AKADEMİK İŞBİRLİĞİ HESAPLAMA

Özet

Bu proje, akademik işbirliği ağlarının analizine ve görselleştirilmesine odaklanmaktadır. Excel dosyalarından alınan veriler işlenerek yazarların işbirlikleri bir ağ grafiği üzerinde görselleştirilmiştir. Flask web framework'ü kullanılarak kullanıcıların Excel dosyalarını yükleyebildiği, analiz sonuçlarını görüntüleyebildiği ve çeşitli analizleri gerçekleştirebildiği bir web tabanlı arayüz oluşturulmuştur. Proje, kısayolların hesaplanması, en işbirlikçi yazarların belirlenmesi gibi analiz özellikleri sunmaktadır.

Giriş

Akademik yayınlar ve araştırmacılar arasındaki işbirliklerinin analizi, bilimsel ilerlemenin anlaşılmasında önemli bir rol oynamaktadır. Bu proje, akademik işbirliklerini daha iyi anlamak ve analiz etmek amacıyla ağ analitiği yöntemlerini kullanarak bu işbirliklerini görselleştirme ve analiz etme imkanı sunmaktadır. Flask web framework ve PyVis kütüphanesi kullanılarak, etkileşimli bir ağ grafiği oluşturulmuş ve çeşitli analizlerin gerçekleştirilmesine olanak sağlanmıştır.

Yöntem

Bu bölümde uygulamanın geliştirilmesinde kullanılan yöntemler detaylı bir şekilde açıklanmıştır.

1) **read_excel_data**

Amaç: Bir Excel dosyasını okuyarak gerekli sütunları ayıklar.

Parametreler:
1. file_path (string): Excel dosyasının yolu.

Döndürür: Gerekli sütunları içeren bir DataFrame.

Detaylar: Excel dosyasının gerekli sütunları içerip içermediğini kontrol eder ve eksikse hata verir.

2) calculate_author_total_papers

Amaç: Her bir yazarın toplam makale sayısını hesaplar.
Parametreler:
data (DataFrame): Yazar bilgilerini içeren DataFrame.

Döndürür: Yazar isimlerini anahtar, toplam makale sayılarını değer olarak içeren bir sözlük.

Detaylar: DataFrame üzerinde dolaşarak 'coauthors' sütununu işler ve her bir yazarın kaç kez geçtiğini hesaplar.

3) calculate_node_weights

Amaç: Belirli bir yazarın işbirlikleri üzerinden düğüm ağırlıklarını hesaplar.
Parametreler:
1. data (DataFrame): Yazar bilgilerini içeren DataFrame.
2. author_id (string): İşbirlikleri hesaplanacak yazarın ID'si.

Döndürür: Ağırlıklarına göre azalan şekilde sıralanmış eş yazarlar ve makale sayıları listesi.

Detaylar: Belirtilen yazarın tüm eş yazarlarını ve bunların toplam makale sayılarını hesaplar.

4) find_most_collaborative_author

Amaç: En çok işbirliği yapan yazarı bulur.

Parametreler:

1. edges (dictionary): İşbirliklerini içeren kenar sözlüğü.

Döndürür: En çok işbirliği yapan yazarın adı ve toplam işbirliği sayısı ile bir demet.

Detaylar: Her bir yazarın işbirliklerini sayar ve en yükseğini belirler.

5) create_manual_graph

Amaç: Yazarlar ve işbirlikleri için grafik yapısını oluşturur.

Parametreler:

1. data (DataFrame): Yazar bilgilerini içeren DataFrame.

Döndürür: Düğümler, kenarlar ve ana yazarların bir demeti.

Detaylar: Yazarları ve eş yazarlarını işler, ORCID'leri eşsiz düğümler olarak tanımlar ve işbirlikleri için kenarlar ekler.

6) visualize_graph_with_output

Amaç: Grafiği pyvis kütüphanesi kullanarak görselleştirir.

Parametreler:

1. nodes (dictionary): Düğüm sözlüğü.
2. edges (dictionary): Kenar sözlüğü.
3. author_total_papers (dictionary): Yazarlar ve toplam makale sayılarını içeren sözlük.
4. highlight_path (list, optional): Vurgulanacak düğüm yolu.

Döndürür: Bir pyvis Network nesnesi.

Detaylar: Grafik ayarlarını yapar, düğüm ve kenarları ekler, isteğe bağlı olarak yolu vurgular.

7) find_shortest_path_between_authors

Amaç: İki yazar arasındaki en kısa yolu Bulur.

Parametreler:

1. start_author (string): Başlangıç yazarı.
2. end_author (string): Bitiş yazarı.
3. nodes (dictionary): Düğüm sözlüğü.
4. edges (dictionary): Kenar sözlüğü.

Döndürür: En kısa yolu temsil eden bir liste.

Detaylar: BFS algoritmasını kullanarak en kısa yolu bulur.

8) calculate_shortest_paths_from_author

Amaç: Belirli bir yazardan tüm düğümlere en kısa yolları hesaplar.

Parametreler:
1. author_id (string): Yazarın ID'si.
2. nodes (dictionary): Düğüm sözlüğü.
3. edges (dictionary): Kenar sözlüğü.

Döndürür: Mesafeler ve önceki düğümleri içeren iki sözlük.
Detaylar: Dijkstra algoritmasını uygular.

