Centrality Algorithms

GRAPH AND AI

TERRI HOARE – FEBRUARY 2021

GRAPH ALGORITHMS MARK NEEDHAM AND AMY E. HODLER, O'REILLY 2019

Centrality Algorithms

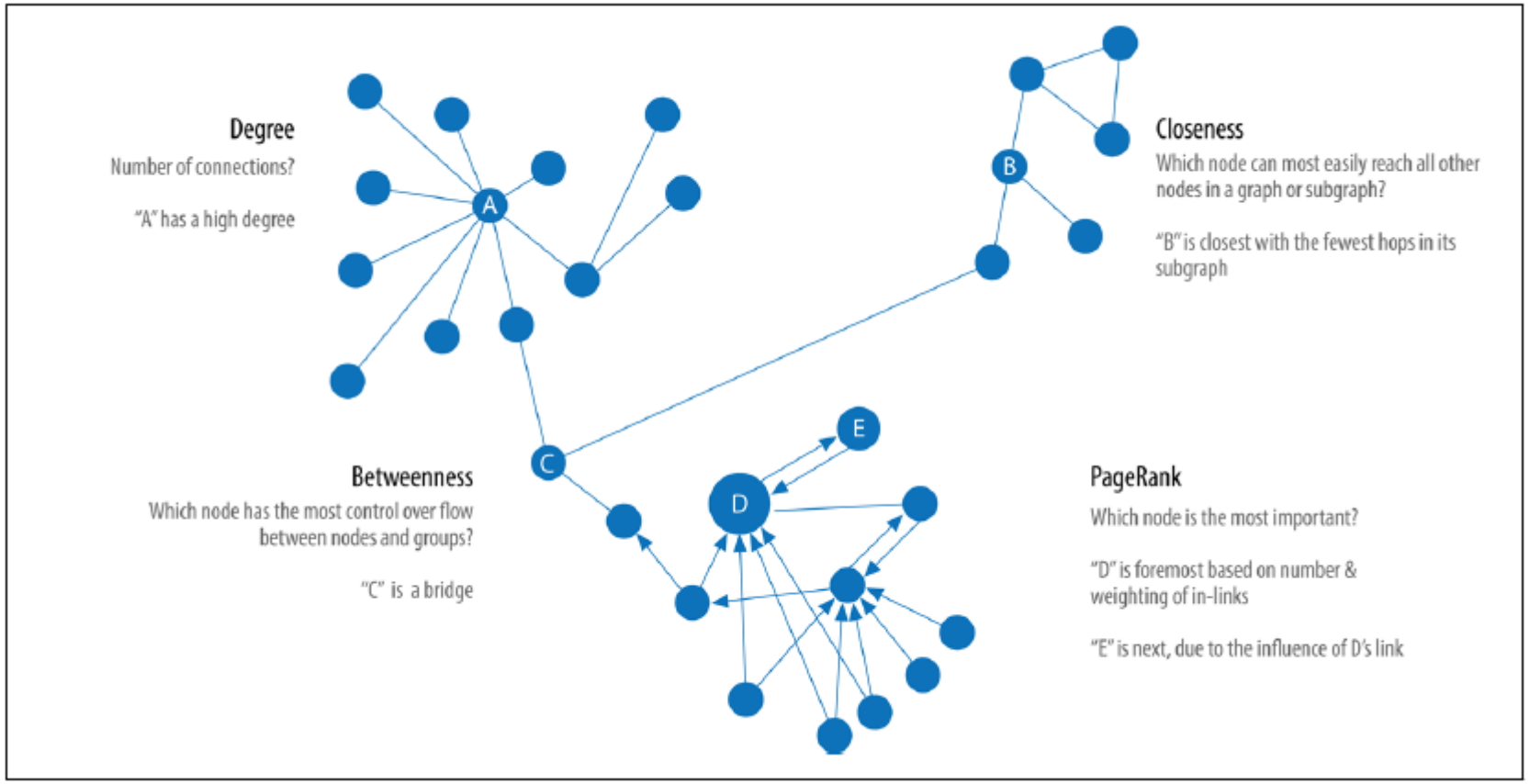
Introduction

Centrality algorithms are used to understand the roles of particular nodes in a graph and their impact on that network. They're useful because they identify the most important nodes and help us understand group dynamics such as credibility, accessibility, the speed at which things spread, and bridges between groups. Although many of these algorithms were invented for social network analysis, they have since found uses in a variety of industries and fields.

- **Degree Centrality** as a baseline metric of connectedness
- **Closeness Centrality** for measuring how central a node is to the group, including two variations for disconnected groups
- **Betweenness Centrality** for finding control points, including an alternative for approximation
- **PageRank** for understanding the overall influence, including a popular option for personalization

