Zayad Almazrouei

DS 210 Final Project

Introduction

The goal of this project is to analyze a large-scale user review dataset (Finefoods dataset) and uncover key patterns in user-product interactions using graph theory. By representing the dataset as a bipartite graph, where users and products are distinct node sets, we leverage degree calculations, centrality measures, and shortest path computations to derive insights. The project demonstrates how graph algorithms can model real-world networks and provide useful information about node connectivity, importance, and graph structure. This analysis is relevant for tasks such as identifying popular products, finding active reviewers, and exploring connectivity within the dataset. The Fine Foods dataset is particularly suitable for graph analysis because it inherently forms a bipartite graph structure, where users and products represent two distinct sets of nodes. Edges between these nodes indicate user reviews, making it an ideal dataset for studying connectivity, influence, and clustering in networks.

Dataset

As the Dataset was too large to show on Github, I have linked the location where you can find the dataset. https://snap.stanford.edu/data/web-FineFoods.html

Goals of the Project

The primary motivation for this project is to extract actionable insights from the Fine Foods dataset to better understand user-product relationships and their significance. By applying graph theory, I aim to achieve the following:

Identify Key Products and Users:

Why: In large datasets like Finefoods, it is essential to determine which products are the most popular and which users contribute the most reviews. This helps businesses identify their most influential products and loyal customers.

How: By calculating the degree of each node (number of connections), we can rank products based on the number of reviews they received and identify highly active users who interact with multiple products.

Measure the Influence of Nodes:

Why: Not all nodes in a graph are equally important. Some users may review a large variety of products, while others focus on specific products. Similarly, some products may appear disproportionately popular.

How: I will use degree centrality to measure the relative influence of nodes. Degree centrality normalizes node degrees, allowing us to identify products and users that play a central role in the dataset.

Understand the Connectivity of the Network:

Why: By analyzing the structure of the graph, I can determine how closely related users and products are. A low average shortest path distance between nodes implies that users are reviewing similar or related products.

How: I calculate the average shortest path distance using Breadth-First Search (BFS) on a sampled subset of nodes. This provides insights into the graph's overall connectivity and cohesion.

Uncover Hidden Patterns in User-Product Interactions:

Why: Understanding how users interact with products can inform strategies such as product bundling, targeted marketing, and personalized recommendations. Identifying patterns such as clusters of highly connected nodes can reveal user behavior trends and product affinities.

How: By analyzing node degrees, centrality values, and connectivity metrics, we identify trends in user activity and product popularity. These insights can help businesses optimize their offerings and target key user segments effectively.

Demonstrate Real-World Applications of Graph Theory:

Why: This project is not just an academic exercise; it highlights how graph theory can be used to solve real-world problems in e-commerce, recommendation systems, and social network analysis.

How: Through careful graph construction, analysis, and interpretation, we demonstrate how large datasets can be transformed into meaningful networks that provide valuable insights for decision-making.

By achieving these goals, we aim to show the practical utility of graph analysis in uncovering patterns, understanding user behavior, and driving data-driven strategies.

Steps and Methodology

1. Graph Construction

The dataset is parsed line by line to create a graph. Each review contributes to the graph structure as follows:

Users as Nodes: Each unique review/userId becomes a user node.

Products as Nodes: Each unique product/productId becomes a product node.

Edges as Reviews: An edge connects a user node to a product node if the user reviewed that product.

This graph is constructed using the read_graph_from_file function, which processes the dataset efficiently:

Code Explanation: Graph Construction

Line Parsing: The function reads each line from the dataset and extracts relevant fields (review/userId, product/productId).

Node Mapping: Nodes are mapped to unique identifiers using a HashMap. If a user or product node does not exist in the graph, it is added.

