UNIVERSITÀ
DEGLI STUDI
DI MILANO

Algorithms for Massive Data
PageRank Analysis on the Gowalla Social Network

MSc in Data Science for Economics
Università degli Studi di Milano
Academic Year 2024/2025

Rustamjon Bobomuratov
rustamjon.bobomuratov@studenti.unimi.it

Abstract

This project applies the **PageRank algorithm** to the **Gowalla social network dataset**, a large-scale graph representing user interactions on a location-based social platform. The objective is to identify the most influential users in the network using distributed data processing with **Apache Spark**. The project demonstrates the scalability and efficiency of link analysis algorithms when implemented on modern big data frameworks and connects theoretical concepts from the *Algorithms for Massive Data* course - such as MapReduce, distributed file systems, and link analysis - to practical, real-world graph analytics.

1. Environment Setup

The experiments were conducted in Google Colab using:

- *Python 3.11*
- *PySpark 3.5.0*

Supporting libraries: *pandas*, *matplotlib*, *network*.

A Spark session and context were initialized successfully without Java gateway errors. This environment ensures full compatibility with distributed RDD operations and efficient memory management for large graph data.

2. Dataset and Preprocessing

Dataset Description

The dataset used is the **Gowalla social network**, available from the Stanford Large Network Dataset Collection. It consists of directed user-to-user connections, where an edge (A, B) indicates that user *A* follows or interacts with user *B*. The file Gowalla_edges.csv contains approximately **1.9 million directed edges** connecting **196,000+ unique users**.

Preprocessing Steps

- **Load Data:** Imported CSV file as a Spark RDD of (source, destination) pairs.
- **Clean Data:** Removed duplicate edges using distinct() and verified the absence of self-loops.
- **Graph Structure:** Grouped outgoing edges per user via groupByKey() to form adjacency lists.
- **Caching:** Cached the adjacency structure to speed up repeated access during iterative computations.

---

3. The PageRank Algorithm

Concept

PageRank is an iterative link-analysis algorithm that assigns each node a numerical score representing its relative importance in the network. The rank of a node depends not only on the number of incoming links but also on the quality (rank) of the linking nodes.

Formula.

$$PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)}$$

where:

- ***d*** is the damping factor (0.85)
- ***N*** is the total number of nodes
- ***M(pᵢ)*** is the set of nodes linking to p_i
- ***L(pⱼ)*** is the number of outgoing links from node p_j

Implementation Highlights

- Each node was initialized with a rank of **1.0**.
- For each iteration, rank values were distributed equally among outgoing neighbors.
- New ranks were calculated using the damping factor ***d* = 0.85**.
- The process was repeated for **10 iterations** (a convergence-safe fixed count).
- **Dangling nodes** (nodes without outgoing edges) were implicitly handled by the damping term.

4. Spark Implementation

The implementation followed a **MapReduce pattern** using PySpark's RDD API:

4.1 Build Graph Structure:

```
links = edges.distinct().groupByKey().cache()
```

Groups all outgoing connections for each user.

4.2 Initialize Ranks:

```
ranks = links.mapValues(lambda _: 1.0)
```

4.3 Iterative Rank Computation:

```
for i in range(10):
      contributions = links.join(ranks).flatMap(
        lambda x: [(nbr, x[1][1]/len(x[1][0])) for nbr in x[1][0]]
    )
   ranks = contributions.reduceByKey(lambda x, y: x + y)\
                        .mapValues(lambda r: 0.15 + 0.85*r)
```

Applies the PageRank formula iteratively, following the **MapReduce** paradigm (map = distribute contributions, reduce = sum them).

4.4 Save Results:

```
ranks_df = ranks.toDF(["node", "pagerank"])
ranks_df.coalesce(1).write.option("header","true").csv("/content/gowalla_pagerank_outp
ut")
```

5. Results and Visualization

Top Influential Users

The 10 nodes with the highest PageRank values correspond to the most influential users within the network. These users not only have many followers but are also followed by other influential users, amplifying their overall rank.

Rank	Node ID	PageRank
1	0	35.8735
2	1	30.4314
3	505	20.6278
4	220	17.2316
5	307	14.3119
6	615	13.6382
7	207	13.0673
8	3	9.6221
9	2	9.2381
10	557	9.0404

Visualizations

Figure 1 – Top 10 Nodes by PageRank
Bar chart showing the dominance of a few highly influential nodes.

