Node and Link Analysis

Pablo Mollá Chárlez

Contents

1 Introduction

Node and Link Analysis plays **a pivotal role in understanding complex networks**, such as social media platforms, web pages, biological systems, and transportation networks. This field focuses on quantifying the importance of nodes (entities) and predicting relationships (links) within a network. We will study the following sections:

- **Hubs and Authorities** help evaluate the significance of nodes, such as identifying influential users in social networks or key pages in web search, using measures like centrality and the HITS algorithm.
- **PageRank**, originally developed by Google, assigns importance scores to nodes, crucial for ranking web pages and enhancing information retrieval systems.
- **Link Prediction** tackles the challenge of forecasting future or missing connections, enabling applications like friend recommendations, network growth modeling, and predicting protein-protein interactions in biology.

By analyzing the interplay of nodes and links, these methods drive advancements in search engines, recommendation systems, and network science.

2 Hubs and Authorities

To identify nodes that are more "**interesting**" or **influential in a network**, we estimate the worth of each node. These measures can combine textual or profile information with the network's link structure, making them valuable in applications like information retrieval. The links between nodes play a crucial role in determining their importance.

2.1 Estimating Node Worth: Centrality Measures

The rank of a node v can be evaluated using centrality measures:

- **Betweenness Centrality**: Measures the extent to which a node lies on the shortest paths between other nodes.

$$\text{Betweenness}(v) = \sum_{v_1, v_2 \neq v} \frac{\#\text{shortest paths from } v_1 \text{ to } v_2 \text{ containing } v}{\#\text{shortest paths from } v_1 \text{ to } v_2}$$

- **Closeness Centrality**: Measures how close a node is to all other nodes.

$$\text{Closeness}(v) = \frac{1}{\sum_{v'} d(v, v')}$$

- **Harmonic Centrality**: Variant of closeness that uses inverse distances.

$$\text{Harmonic}(v) = \sum_{v' \neq v} \frac{1}{d(v, v')}$$

- **Spectral Centrality**: Based on eigenvalues of the adjacency matrix. - Katz Centrality:

$$\text{Katz}(v) = \sum_{k} \sum_{v'} \alpha^k A^k_{vv'}$$

  HITS, Eigenvector Centrality, and PageRank measure importance based on node connectivity.

- **Degree Centrality**: Measures the number of direct connections a node has.

  Example: In a network with unweighted edges, a node with 5 neighbors has a degree centrality of 5.

2.2 Principle of Repeated Movement

The principle suggests a two-way relationship between nodes:

- **Authorities**: Nodes that are important because many good nodes link to them.
- **Hubs**: Nodes that are important because they link to many good authorities.

Each node $i \in V$ has both, an authority score a_i and a hub score h_i. These scores are updated iteratively.

2.3 HITS Algorithm

The HITS Algorithm (Hyperlink-Induced Topic Search) is a graph-based method used to identify two key types of nodes in a network: authorities and hubs. As previously mentioned, authorities are nodes that contain valuable or relevant information, while hubs are nodes that link to many authorities. The algorithm iteratively assigns and updates two scores, authority score and hub score based on the structure of incoming and outgoing links in the network.

1. **Initialization**: Assign initial scores: $a_i = 1, \quad h_i = 1 \quad \forall i \in V$
2. **Authority Update**: Update authority scores: $a_i = \sum_{j \in I(i)} h_j$

   A node's authority score is determined by the sum of the hub scores of the nodes linking to it.
3. **Hub Update**: Update hub scores: $h_i = \sum_{j \in O(i)} a_j$

   A node's hub score is determined by the sum of the authority scores of the nodes it links to
4. **Normalization**: Normalize scores so that:

$$\sum_i a_i = 1, \quad \sum_i h_i = 1$$

These scores are updated iteratively using matrix operations until they converge, highlighting the most authoritative and hub-like nodes in the network. The algorithm repeats these steps for a fixed number of iterations k, improving scores with each iteration.