Edge Creation: An edge is created between the user node and the product node. Error Handling: Invalid UTF-8 lines are skipped using String::from_utf8_lossy, ensuring robustness.

```rust
pub fn read_graph_from_file(file_path: &str) -> Graph<String, ()> {
    let mut graph = Graph::<String, ()>::with_capacity(0, 0);
    let mut node_map: HashMap<String, NodeIndex> = HashMap::new();
    let mut current_user: Option<String> = None;
    let mut current_product: Option<String> = None;

    let file = File::open(file_path).expect("Failed to open dataset file");
    let reader = BufReader::new(file);

    for line in reader.split(b'\n') {
        match line {
            Ok(line) => {
                let line = String::from_utf8_lossy(&line).trim().to_string();

                // A blank line marks the end of a record
                if line.is_empty() {
                    if let (Some(user), Some(product)) = (current_user.take(), current_product.take()) {
                        let user_node = *node_map.entry(user.clone()).or_insert_with(|| graph.add_node(user));
                        let product_node = *node_map.entry(product.clone()).or_insert_with(|| graph.add_node(product));
                        graph.add_edge(user_node, product_node, ());
                    }
                    continue;
                }

                // Parse key-value pairs
                if let Some((key, value)) = line.split_once(": ") {
                    match key.trim() {
                        "review/userId" => current_user = Some(value.trim().to_string()),
                        "product/productId" => current_product = Some(value.trim().to_string()),
                        _ => {}
                    }
                }
            }
            Err(e) => {
                eprintln!("Skipping invalid line: {:?}", e);
            }
        }
    }

    graph
}
```

2. Graph Analysis

The constructed graph is analyzed to extract meaningful insights about user-product interactions. The following metrics are calculated:

Degree:

- The degree of a node represents the number of connections it has.
- For products: Degree indicates the number of reviews.
- For users: Degree indicates the number of products reviewed.

Code for Degree Calculation:

```rust
/// Calculates the degree (number of connections) for each node in the graph.
pub fn calculate_degree(graph: &Graph<String, ()>) -> HashMap<NodeIndex, f64> {
    let mut degrees = HashMap::new();
    for node in graph.node_indices() {
        degrees.insert(node, graph.neighbors(node).count() as f64);
    }
    degrees
}
```

Degree Centrality:

- What It Measures: Degree centrality normalizes the degree of a node by dividing it by the maximum possible degree in the graph.
- Purpose: This metric identifies nodes that play a central role relative to the entire network, allowing us to find influential users and products.

Code for Degree Centrality Calculation:

```rust
/// Calculates the degree centrality (normalized degree) for each node.
pub fn calculate_degree_centrality(graph: &Graph<String, ()>) -> HashMap<NodeIndex, f64> {
    let mut degree_centrality = HashMap::new();

    if graph.node_count() > 1 {
        let max_possible_degree = (graph.node_count() - 1) as f64;

        for node in graph.node_indices() {
            let degree = graph.neighbors(node).count() as f64;
            degree_centrality.insert(node, degree / max_possible_degree);
        }
    } else {
        for node in graph.node_indices() {
            degree_centrality.insert(node, 0.0); // For empty or single-node graph
        }
    }

    degree_centrality
}
```

Average Shortest Path Distance:

To compute the average shortest path distance, we sample a subset of nodes from the graph rather than analyzing every node. Sampling is necessary for large graphs to reduce computational overhead while still obtaining representative results. The sampled nodes are used as starting points for Breadth-First Search (BFS), calculating distances to all reachable nodes. These distances are averaged across all sampled nodes to approximate the graph's overall connectivity.

Code for Average Shortest Path Distance Calculation:

```rust
pub fn calculate_average_distance(graph: &Graph<String, ()>, sample_size: usize) -> f64 {
    use rand::seq::SliceRandom;
    use rand::thread_rng;

    let mut rng = thread_rng();
    let nodes: Vec<_> = graph.node_indices().collect();
    let sampled_nodes: Vec<_> = nodes.choose_multiple(&mut rng, sample_size).cloned().collect();

    let mut total_distance = 0;
    let mut total_pairs = 0;

    for &start_node in &sampled_nodes {
        let distances = bfs_distances(graph, start_node);
        for (_, distance) in distances {
            total_distance += distance;
            total_pairs += 1;
        }
    }

    if total_pairs > 0 {
        total_distance as f64 / total_pairs as f64
    } else {
        0.0
    }
```

