Project Report: Seed-Based and Seedless De-Anonymization of Graphs

1. Project Overview

This project focuses on de-anonymizing nodes between two graphs using both seed-based and seedless de-anonymization approaches. The goal is to map nodes from an anonymized graph (Graph1) to a reference graph (Graph2) by leveraging structural and feature-based similarities.

2. Data Description

Graph1 (ca-GrQc_Graph1_edgelist.txt): Contains the edge list for an anonymized graph.

Graph2 (ca-GrQc_Graph2_edgelist.txt): Reference graph used for mapping.

Node Mapping (ca-GrQc_node_mapping.txt): Contains a set of known mappings (seeds) between Graph1 and Graph2 nodes.

3. Feature Extraction

For both graphs, the following features are extracted for each node:

1. Degree: Number of connections a node has.
2. Clustering Coefficient: Measure of the likelihood that a node's neighbors are connected.
3. Average Neighbor Degree: Average degree of a node's neighbors.

```
def extract_features(G):
    features = []
    for node in G.nodes():
        degree = G.degree[node]
        clustering = nx.clustering(G, node)
        avg_neighbor_degree = nx.average_neighbor_degree(G)[node]
        features.append([node, degree, clustering, avg_neighbor_degree])
    return pd.DataFrame(features, columns=["Node", "Degree", "Clustering", "AvgNeighborDegree"])
```

4. Seed-Based De-Anonymization

Seed-based de-anonymization starts with the known mappings (seed nodes) and extends the mapping iteratively by finding the best matches for neighbors of the mapped nodes using cosine similarity of features.

```
def seed_based_mapping(seed_mapping, features_G1, features_G2, G1, G2):
    extended_mapping = seed_mapping.copy()
    for g1_seed, g2_seed in seed_mapping.items():
        g1_neighbors = set(G1.neighbors(g1_seed)) - set(seed_mapping.keys())
        g2_neighbors = set(G2.neighbors(g2_seed)) - set(seed_mapping.values())

        for g1_node in g1_neighbors:
            best_match = None
            best_score = float('-inf')
            for g2_node in g2_neighbors:
                      g1_features = features_G1[features_G1["Node"] == g1_node].iloc[:,
1:].values.flatten()
                      g2_features = features_G2[features_G2["Node"] == g2_node].iloc[:,
1:].values.flatten()
                score = cosine_similarity([g1_features], [g2_features])[0, 0]
                if score > best_score:
                    best_match = g2_node
                    best_score = score
            if best_match:
                extended_mapping[g1_node] = best_match
    return extended_mapping
```