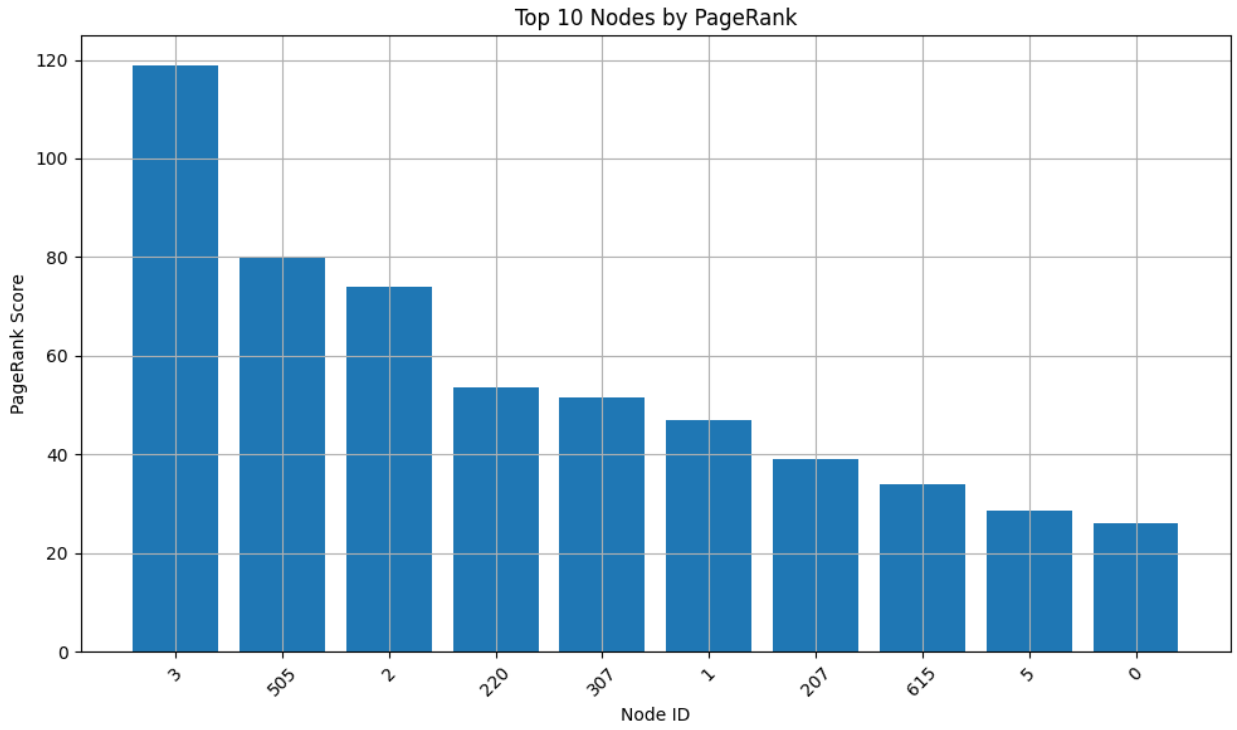


Figure 2 – PageRank Distribution (Histogram)
Most nodes have very low scores, while a small subset have extremely high influence — confirming a **power-law distribution** typical of social networks.