2.3.1 HITS Matrix View

The HITS algorithm in matrix form provides a formal way to compute authority (a) and hub (h) scores using linear algebra and iterative updates. The key components include:

- **Matrix** A: Represents the adjacency matrix of the directed graph, where $A[i, j] = 1$ if node i links to node j.
- **Authority Vector** (a): Where $a[i]$ is the authority score of node i. Nodes with high $a[i]$ are highly trusted or linked to by good hubs.
- **Hub Vector** (h): The hub score of node i is $h[i]$. Nodes with high $h[i]$ link to many authoritative nodes.

The matrix update at iteration k can be done as follows:

- **Authority Update**:

$$a^{(k)} = A^\top h^{(k-1)}$$

  It computes new authority scores based on the hub scores of nodes linking to them, and an improved version which simplifies the computations by only depending on A:

$$\boxed{\mathrm{a}^{(k)} = (A^\top A)^{k-1} A^\top h^{(0)}}$$

- **Hub Update**:

$$h^{(k)} = Aa^{(k)}$$

It computes new hub scores based on the authority scores of nodes they link to, and its improved version is:

$$\boxed{\mathrm{h}^{(k)} = (AA^{\top})h^{(k-1)}}$$

2.3.2 HITS Example

Consider the following network with three nodes where A links to B and C, B links to C and C links to no other nodes. Let's follow the steps as described.

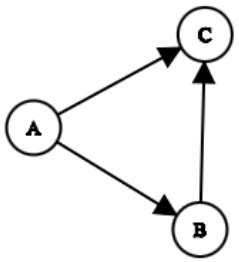

Figure 1: HITS Graph

1. **Step 1:** Initialization Assign initial scores to all nodes:

$$a_A = 1, \quad a_B = 1, \quad a_C = 1, \quad h_A = 1, \quad h_B = 1, \quad h_C = 1$$

2. **Step 2:** Authority Update Rule

$$\begin{aligned} a_A &= 0 \quad \text{(no incoming links for } A\text{)} \\ a_B &= h_A = 1 \quad \text{(linked by } A\text{)} \\ a_C &= h_A + h_B = 2 \quad \text{(linked by } A \text{ and } B\text{)} \end{aligned}$$

3. **Step 3:** Hub Update Rule

$$\begin{aligned} h_A &= a_B + a_C = 1 + 2 = 3 \quad \text{(links to } B \text{ and } C\text{)} \\ h_B &= a_C = 2 \quad \text{(links to } C\text{)} \\ h_C &= 0 \quad \text{(no outgoing links for } C\text{)} \end{aligned}$$

4. **Step 4:** Normalize the scores so that their sum equals 1 for each set (authority and hub scores).

$$\begin{aligned} a_A = 0;\ a_B = \frac{1}{3};\ a_C = \frac{2}{3} \\ h_A = \frac{3}{5};\ h_B = \frac{2}{5};\ h_C = 0 \end{aligned}$$

5. **Step 5:** Repeat the authority and hub updates for a fixed number of iterations or until the scores converge.

At the end of the iterations C will have the highest authority score since it is linked to by two nodes and A will have the highest hub score since it links to two authoritative nodes.

3 PageRank

3.1 Definitions

PageRank is an algorithm used to rank nodes (e.g., web pages) in a graph based on their importance. Developed by Larry Page and Sergey Brin, it is fundamental to search engines like Google. The algorithm models a random surfer navigating the web, starting at a random page and either following a link or jumping to a random page with a certain probability (the damping factor). The importance of a page is determined by the likelihood that the surfer will land on that page, which is calculated as the stationary distribution of a Markov chain formed by the web's link structure. Pages with more inbound links from other important pages receive higher ranks.