9) TreeNode ve BinarySearchTree Sınıfları

Amaç: Yazarları yönetmek için bir ikili arama ağacı (BST) yapısı sağlar.

TreeNode Özellikleri:
- **author_name**: Yazarın adı.
- **left**: Sol çocuk düğüm.
- **right**: Sağ çocuk düğüm.

BinarySearchTree Metotları:

- **insert**: Bir yazarı BST'ye ekler.
- **delete**: BST'den bir yazarı siler.
- **_insert:** Ekleme için yardımcı metot.
- **_delete**: Silme için yardımcı metot.
- **_min_value_node**: Minimum değere sahip düğümü bulur.
- **inorder_traversal**: Yazar adlarını toplar.

10) get_authors_queue

Amaç: Verilerden benzersiz yazarların bir listesini çıkarır.

Parametreler:
1. data (DataFrame): Yazar bilgilerini içeren DataFrame.

Döndürür: Benzersiz yazar isimlerini içeren bir liste.

Detaylar: DataFrame üzerinden dolaşarak eşsiz yazar isimlerini toplar.

11) create_bst_from_queue

Amaç: Yazar kuyruğundan bir ikili arama ağacı (BST) oluşturur.

Parametreler:
1. queue (list): Yazar listesini içeren kuyruk.

Döndürür: Bir BinarySearchTree nesnesi.

Detaylar: Kuyruktaki her yazarı BST'ye ekler.

12) visualize_bst

Amaç: İkili arama ağacını pyvis kullanarak görselleştirir.

Parametreler:
1. bst (BinarySearchTree): Görselleştirilecek BST.

Detaylar: Pyvis grafiğine düğümler ve kenarlar ekler, ardından görselleştirmeyi bir tarayıcıda açar.

13) **find_longest_path**

Amaç: Belirli bir düğümden başlayan en uzun yolu bulur.

Parametreler:
1. start_node (string): Başlangıç düğümünün ID'si.
2. visited (set, optional): Ziyaret edilen düğümler.
3. current_path (list, optional): Şu anki yol.

Döndürür: En uzun yolu temsil eden bir liste.

Detaylar: Rekürsif olarak yolları keşfeder ve en uzun olanını bulur.

Flask Yönlendirmeleri

- ***/calculate_node_weights/<author_id>***

  Amaç: Belirli bir yazar için düğüm ağırlıklarını hesaplar.

  Metot: GET

  Parametreler:
  1. author_id (URL parametresi): Yazarın ID'si.

  Döndürür: Düğüm ağırlıklarıyla JSON çıktısı.

- ***/shortest_paths_from_author***

  Amaç: Belirli bir yazardan başlayarak en kısa yolları hesaplar.

  Metot: POST

  Parametreler:
  1. author_id (form verisi): Yazarın ID'si.

  Döndürür: En kısa yolların HTML tablosu.

- ***/shortest_path/<start_author>/<end_author>***

  Amaç: İki yazar arasındaki en kısa yolu bulur.

  Metot: GET

  Parametreler:
  1. start_author ve end_author (URL parametreleri): Yazarların ID'leri veya isimleri.

  Döndürür: En kısa yolu ve görselleştirilmiş grafiği içeren HTML yanıtı.

- ***/calculate_collaborators/<author_input>***

  Amaç: Belirli bir yazarın işbirliği yaptığı kişi sayısını hesaplar.

  Metot: GET

  Parametreler:
  1. author_input (URL parametresi): Yazarın ID'si veya adı.

  Döndürür: İşbirlikçi sayısını içeren HTML yanıtı.

- ***/create_bst***

  Amaç: Yazar kuyruğundan bir BST oluşturur ve görselleştirir.

  Metot: POST

Parametreler:

1. author_id (form verisi): Silinmesi gereken yazarın ID'si (isteğe bağlı).

Döndürür: BST görselleştirmesi ve yazar detaylarını içeren HTML yanıtı.

- ***/longest_path***

Amaç: Belirli bir yazardan başlayan en uzun yolu bulur ve görselleştirir.

Metot: POST

Parametreler:

1. author_id (form verisi): Yazarın ID'si.

Döndürür: En uzun yolu ve görselleştirilmiş grafiği içeren HTML yanıtı.

Deneysel Sonuçlar

Projede kullanılan veri seti üzerinden aşağıdaki sonuçlar elde edilmiştir:

1. **En çok işbirliği yapan yazar:** Rajeev Kumar, 344 (0000-0002-6277-2626)

2. Graf görselleştirmesiyle önemli yazarlar ve işbirlikleri görsel olarak analiz edilmiştir.

Sonuç

Bu proje, akademik işbirlikleri analizinde etkili bir araç sunmuştur. Ağ grafikleri ve algoritmalar kullanılarak, işbirliklerinin kapsamlı bir analizi gerçekleştirilmiştir. Flask tabanlı web arayüzü, kullanıcıların analiz sonuçlarına kolayca erişmesini sağlamıştır. İsterler bir menüde butonlar halinde belirtilmiş olup üzerlerine tıklandığında işlevlerini başarılı bir şekilde yerine getirmektedirler.

Kaynakça

1. PyVis Library Documentation. https://pyvis.readthedocs.io
2. Flask Framework Documentation. https://flask.palletsprojects.com/