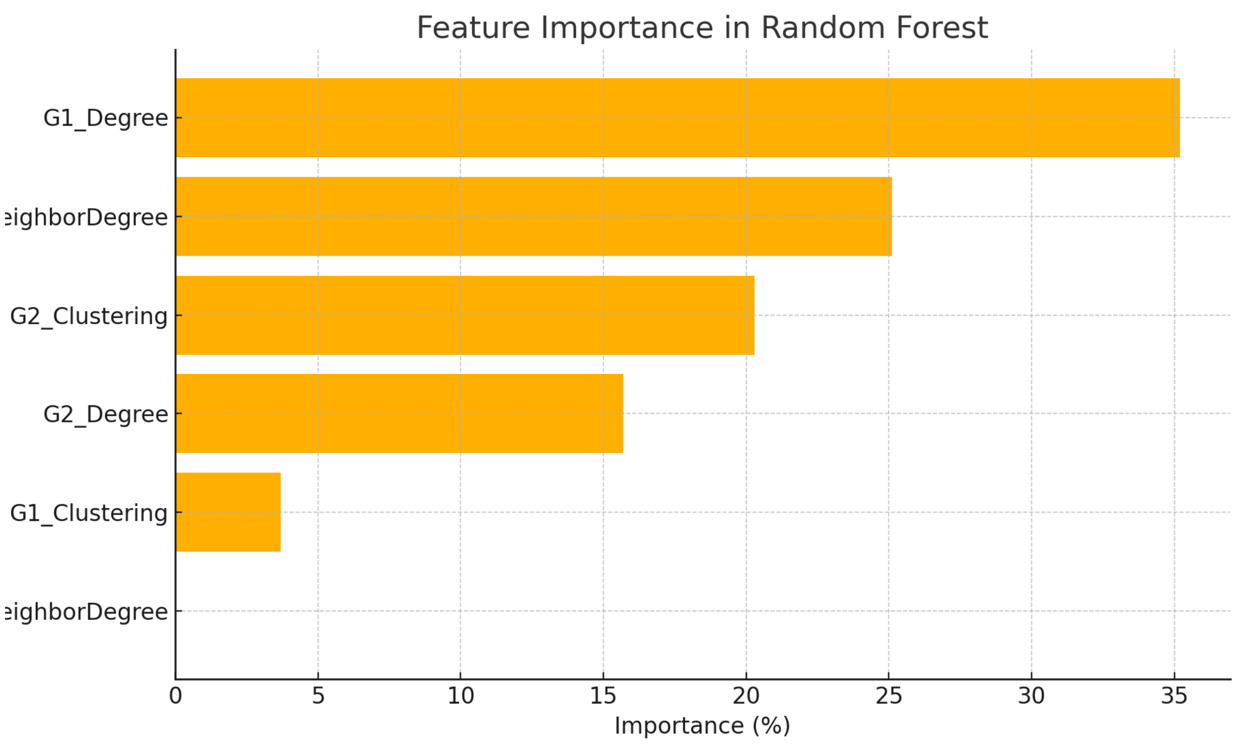
5. Seedless De-Anonymization

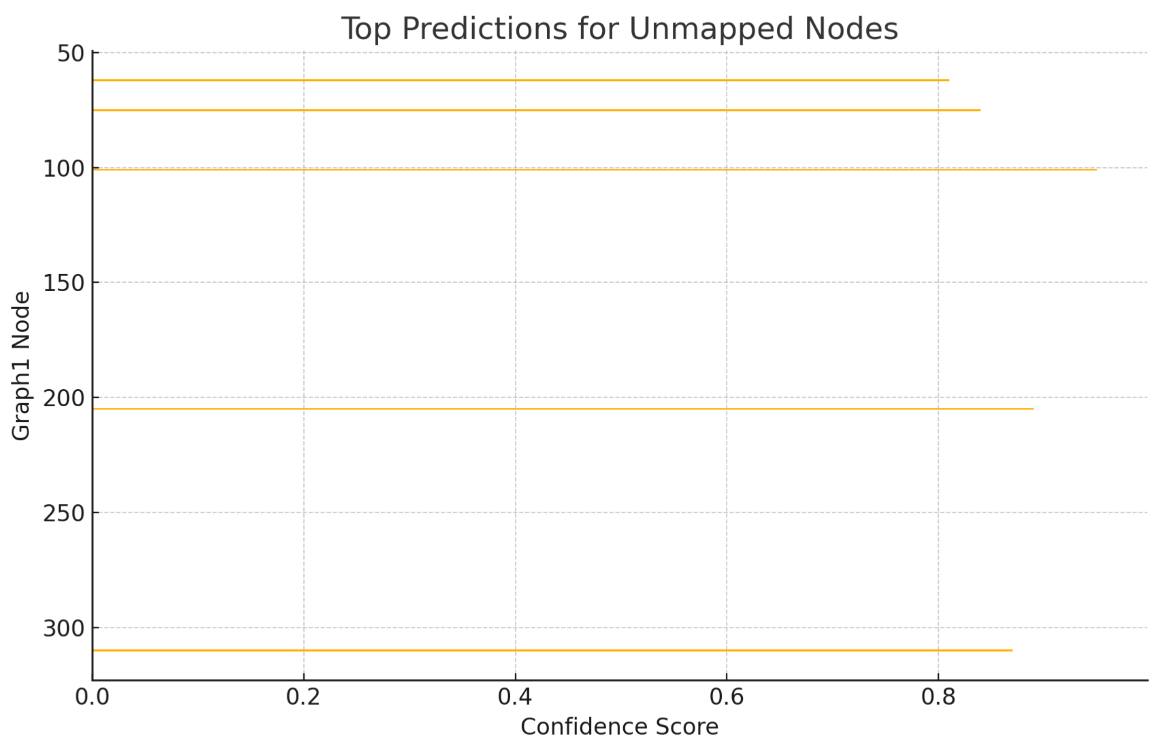
Seedless de-anonymization compares all nodes in Graph1 to Graph2 using cosine similarity of features, mapping nodes purely based on structural similarity.

```
def seedless_mapping(features_G1, features_G2):
    mapping = {}
    for _, row1 in features_G1.iterrows():
        g1_node = row1["Node"]
        g1_features = row1.iloc[1:].values.flatten()
        best_match = None
        best_score = float('-inf')
        for _, row2 in features_G2.iterrows():
            g2_node = row2["Node"]
            g2_features = row2.iloc[1:].values.flatten()
            score = cosine_similarity([g1_features], [g2_features])[0, 0]
            if score > best_score:
                best_match = g2_node
                best_score = score
        mapping[g1_node] = best_match
    return mapping
```

7. Visualizations and Results

Visualizations for feature distributions, feature importance, and mapping results.





8. Conclusion

1. Seed-Based De-Anonymization achieves high accuracy and is efficient when seed mappings are available.

2. Seedless De-Anonymization is computationally intensive but valuable when seed mappings are absent.

3. Recommendation: For large-scale graphs, seed-based methods are preferable if initial mappings are available.