Centrality Algorithms

Types of Questions They Answer



Graph Algorithms Mark Needham and Amy E. Hodler, O'Reilly 2019

Centrality Algorithms

Types of Questions They Answer

Algorithm type	What it does	Example use	Spark example	Neo4j example
Degree Centrality	Measures the number of relationships a node has	Estimating a person's popularity by looking at their in-degree and using their out-degree to estimate gregariousness	Yes	No
Closeness Centrality Variations: Wasserman and Faust, Harmonic Centrality	Calculates which nodes have the shortest paths to all other nodes	Finding the optimal location of new public services for maximum accessibility	Yes	Yes
Betweenness Centrality Variation: Randomized-Approximate Brandes	Measures the number of shortest paths that pass through a node	Improving drug targeting by finding the control genes for specific diseases	No	Yes
PageRank Variation: Personalized PageRank	Estimates a current node's importance from its linked neighbors and their neighbors (popularized by Google)	Finding the most influential features for extraction in machine learning and ranking text for entity relevance in natural language processing.	Yes	Yes

Graph Algorithms Mark Needham and Amy E. Hodler, O'Reilly 2019

Centrality Algorithms

Example Graph Data – Twitter Social Graph

Centrality algorithms are relevant to all graphs, but social networks provide a very relatable way to think about dynamic influence and the flow of information. The examples in this chapter are run against a small Twitter-like graph. You can load the nodes and relationships files using the accompanying CYPHER script "import centrality.cypher."

id
Alice
Bridget
Charles
Doug
Mark
Michael
David
Amy
James

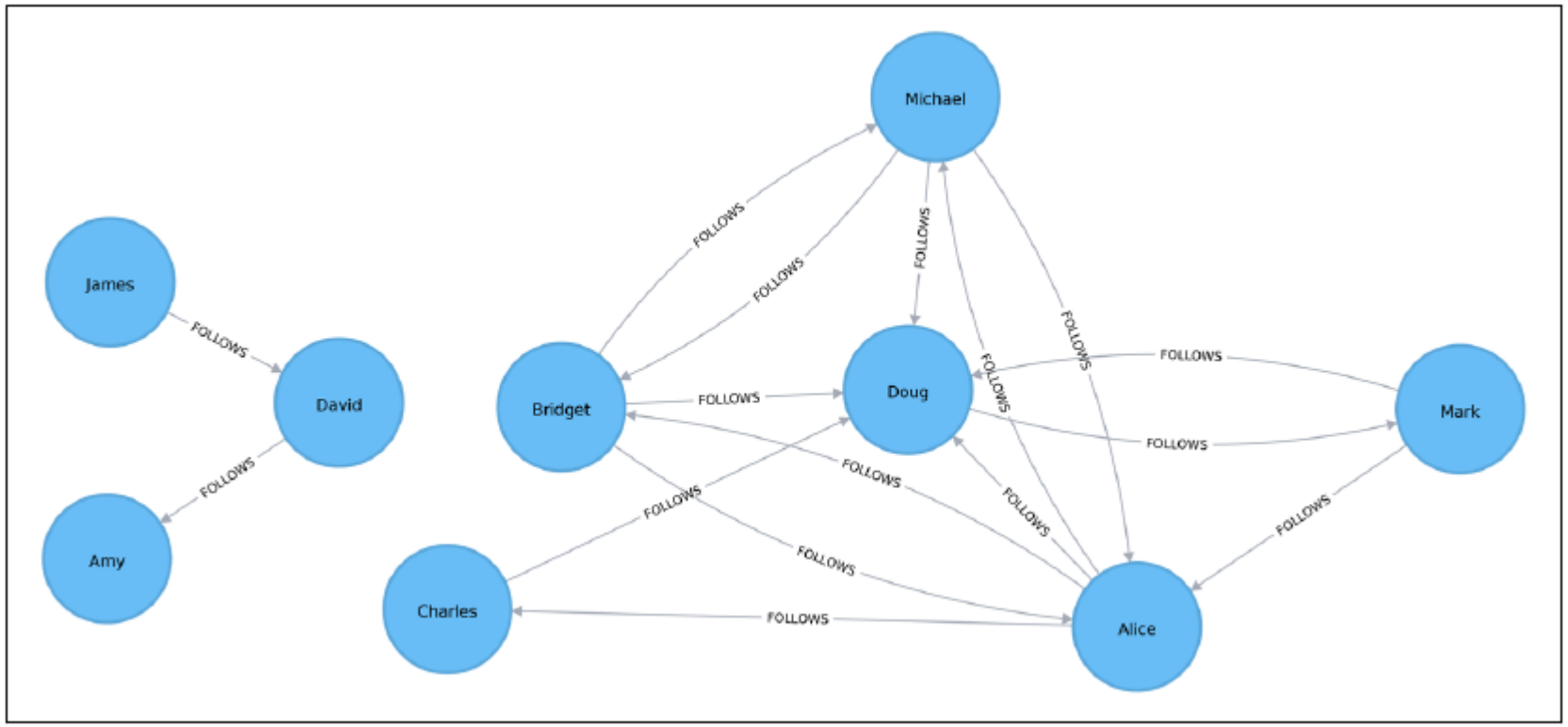
src	dst	relationship
Alice	Bridget	FOLLOWS
Alice	Charles	FOLLOWS
Mark	Doug	FOLLOWS
Bridget	Michael	FOLLOWS
Doug	Mark	FOLLOWS
Michael	Alice	FOLLOWS
Alice	Michael	FOLLOWS
Bridget	Alice	FOLLOWS
Michael	Bridget	FOLLOWS

src	dst	relationship
Charles	Doug	FOLLOWS
Bridget	Doug	FOLLOWS
Michael	Doug	FOLLOWS
Alice	Doug	FOLLOWS
Mark	Alice	FOLLOWS
David	Amy	FOLLOWS
James	David	FOLLOWS

Graph Algorithms Mark Needham and Amy E. Hodler, O'Reilly 2019

Centrality Algorithms

Twitter Social Graph Model



Graph Algorithms Mark Needham and Amy E. Hodler, O'Reilly 2019

Centrality Algorithms

Degree Centrality

Degree Centrality is the simplest of the algorithms. It counts the number of incoming and outgoing relationships from a node and is used to find popular nodes in a graph. Degree Centrality was proposed by Linton C. Freeman in his 1979 paper "Centrality in Social Networks: Conceptual Clarification".