- **Definition 1:** Important nodes are those linked to by other important nodes. This definition is recursive.
- **Definition 2:** The **Random Surfer Model**, where a random surfer walks on the graph:
  1. The surfer starts at a node (e.g., a web page) and chooses randomly one of the outgoing links (e.g., another page).
  2. The surfer continues this process for subsequent nodes.
  3. At each step, the surfer has a **probability** $1-\alpha$ (called the damping factor) of jumping randomly to any other node in the graph.

  The importance of a page is defined as the stationary probability that the surfer is on that page as time approaches infinity.

The two definitions are mathematically equivalent.

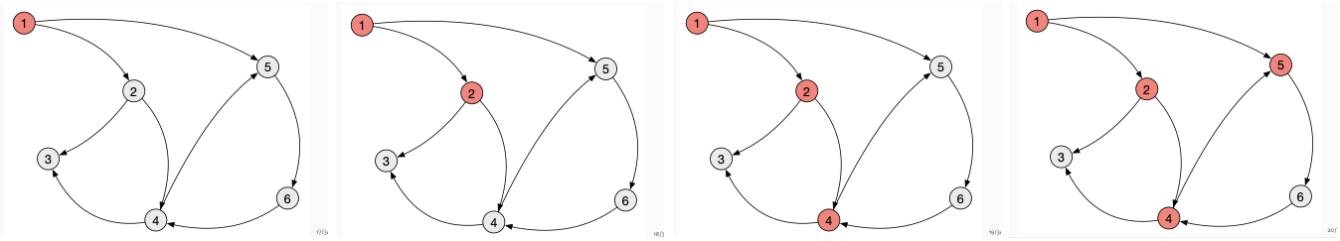

Figure 2: PageRank: Random Surfer

3.2 PageRank Equation and Algorithm

For a graph $G = (V, E)$ with n nodes, where each node i has:

- Incoming neighbours: I_i,
- Outgoing neighbours: O_i,

The PageRank of node i, $p(i)$ or similarly, the probability of the random surfer choosing the node i is given by:

$$p(i) = \alpha \sum_{j \in I_i} \frac{p(j)}{|O_j|} + \frac{1-\alpha}{n}$$

where α is the damping factor.

3.2.1 PageRank Algorithm

1. Start with initial values $p(i) = \frac{1}{n}$ for all i.
2. Iteratively apply the PageRank equation for each node i.
3. Stop when the probabilities converge (i.e., when the difference between iterations is below a threshold).

3.2.2 Monte Carlo Approximation:

It's possible to simulate N random walks on the graph via the Monte Carlo Approximation. The PageRank of node i approximated is:

$$p(i) = \frac{v_i}{N}$$

where v_i is the number of visits to node i among the N random walks.

3.3 PageRank: Matrix View

To visualize the PageRank algorithm using matrix variables, we need to define the following matrix and vector:

- A transition matrix M, where:

$$m_{ij} = \frac{a_{ij}}{|O_i|}$$

  Here, a_{ij} is the entry of the adjacency matrix, and M is a stochastic matrix (each column sums to 1).

- A teleportation vector t, which is also stochastic:

$$t = \left(\frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n}\right)^T$$

3.4 Matrix View and Markov Chains

The PageRank process can be seen as a Markov chain:

- A Markov Chain models a stochastic sequence of events (states) where the current state depends only on the previous state.
- **Ergodic Markov Chain:** Always converges to a stationary probability distribution of states. The conditions are:
  - *Irreducible:* There is a sequence of transitions from any state to another (no disconnected subsets of states).
  - *Aperiodic:* No state has a period greater than 1.