Results

Top 10 Nodes by Degree: The nodes with the highest degree represent either highly active reviewers or highly popular products:

```
Top 10 Nodes by Most Reviewed Products / Active Users (Degree):
1. Node 480: 448.0000
2. Node 1557: 421.0000
3. Node 139: 389.0000
4. Node 1644: 365.0000
5. Node 180: 256.0000
6. Node 5453: 204.0000
7. Node 24: 201.0000
8. Node 3310: 199.0000
9. Node 596: 178.0000
10. Node 711: 176.0000
```

Top 10 Nodes by Degree Centrality and Average Distance

The relative importance of nodes based on their connectivity:

The average shortest path distance between sampled nodes in the graph is approximately 0.63, suggesting a highly connected structure.

```
Top 10 Nodes by Normalized Connectivity (Degree Centrality):
1. Node 480: 0.0014
2. Node 1557: 0.0013
3. Node 139: 0.0012
4. Node 1644: 0.0011
5. Node 180: 0.0008
6. Node 5453: 0.0006
7. Node 24: 0.0006
8. Node 3310: 0.0006
9. Node 596: 0.0005
10. Node 711: 0.0005

Average Distance Between Sampled Nodes: 0.63
```

Insights and Discussion

1. Popular Products:

Observation: Products with the highest degree are the most reviewed, which means they attract significant user attention. For instance, the product corresponding to Node 480 has the highest number of reviews (448 connections), showcasing its widespread popularity.

Implication: Such products are likely to be highly rated or have distinctive features that attract consumers. They can be leveraged in marketing campaigns as flagship items to drive sales or included in product bundles to improve visibility for less popular items.

2. Active Users:

Observation: Users with the highest degree centrality are the most active in reviewing multiple products. These users act as key influencers within the network, as their reviews potentially sway public opinion on a wide variety of products.

Implication: Engaging these active users through loyalty programs, discounts, or personalized offers can encourage further reviews and feedback. They could also be targeted for beta testing or early product launches, given their consistent activity.

3. Graph Connectivity:

Observation: The low average shortest path distance (~0.63) indicates a highly interconnected graph, meaning most nodes are only a few steps away from one another. This reflects the dense interaction between users and products in the dataset.

Implication: A tightly connected network suggests that users are reviewing a broad range of products, possibly within overlapping categories. This is a valuable insight for cross-selling strategies or recommendations, as products are likely to share a common user base.

4. Degree Distribution:

Observation: The degree distribution reveals that a small subset of nodes (products and users) dominate the interactions, while the majority have significantly fewer connections. This is a

typical feature of real-world networks, where a few items or individuals play a disproportionately large role.

Implication: For businesses, identifying these key nodes is crucial. For products, they represent best-sellers or niche items with dedicated audiences. For users, they signify power users who can be prioritized in engagement strategies.

5. Potential Product Clusters:

Observation: Nodes with similar degrees and connections are likely to represent products or users within specific categories. For instance, multiple users might frequently review products within the same genre (e.g., organic snacks or beverages).

Implication: By identifying these clusters, businesses can group similar products for targeted promotions or highlight them together in recommendation systems. This also aids in understanding user preferences within specific niches.

6. Practical Applications:

E-Commerce: Insights from this graph analysis can guide strategies like dynamic pricing for popular products, personalized recommendations for active users, and efficient inventory management.

Marketing: The analysis highlights products with high engagement potential, enabling targeted marketing campaigns that focus on trending or frequently reviewed items.

Product Development: Patterns of user behavior and product popularity can inform decisions on product enhancements or new product introductions.

In conclusion, this project demonstrates the practical application of graph theory in analyzing a real-world dataset. By transforming the Fine Foods dataset into a bipartite graph and leveraging graph metrics such as degree, degree centrality, and average shortest path distance, we successfully extracted meaningful insights about user-product interactions.

Key findings include:

- Identifying the most popular products, which can serve as focal points for marketing and bundling strategies.
- Highlighting highly active users who influence the network and can be prioritized for engagement programs.
- Unveiling a highly interconnected network structure, indicative of shared interests among users and overlapping product categories.