Link Analysis on IMDB Dataset

Margherita Menegazzi, 974494

December 2022

1 Introduction

The aim of this project was to implement a Page Rank algorithm in order to analyze a dataset containing information about movies obtained from the IMDB website. More specifically, the Map-Reduce methodology was utilized to identify movies that share an actor or actress in the cast.

2 Dataset

The IMDB dataset was loaded from Kaggle in the Colab environment. The original dataset was composed of several files, for the purpose of this project only one file called 'title.principals.tsv' was necessary. This file contained the following columns:

- tconst (string) - alphanumeric unique identifier of the title.
- ordering (integer) – a number to uniquely identify rows for a given title Id.
- nconst (string) - alphanumeric unique identifier of the person's name.
- category (string) - the category of job that person was in.
- job (string) - the specific job title if applicable, else.
- characters (string) - the name of the character played if applicable, else.

Given the dimension of this dataset, when it was imported as a dataframe only the first 10.000 rows were selected for the analysis in order to make computations manageable on Colab.

First, the dataset was filtered in order to consider only the rows where the attribute 'category' corresponded to 'actor'.
Then, the following unwanted columns were dropped: 'ordering', 'category', 'job' and 'characters'.

Afterwards, the remaining columns of interest were renamed with more meaningful words: 'tconst' was renamed to 'title' and 'nconst' was renamed as 'actor'. Lastly, some useless characters were dropped from our attributes of interest so that they would both become more readable and easier to use for computations.

3 Analysis

The spark dataframe was transformed into a Resilient Distributed Dataset.
A first map task was performed to create key-value pairs where the key was 'title' and the value was 'actor'. This ordering was then inverted in order to have 'actor' as key and 'title' as value.
A join operation was performed in order to link couples of movies having one actor in common.
The result of this join operation was utilized in order to build the Adjacency Matrix, which indicates if two nodes (the movies) have an edge that connects them (an actor).
Then the Transition Matrix was built, this matrix indicates the probability to go from one node to another node.
A map function was applied to the Transition Matrix in order to create a keys list representing each movie id once distinctly.
Then a dictionary was created to connect every movie id with its position in sequence. This dictionary was used inside a function that does two things:

- takes the connection matrix and substitutes the original movie ids with the new sequential position number
- creates a new key-value pair structure where the key is the arrival node and the value is a tuple that has the starting node and the probability of going from the starting node to the arrival node

Two arrays were created for the page-rank: one for the page-rank values at time t and one for the page-rank values at time t-1.
A function was defined to measure the distance that will be used in order to make the page-rank algorithm converge. This distance is the mean square error between the page rank at time t-1 and the page rank at time t.
Lastly, the page-rank values were computed. The tolerance for convergence was set at 10e-70 and the maximum number of iterations was set at 350.

4 Results

The algorithm converges after 154 iterations, the page-rank values for the first 5 movies were printed:

- Movie 1: 0.00048638132295720214
- Movie 2: 0.00048638132295719856

- Movie 3: 0.00048638132295719785
- Movie 4: 0.00048638132295719807
- Movie 5: 0.00048638132295719357