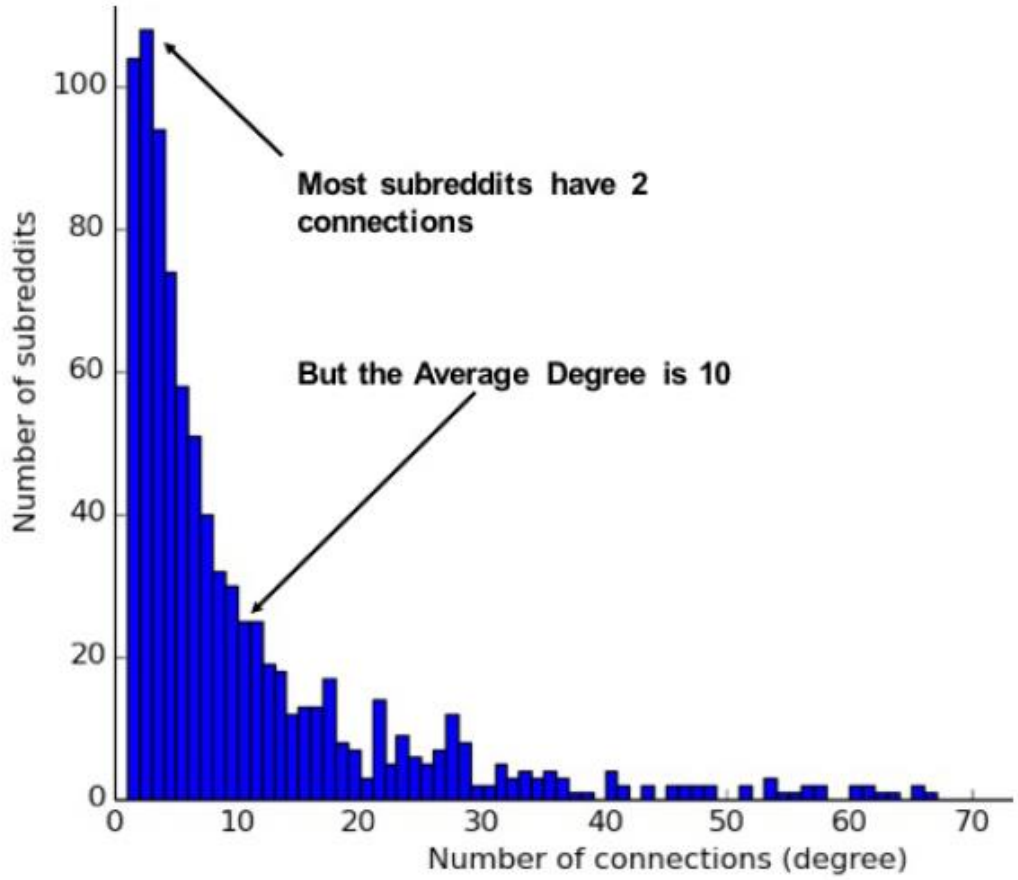
Reach

Understanding the reach of a node is a fair measure of importance. How many other nodes can it touch right now? The **degree** of a node is the number of direct relationships it has, calculated for in-degree and out-degree. You can think of this as the immediate reach of node. For example, a person with a high degree in an active social network would have a lot of immediate contacts and be more likely to catch a virus circulating in their network.

Centrality Algorithms

Average Degree of a Network

The **average degree** of a network is simply the total number of relationships divided by the total number of nodes; it can be heavily skewed by high degree nodes. The **degree distribution** is the probability that a randomly selected node will have a certain number of relationships. The actual distribution of connections among subreddit topics (Figure). If you simply took the average, you'd assume most topics have 10 connections, whereas in fact most topics only have 2 connections.

