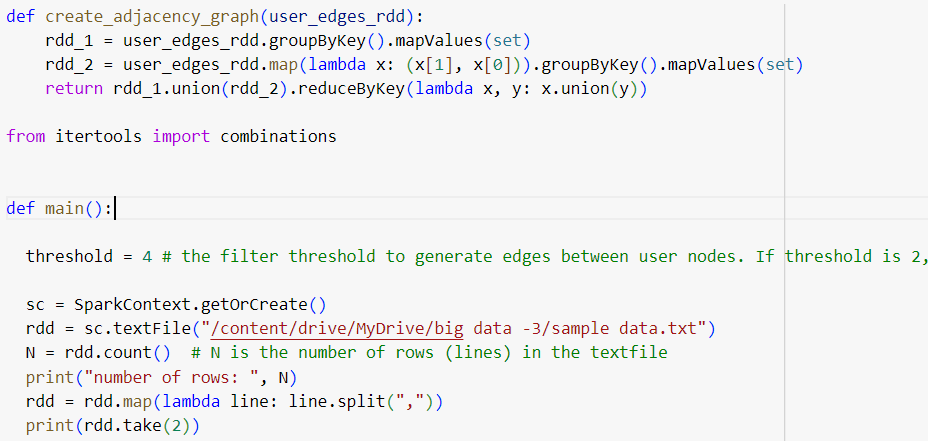
**Distributed Computing with Big Data Community Detection**

**Task 1: Graph Construction**

The task is begun by exploring the dataset's size and structure. An essential data structure is built, including users\_rdd, which stores user information and users\_dict, which optimizes data retrieval. A connection is also established between users based on a specified threshold and lays the groundwork for network analysis by creating an adjacency graph, user\_adjacency\_rdd. It also forms critical dictionaries for user neighbourhoods and node degrees.

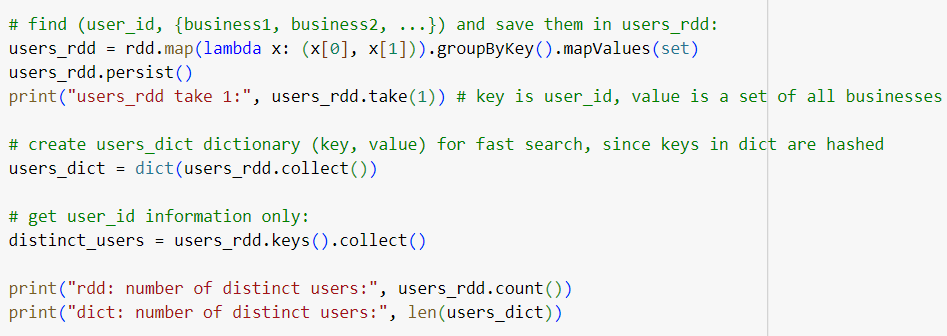
**Creating an Adjacency Graph**

A function, **create\_adjacency\_graph**, is defined to construct an adjacency graph from the input data. This processes the user\_edges\_rdd RDD, grouping user interactions and forming connections between users based on a specified threshold. The resulting graph represents users as nodes, and edges are established if users have reviewed the same business at least as many times as the threshold.



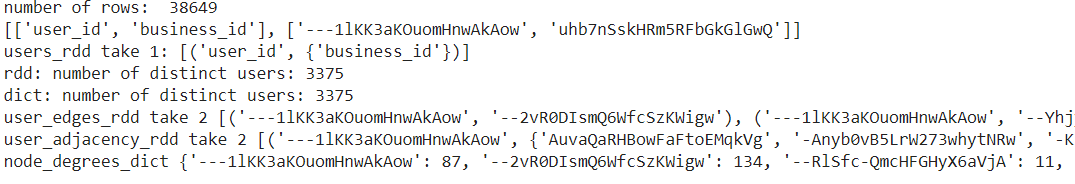
**Creating a User Dictionary**

A user dictionary and related data structures are built which operate on the RDD containing user-business interactions and perform the tasks. It constructs users' RDD by grouping data by user ID and creating sets of businesses each user has reviewed. Users RDD stores these associations as key-value pairs where 'user\_id' is the key, and the set of businesses they've reviewed is the value. This RDD is persisted for efficient access. A Python dictionary is created from 'users\_rdd,' allowing for rapid key-based searches due to dictionary key hashing. A list of distinct 'user\_id' values is obtained from users' rdd.keys,' providing the number of unique users in the dataset.



**User Edges and Adjacency Graph Construction**

Edges are created between users based on their shared business reviews. It iterates through all possible pairs of distinct users, checks if they have reviewed a sufficient number of common businesses as defined by the threshold and forms edges between them if the condition is met. The resulting edges are stored in an RDD. An adjacency graph is constructed, representing user nodes and their neighbouring users.