The PageRank vector at step k is given by:

$$p^{\langle k \rangle} = \alpha M p^{\langle k-1 \rangle} + (1-\alpha)t$$

Where the terms included are described as follows:

- $p^{\langle k \rangle}$ is the PageRank vector at step k, representing the probabilities of being on each node at that time step.
- α is the damping factor (typically set around 0.85), indicating the probability of following a link from the current page.
- M is the transition matrix, where each entry M_{ij} represents the probability of moving from node j to node i (i.e., $\frac{a_{ij}}{|O_j|}$, where a_{ij} is the adjacency matrix entry and $|O_j|$ is the number of outgoing links from node j).
- $p^{\langle k-1 \rangle}$ is the PageRank vector from the previous step.
- t is the teleportation vector, which represents the probability distribution for jumping to any node uniformly at random (usually $\frac{1}{n}$ for each node if there are n nodes).
- $(1-\alpha)$ represents the probability of the random surfer "teleporting" to any node instead of following a link.

This converges to the stationary distribution:

$$p^{\infty} = (I - \alpha M)^{-1}(1-\alpha)t$$

where the teleportation vector ensures the Markov chain is ergodic.

3.5 Variants of PageRank

Depending on the teleportation behavior, we have different variants:

- **Classic PageRank:** The surfer can jump to any node with equal probability.
- **Personalized PageRank:** The surfer can only jump to their start page:

$$t = (0, 1, 0, \ldots, 0)^T$$

- **Topic-Sensitive PageRank:** The surfer can only jump to a specific set of topic-related nodes:

$$t = \left(0, \frac{1}{l}, \frac{1}{l}, \ldots, 0\right)^T$$

  where l is the size of the topic-related set.

4 Link Prediction

Social networks are evolving, and new relationships (links) appear over time. The Link Prediction Problem involves predicting which links are more likely to appear in a social network. This assumes that links can be predicted through analysis based only on the social network itself. The applications of Link Prediction include:

- New link recommendation (e.g., new friends)
- Missing link inference
- Analyzing network evolution

4.1 Link Scoring Function

The goal is to estimate the score of potential links for a graph $G = (V, E)$, i.e., a function defined on the missing links $E' = (V \times V) \setminus E$:

$$\text{score} : E' \to \mathbb{R}^+$$

For a given node i, $\text{score}(i, j)$ ranks all unlinked nodes j relative to i — the highest scores correspond to the most likely new links.

4.2 Node Neighbourhood Scores

Node Neighbourhood Scores rely only on immediate and local connections (e.g., shared or total neighbours). Computationally inexpensive and work well for dense, locally connected graphs.

- **Common Neighbours**: Counts the number of common neighbours:

$$\text{score}(i, j) = |N(i) \cap N(j)|$$

- **Jaccard Coefficient**: Measures similarity between neighbourhood sets:

$$\text{score}(i, j) = \frac{|N(i) \cap N(j)|}{|N(i) \cup N(j)|}$$

- **Preferential Attachment**: The score is proportional to the degrees of each node:

$$\text{score}(i, j) = k_i \cdot k_j$$

4.3 Path-Based Scores

Extend the scope to paths connecting nodes, incorporating the global structure of the graph. Can capture more nuanced relationships but are computationally more expensive.

- **Inverse Distance**: The score is inversely proportional to the distance between nodes:

$$\text{score}(i, j) = \frac{1}{d_{ij}}$$

- **Katz**: A weighted sum of all paths between i and j:

$$\text{score}(i, j) = \sum_{l=1}^{\infty} \beta^l \cdot |\text{paths}_{i,j}^{\langle l \rangle}|$$

  where $\beta \in (0, 1)$.

4.4 Random Walk-Based Scores

Model dynamic processes like information flow or exploration, capturing deeper structural and probabilistic relationships. Often require iterative computations and probabilistic modeling.

- **Hitting Time**: Time it takes a random walk from i to reach j:

$$\text{score}(i, j) = H_{i,j}$$

- **Personalized PageRank**: Any PageRank-related measure where the teleportation vector is rooted at i.
- **SimRank**: A recursive score based on neighbours:

$$\text{score}(i, j) = \gamma \cdot \frac{\sum_{a \in N(i)} \sum_{b \in N(j)} \text{score}(a, b)}{k_i \cdot k_j}$$

  where γ is a damping factor.