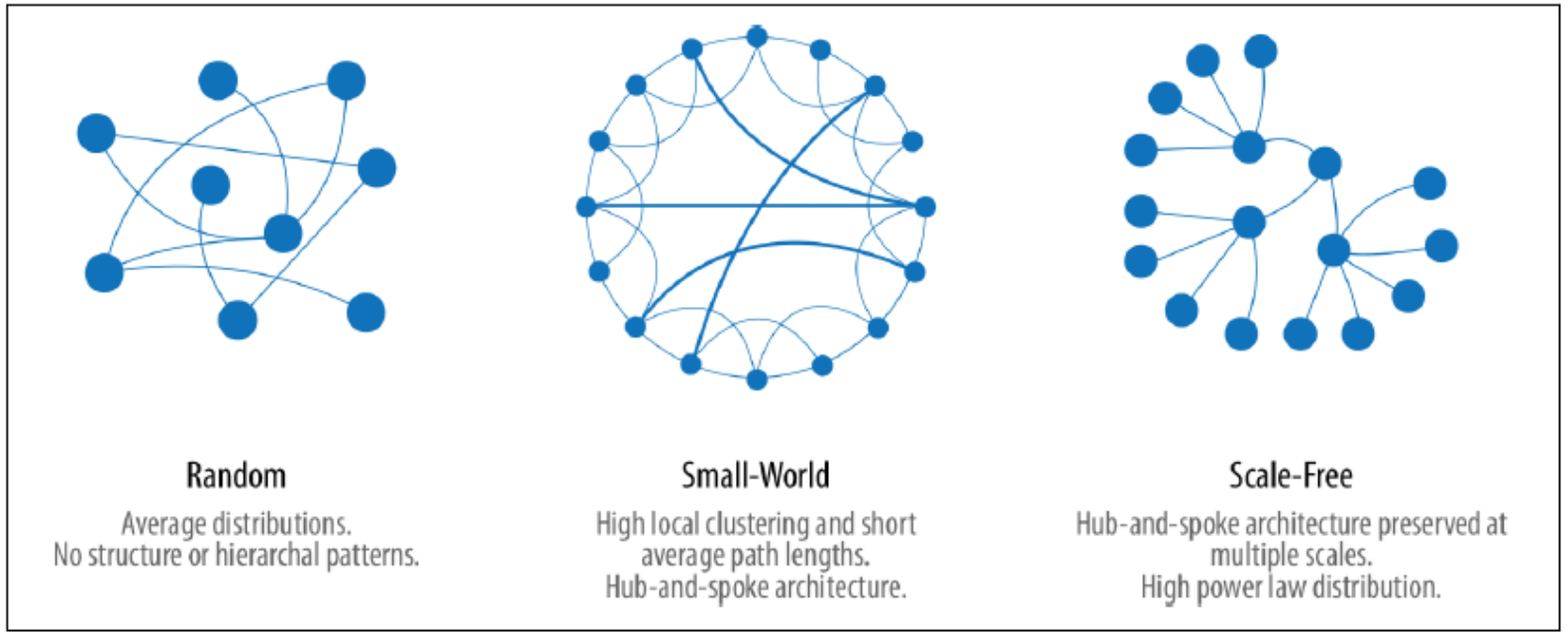


Graph Algorithms Mark Needham and Amy E. Hodler, O'Reilly 2019

Centrality Algorithms

Average Degree of a Network

These measures are used to categorize network types such as the **scale-free** or **small-world** networks. They also provide a quick measure to help estimate the potential for things to spread or ripple throughout a network.



Graph Algorithms Mark Needham and Amy E. Hodler, O'Reilly 2019

Centrality Algorithms

When to Use Degree Centrality

Use Degree Centrality if you're attempting to analyse influence by looking at the number of incoming and outgoing relationships or find the "popularity" of individual nodes. It works well when you're concerned with immediate connectedness or near-term probabilities. However, Degree Centrality is also applied to global analysis when you want to evaluate the minimum degree, maximum degree, mean degree, and standard deviation across the entire graph.

Centrality Algorithms

Degree Centrality - Example Use Cases

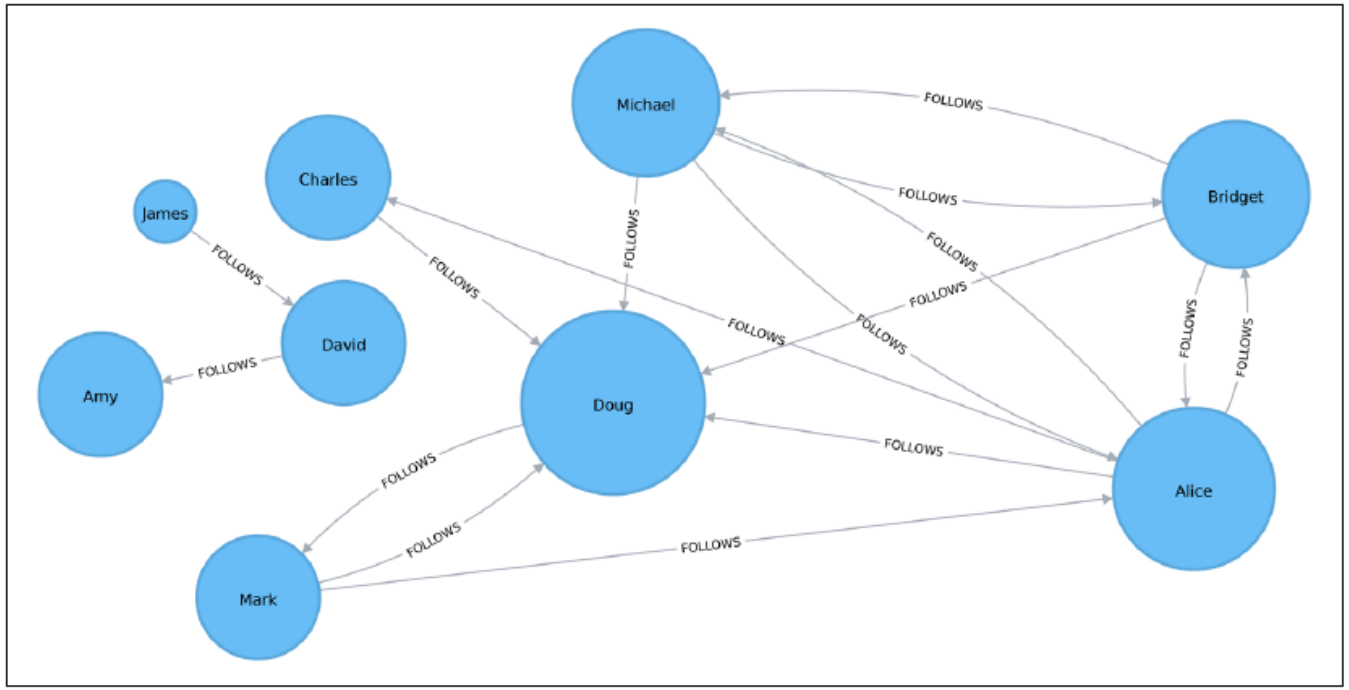
- Identifying powerful individuals though their relationships, such as connections of people in a social network. For example, in BrandWatch's "Most Influential Men and Women on Twitter 2017", the top 5 people in each category have over 40 million followers each. https://www.brandwatch.com/blog/react-influential-men-and-women-2017/