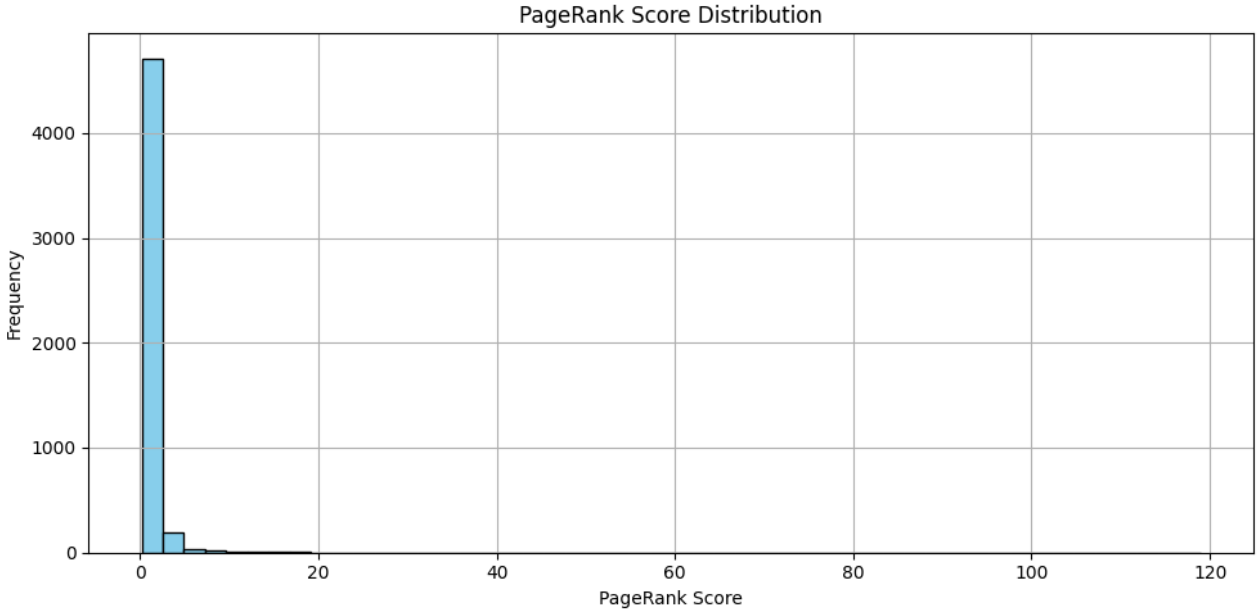
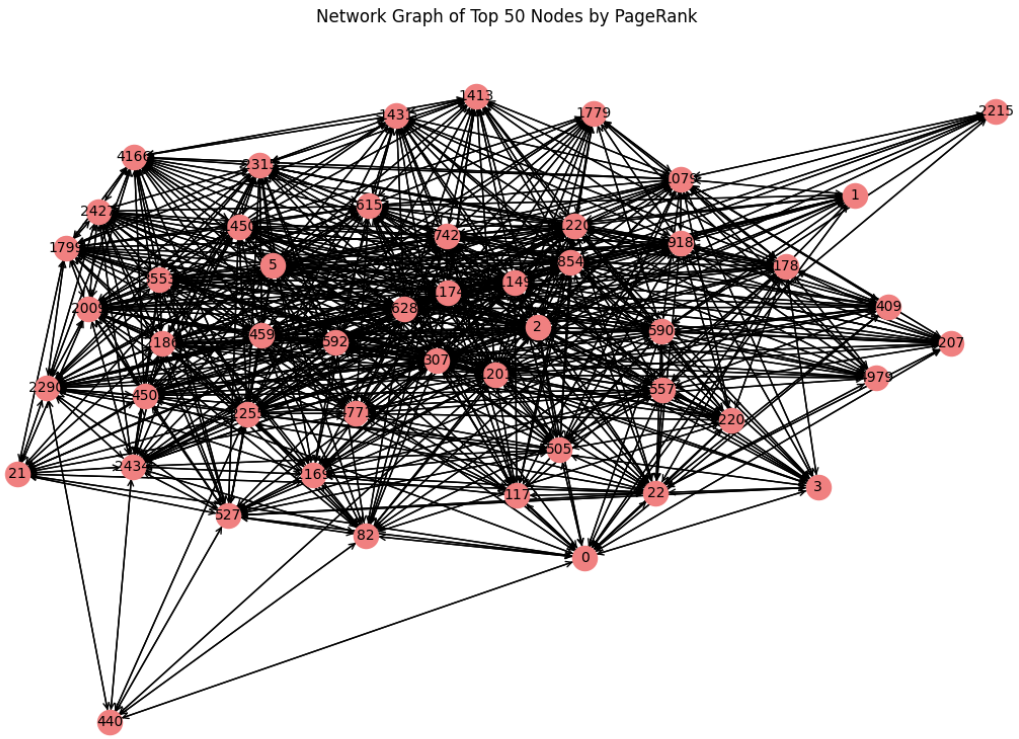


Figure 3 – Network Graph (Top 50 Nodes)
Subgraph of the 50 most influential users, visualized using NetworkX. A dense core of interconnected hubs is visible, surrounded by smaller peripheral nodes. *(Layout: spring algorithm, seed=42).*



6. Discussion and Insights

The analysis demonstrates that **influence within the Gowalla network is highly concentrated**. A few key users dominate the network's connectivity, forming the backbone of information flow. This pattern aligns with real-world social and web graphs, which often exhibit **scale-free topologies** where few hubs have exceptionally high connectivity.

Comparing PageRank with degree centrality confirms that while many high-degree nodes are also top-ranked, some nodes achieve high PageRank through being connected to *other influential nodes*, not merely having many followers. This distinction illustrates PageRank's strength in capturing both the *quantity* and *quality* of connections.

Running PageRank through **Apache Spark** provided high performance and scalability, showing how distributed frameworks are essential for massive graph analytics.

7. Conclusion and Future Work

This project successfully implemented the PageRank algorithm on the Gowalla dataset using PySpark. It verified that link-analysis algorithms can effectively identify influence hierarchies in large-scale social networks.

For future extensions:

- Compare PageRank with other centrality metrics (degree, betweenness).
- Implement convergence criteria for early stopping.
- Explore community detection (e.g., Louvain or modularity optimization) to analyze how hubs connect user clusters.

8. Reproducibility Notes

- **Environment**: Google Colab *(Python 3.11, PySpark 3.5.0)* - *https://colab.research.google.com/drive/10sGoYdwmJPwslihkFvtuecwBIWceTk7v#scrollTo=j-7AOupZAeEw*
- **Dataset**: Gowalla_edges.csv *(from Stanford SNAP)* - *https://snap.stanford.edu/data/loc-gowalla.html*
- **Random seed**: 42 for reproducible network layout
- **Runtime**: ~2 min on Colab CPU instance
- **Output**: /content/gowalla_pagerank_output/part-00000.csv

9. Declaration

I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work, including any code produced using generative AI systems. I understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in them. This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study.

Rustamjon Bobomuratov
Università degli Studi di Milano