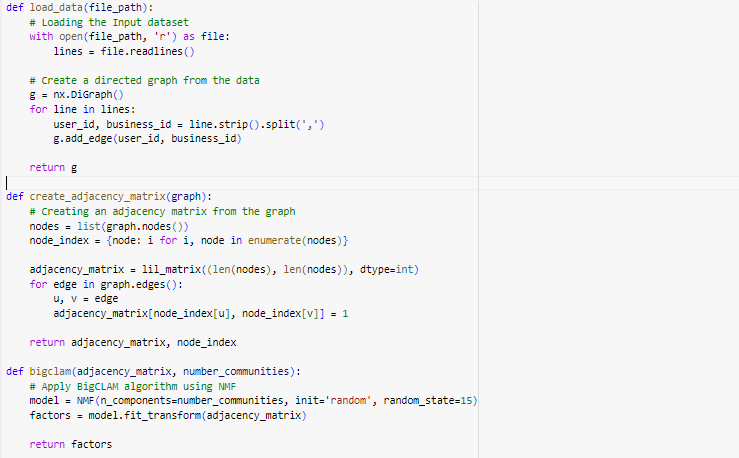
This output provides key insights into the dataset and graph construction. The dataset's size (38649 rows) and format are provided. The 'users\_rdd' structure is introduced, representing users and their reviewed businesses. The count of distinct users (3375) is mentioned, along with the creation of the 'users\_dict' for efficient data retrieval. A user edges based on the specified threshold and displays two sample edges is generated. An adjacency graph is formed and two important dictionaries are created to capture neighbourhood information and node degrees.

**Task 2: Implement BigCLAM to detect communities**

Task 2, focuses on community detection using the BigCLAM algorithm, an essential tool for uncovering groups of nodes with similar interaction patterns within networks. This task involves loading an input dataset, creating an adjacency matrix, applying the BigCLAM algorithm through non-negative matrix factorization, and finally, detecting and printing the emerging communities. It provides a practical demonstration of how machine learning and network analysis work together to reveal hidden structures in large-scale networks.

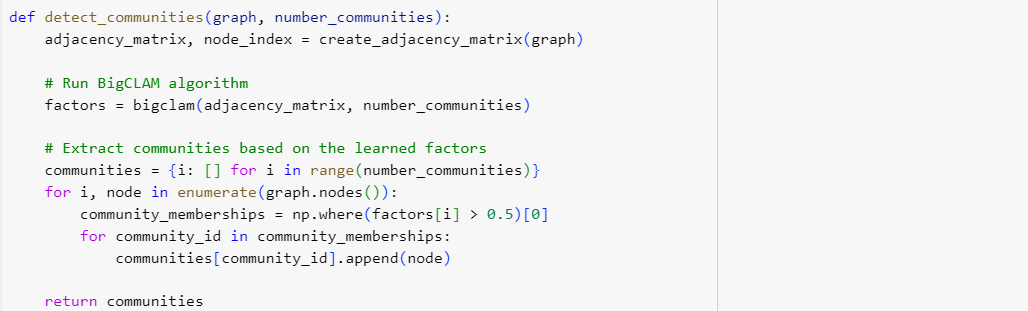
**Data preparation and community detection using the BigCLAM algorithm.**

Essential data preparation and community detection using the BigCLAM algorithm are done. The data is loaded from a specified file and constructs a directed graph representing user-business relationships. An adjacency matrix from this graph encodes node connections. The BigCLAM algorithm is applied through Non-Negative Matrix Factorization to identify communities within the graph, resulting in factors representing node-community affiliations. This code sets the stage for community detection in our social network dataset, facilitating the analysis of community structures.



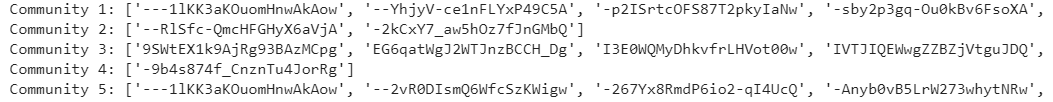
**Detecting Communities with BigCLAM Algorithm**

The detect\_communities function, analyzes a given graph and aims to identify communities within it. It does this by creating an adjacency matrix and node index from the graph, then employing the BigCLAM algorithm with Non-Negative Matrix Factorization to estimate community memberships. It forms communities by grouping nodes with substantial affiliations together, effectively partitioning the graph into distinct community structures.



**Community Detection Using BigCLAM**

The input dataset is loaded from a file, a graph representation of the data is built and then the BigCLAM algorithm is applied to detect communities within the graph. The input dataset consists of user-business interactions and this data is used to create a directed graph. The BigCLAM algorithm is employed with a specified number of communities and based on the learned factors, nodes are assigned to different communities. The detected communities are printed, showing the members of each community.