- Separating fraudsters from legitimate users of an online auction site. The weighted centrality of fraudsters tends to be significantly higher due to collusion aimed at artificially increasing prices. Read more in the paper by P. Bangcharoensap et al., "Two Step Graph-Based Semi-Supervised Learning for Online Auction Fraud Detection".

Centrality Algorithms

Degree Centrality - Example Use Cases

Doug is the most popular user in our Twitter graph, with five followers (in-links). All other users in that part of the graph follow him and he only follows one person back. In the real Twitter network, celebrities have high follower counts but tend to follow few people. We could therefore consider Doug a celebrity!



Centrality Algorithms

Closeness Centrality

Closeness Centrality is a way of detecting nodes that are able to spread information efficiently through a subgraph. The measure of a node's centrality is its average farness (inverse distance) to all other nodes. Nodes with a high closeness score have the shortest distances from all other nodes. For each node, the Closeness Centrality algorithm calculates the sum of its distances to all other nodes, based on calculating the shortest paths between all pairs of nodes. The resulting sum is then *inverted* to determine the closeness centrality score for that node.

Centrality Algorithms

Closeness Centrality

The closeness centrality of a node is calculated using the formula:

$$C(u) = \frac{1}{\sum_{v=1}^{n-1} d(u,v)}$$

where u is a node; n is the number of nodes in the graph; $d(u, v)$ is the shortest-path distance between another node v and u. It is more common to normalize this score so that it represents the average length of the shortest paths rather than their sum. This adjustment allows comparisons of the closeness centrality of nodes of graphs of different sizes. The formula for normalized closeness centrality is as follows:

$$C_{norm}(u) = \frac{n-1}{\sum_{v=1}^{n-1} d(u,v)}$$

Centrality Algorithms

When to Use Closeness Centrality?

Apply Closeness Centrality when you need to know which nodes disseminate things the fastest. Using weighted relationships can be especially helpful in evaluating interaction speeds in communication and behavioural analyses.

Centrality Algorithms

Closeness Centrality – Example Use Cases

- Uncovering individuals in very favourable positions to control and acquire vital information and resources within an organization. One such study is "Mapping Networks of Terrorist Cells", by V. E. Krebs.

- As a heuristic for estimating arrival time in telecommunications and package delivery, where content flows through the shortest paths to a predefined target. It is also used to shed light on propagation through all shortest paths simultaneously, such as infections spreading through a local community. Find more details in "Centrality and Network Flow", by S. P. Borgatti.

- Evaluating the importance of words in a document, based on a graph-based key phrase extraction process. This process is described by F. Boudin in "A Comparison of Centrality Measures for Graph-Based Keyphrase Extraction".

