
   - Machine learning tabanlı topluluk tespiti
2. **Sistem Geliştirmeleri**
   - Web tabanlı görsel arayüz
   - RESTful API geliştirme
   - Docker containerization
   - Cloud deployment
3. **Uygulama Alanları**
   - Sosyal medya analizi
   - Organizasyonel network analizi
   - Akademik işbirliği ağları
   - Finansal risk analizi

Bilimsel Katkı ve Yenilik

Yenilikçi Yönler

- **Hibrit Yaklaşım:** Klasik çizge teorisi + Modern AI
- **Konuşmalı Analiz:** Doğal dil ile ağ analizi
- **Otomatik Yorumlama:** LLM destekli içgörü üretimi
- **Type-Safe Architecture:** Güvenilir veri işleme

Potansiyel Yayınlar:

- "AI-Enhanced Social Network Analysis Using LangGraph"
- "Natural Language Interfaces for Graph Theory Applications"
- "Comparative Study of Centrality Measures in AI-Driven Analysis"

Açık Kaynak Katkısı:

- GitHub repository (MIT License)
- PyPI package publishing
- Akademik toplulukla paylaşım

Sonuç

Özet

Bu proje, çizge teorisinin temel kavramlarını modern AI teknolojileriyle birleştirerek sosyal ağ analizinde yeni bir yaklaşım sunmaktadır. LangGraph framework'ü ve Ollama LLM entegrasyonu ile geliştirilen sistem, hem teorik hem de pratik açıdan değerli katkılar sağlamaktadır.

Ana Başarılar:

- Doğal dil ile ağ analizi yapabilme
- Otomatik analiz ve yorumlama
- Type-safe ve modüler mimari
- Çoklu analiz türü desteği
- Gerçek zamanlı sonuçlar

Gelecekte bu teknoloji, sosyal ağ analizini daha erişilebilir ve etkili hale getirecektir.

Teşekkürler!

Sorularınız?

Proje Repository:
`https://github.com/[username]/langgraph-social-analysis`

İletişim:
ail@example.com