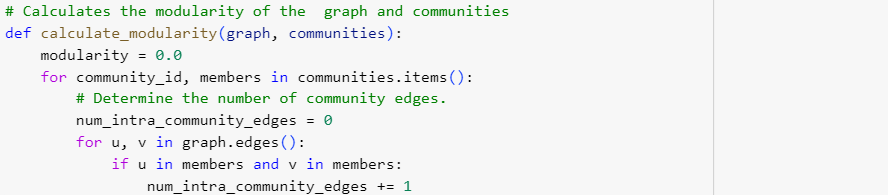


**Task 3: Finding the optimal number of communities**

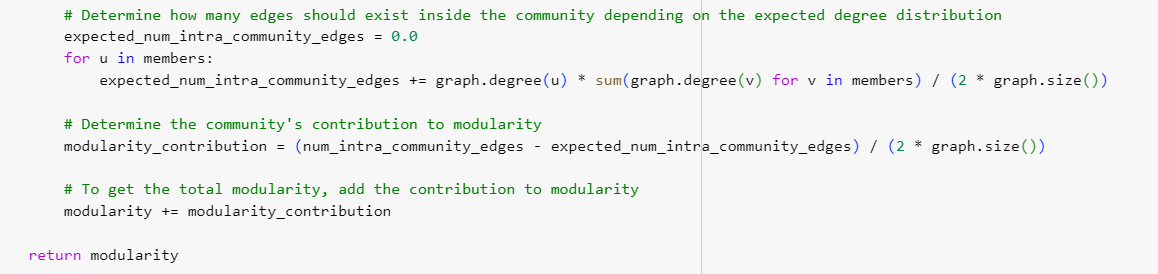
Task 3, focuses on identifying the optimal number of communities within a network. Communities, representing groups of densely interconnected nodes, are essential for network analysis. To achieve this, modularity is employed, a metric that measures community quality. By adjusting the number of communities and computing modularity scores, the configuration that maximizes modularity is identified. This task helps uncover meaningful patterns within complex networks, facilitating effective network analysis and visualization.

**Modularity Calculation for Community Evaluation**

The modularity of a graph with respect to the detected communities Is calculated. The function iterates through each community identified in the community dictionary. For each community, it counts the number of edges that exist exclusively within that community. These are edges where both nodes are members of the same community. Modularity is computed based on the difference between the observed number of intra-community edges and the expected number of such edges in a random network.

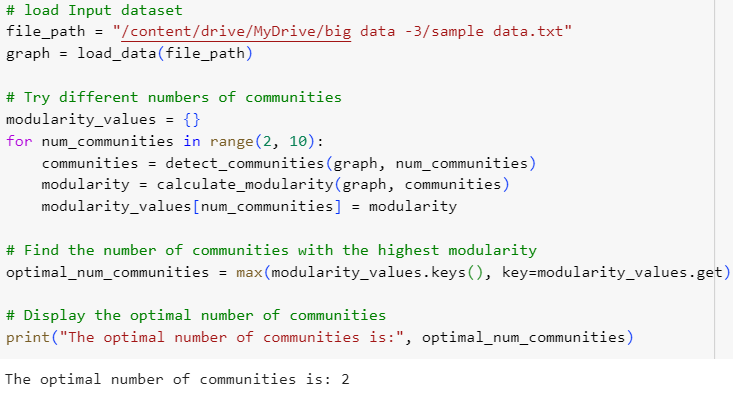


The modularity of detected communities within a graph is calculated, which is a measure of how well these communities are structured. Modularity is computed by evaluating the difference between the actual number of edges within communities and the expected number of such edges, given the degree distribution of nodes in the communities. The contribution of each community to the overall modularity is then determined, and this contribution is added up for all communities to obtain the total modularity score.



**Optimal Community Detection**

The optimal number of communities within a given graph is identified. This is by loading the input data and constructing a graph. Then iterate through a range of community numbers, calculating the modularity for each community configuration. Modularity measures the quality of community detection, with higher values indicating better results. The code identifies the number of communities that maximize modularity, revealing the optimal community structure for the graph.



**Challenges**

The most challenging aspect of the three tasks was undoubtedly the intricate nature of network data analysis. Task 1 began with the labour-intensive process of data preprocessing, transforming raw data into a structured graph. Task 2 introduced the BigCLAM algorithm, a sophisticated community detection method that required a deep understanding of matrix factorization and network modelling. The intricacies of parameter tuning and optimizing performance added to the complexity. In Task 3, determining the optimal number of communities involved computationally intensive iterations and modularity calculations.

**Interesting part**

The most enjoyable aspect of all three tasks was the opportunity to dive into real-world network analysis problems. Task 1 provided a sense of satisfaction as the data transformed into a meaningful graph. Task 2 was intellectually engaging, as it involved implementing state-of-the-art techniques to uncover hidden patterns and communities within the network. Finally, Task 3 offered a sense of accomplishment when discovering the optimal community structure. Despite the challenges, the tasks were immensely rewarding and provided valuable experience in data-driven decision-making.

**References**

Marco, Meoni. (2018). Mining Predictive Models for Big Data Placement.

Li, Wang., Dong, Li., Yousong, Zhu., Lu, Tian., Yi, Shan. (2021). Cross-Dataset Collaborative Learning for Semantic Segmentation. arXiv: Computer Vision and Pattern Recognition.

Mark, C., Lewis. (2019). Big Data Analytics with Spark. Doi: 10.1145/3287324.3287551

BigCLAM: <https://infolab.stanford.edu/~crucis/pubs/paper-nmfagm.pdf>

Girvan Newman Algorithm in pySpark: <https://github.com/nipunmanral/Community-Detection-In-Graphs>