Centrality Algorithms

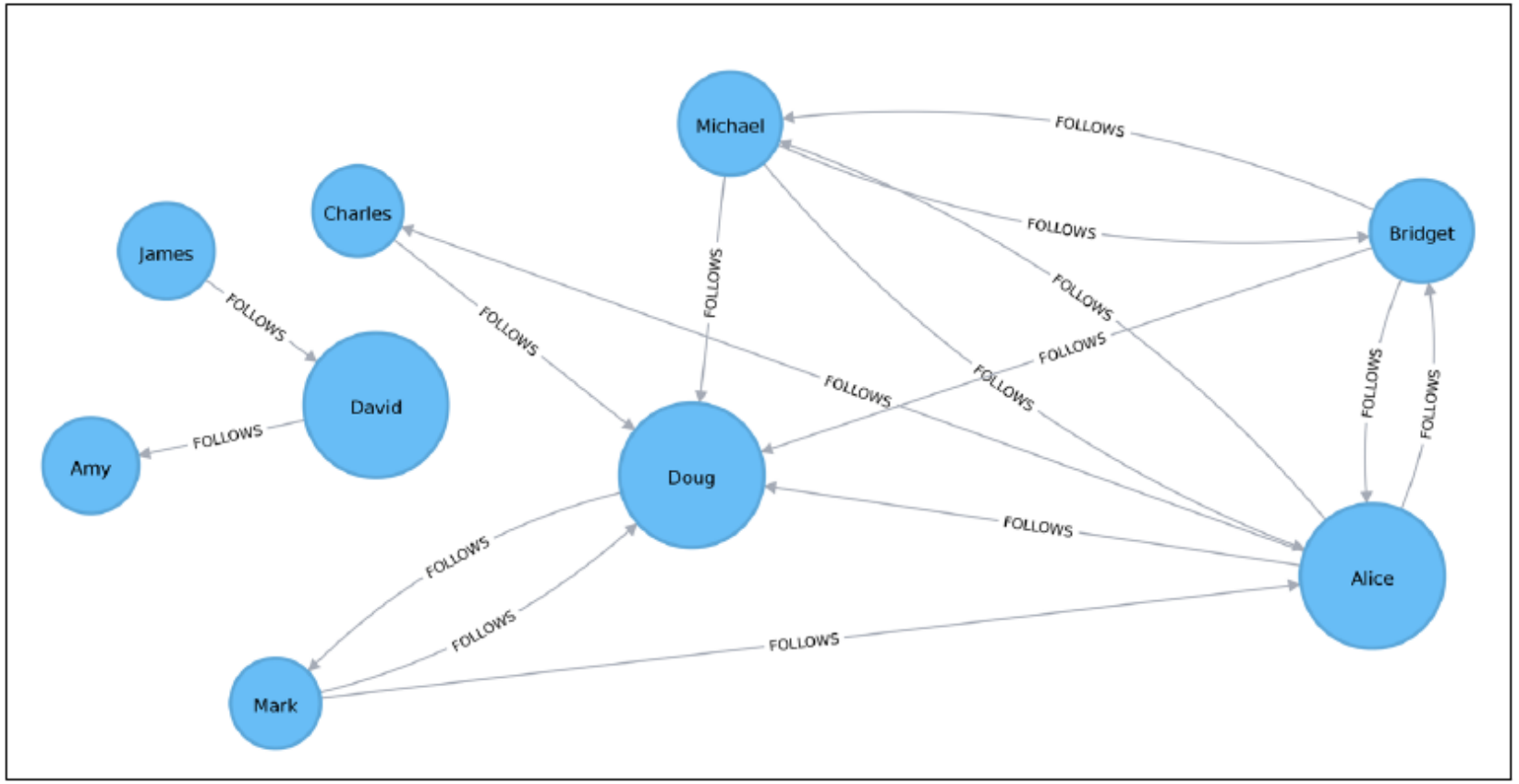
Closeness Centrality – Connected Graphs

Closeness Centrality works best on connected graphs. When the original formula is applied to an unconnected graph, we end up with an infinite distance between two nodes where there is no path between them and an infinite closeness centrality score when we sum up all the distances from that node. To avoid this issue, a variation on the original formula is available. Figure (next slide) illustrates that even though David has only a few connections, within his group of friends that's significant. In other words, this score represents the closeness of each user to others within their subgraph but not the entire graph. Neo4j's implementation of Closeness Centrality uses the formula: $C(u) = \frac{1}{\sum_{v=1}^{n-1} d(u,v)}$

where u is a node; $\boldsymbol{n}$ **is number of nodes in the same component (subgraph or group) as** ***u***, $d(u,v)$ is the shortest-path distance between another node v and u.

Centrality Algorithms

Closeness Centrality – Connected Graphs



Graph Algorithms Mark Needham and Amy E. Hodler, O'Reilly 2019

Centrality Algorithms

Closeness Centrality – Variations

The Closeness Centrality score represents closeness to others within their subgraph but not the entire graph. In the strict interpretation of the Closeness Centrality algorithm, all the nodes in our graph would have a score of ∞ because every node has at least one other node that it's unable to reach. However, it's usually more useful to implement the score per component. Ideally, we'd like to get an indication of closeness across the whole graph. **Wasserman Faust** and **Harmonic Centrality** are two such variations.

Centrality Algorithms

Closeness Centrality Variations – Wasserman Faust

Stanley Wasserman and Katherine Faust came up with an improved formula for calculating closeness for graphs with multiple subgraphs without connections between those groups. Details on their formula are in their book, *Social Network Analysis: Methods and Applications*. The result of this formula is a ratio of the fraction of nodes in the group that are reachable to the average distance from the reachable nodes. The formula is as follows:

$$C_{WF}(u) = \frac{n-1}{N-1}\frac{n-1}{\sum_{v=1}^{n-1} d(u,v)}$$

where u is a node; N is the total node count, n is the number of nodes in the same component as u, $d(u,v)$ is the shortest-path distance between another node v and u. (Neo4j parameter **improved:true**)

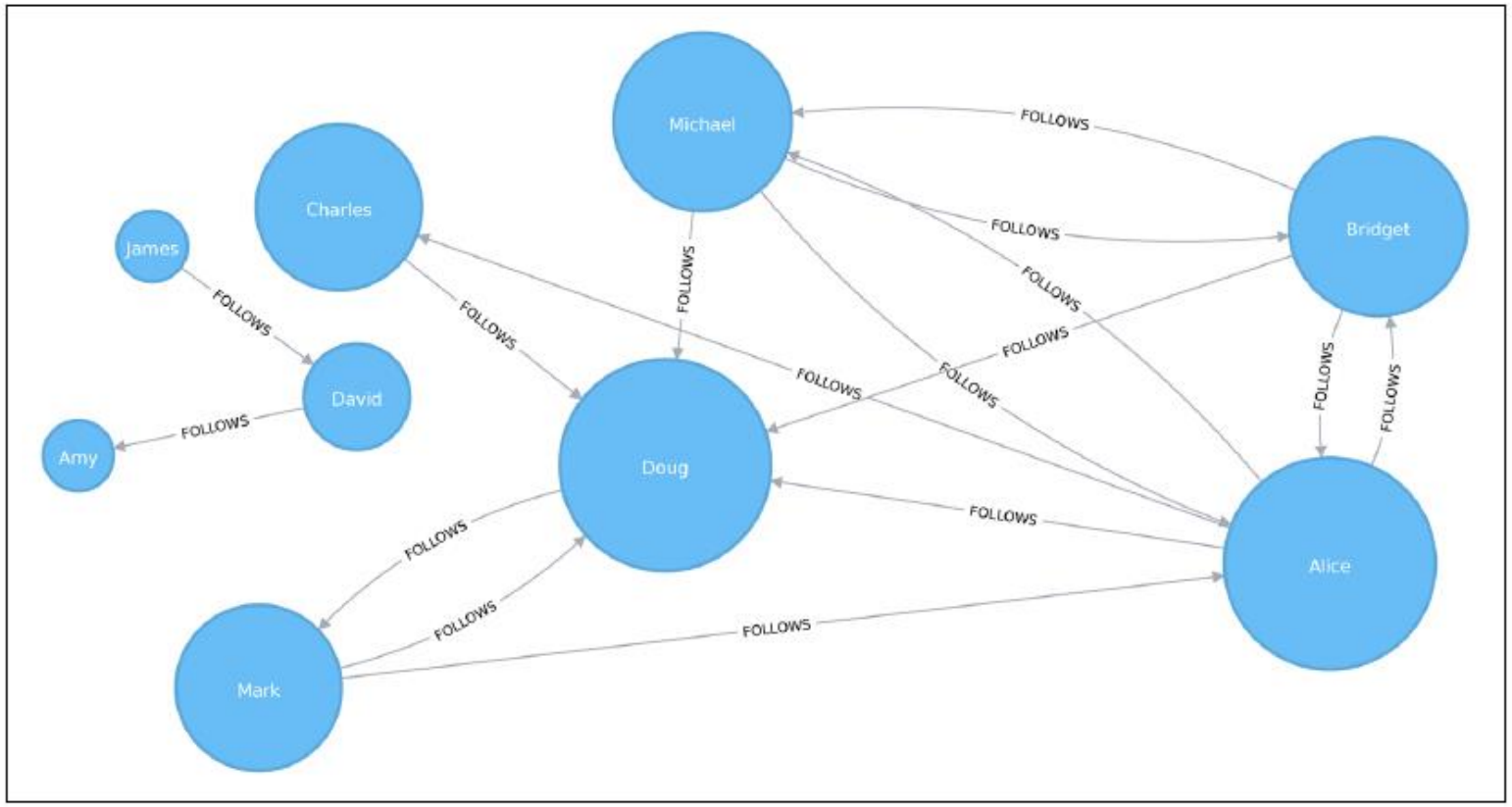
Centrality Algorithms

Closeness Centrality Variations – Wasserman Faust

As Figure shows (next slide), the results are now more representative of the closeness of nodes to the entire graph. The scores for the members of the smaller subgraph (David, Amy, and James) have been dampened, and they now have the lowest scores of all users. This makes sense as they are the most isolated nodes. This formula is more useful for detecting the importance of a node across the entire graph rather than within its own subgraph.

Centrality Algorithms

Closeness Centrality Variations – Wasserman Faust



Graph Algorithms Mark Needham and Amy E. Hodler, O'Reilly 2019

Centrality Algorithms

Closeness Centrality Variations – Harmonic Centrality

Harmonic Centrality (also known as Valued Centrality) is a variant of Closeness Centrality, invented to solve the original problem with unconnected graphs. In "Harmony in a Small World", M. Marchiori and V. Latora proposed this concept as a practical representation of an average shortest path.

When calculating the closeness score for each node, rather than summing the distances of a node to all other nodes, it sums the inverse of those distances. This means that infinite values become irrelevant and are handled cleanly.

Centrality Algorithms

Closeness Centrality Variations – Harmonic Centrality

The raw harmonic centrality for a node is calculated using the following formula:

$$H(u) = \sum_{v=1}^{n-1} \frac{1}{d(u,v)}$$

where u is a node; n is the number of nodes in the graph; $d(u,v)$ is the shortest-path distance between another node v and u. As with closeness centrality, we can also calculate a normalized harmonic centrality with the following formula:

$$H_{norm}(u) = \frac{\sum_{v=1}^{n-1} \frac{1}{d(u,v)}}{n-1}$$

Centrality Algorithms

Closeness Centrality Variations – Harmonic Centrality

Using the Neo4j implementation, the results from the Harmonic Centrality algorithm differ from those of the original Closeness Centrality algorithm but are similar to those from the Wasserman and Faust improvement. Either algorithm can be used when working with graphs with more than one connected component.

Centrality Algorithms

Betweenness Centrality

Sometimes the most important cog in the system is not the one with the most overt power or the highest status. Sometimes it's the middlemen that connect groups or the brokers who the most control over resources or the flow of information. Betweenness Centrality is a way of detecting the amount of influence a node has over the flow of information or resources in a graph. It is typically used to find nodes that serve as a bridge from one part of a graph to another.

The Betweenness Centrality algorithm first calculates the shortest (weighted) path between every pair of nodes in a connected graph. Each node receives a score, based on the number of these shortest paths that pass through the node. The more shortest paths that a node lies on, the higher its score. Betweenness Centrality was considered one of the "three distinct intuitive conceptions of centrality" when it was introduced by Linton C. Freeman in his 1971 paper, "A Set of Measures of Centrality Based on Betweenness".

Centrality Algorithms

Betweenness Centrality – Bridges and Control Points

A bridge in a network can be a node or a relationship. In a very simple graph, you can find them by looking for the node or relationship that, if removed, would cause a section of the graph to become disconnected. However, as that's not practical in a typical graph, we use a Betweenness Centrality algorithm. We can also measure the betweenness of a cluster by treating the group as a node. A node is considered *pivotal* for two other nodes if it lies on *every* shortest path between those nodes, as shown in Figure (next slide).

Centrality Algorithms

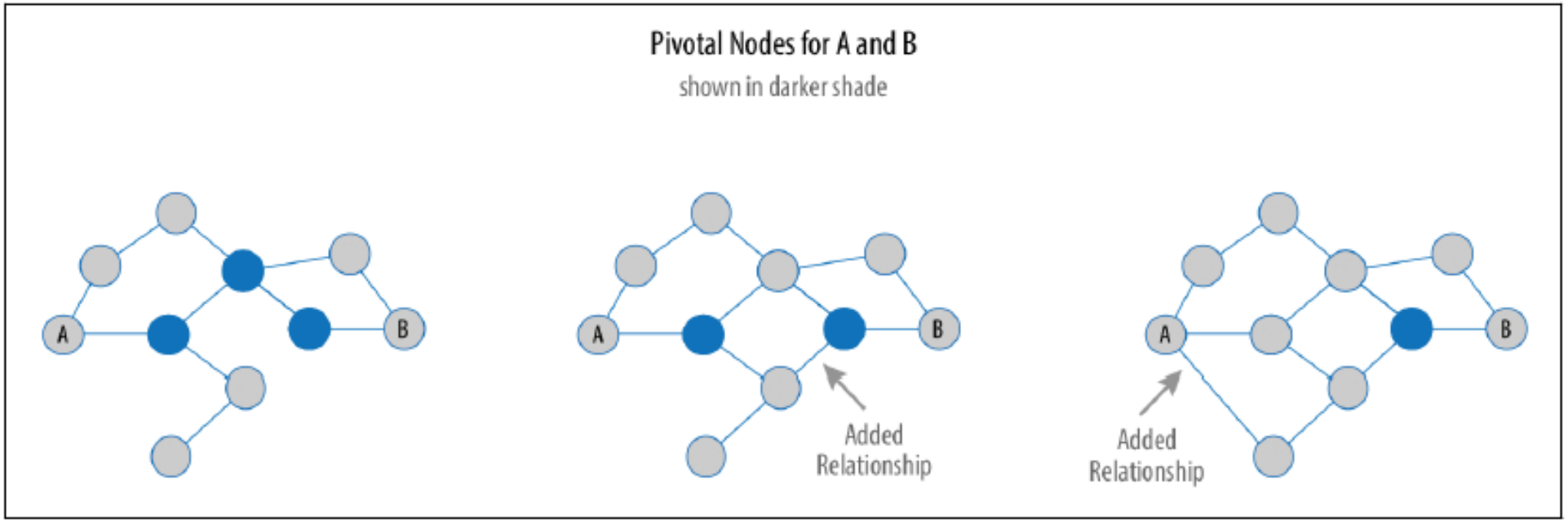
Betweenness Centrality – Bridges and Control Points

A bridge in a network can be a node or a relationship. In a very simple graph, you can find them by looking for the node or relationship that, if removed, would cause a section of the graph to become disconnected. However, as that's not practical in a typical graph, we use a Betweenness Centrality algorithm. We can also measure the betweenness of a cluster by treating the group as a node. A node is considered *pivotal* for two other nodes if it lies on *every* shortest path between those nodes, as shown in Figure (next slide).

Pivotal nodes lie on every shortest path between two nodes. Creating more shortest paths can reduce the number of pivotal nodes for uses such as risk mitigation. Pivotal nodes play an important role in connecting other nodes—if you remove a pivotal node, the new shortest path for the original node pairs will be longer or more costly. This can be a consideration for evaluating single points of vulnerability.

Centrality Algorithms

Betweenness Centrality – Bridges and Control Points



Graph Algorithms Mark Needham and Amy E. Hodler, O'Reilly 2019

Centrality Algorithms

Calculating Betweenness Centrality

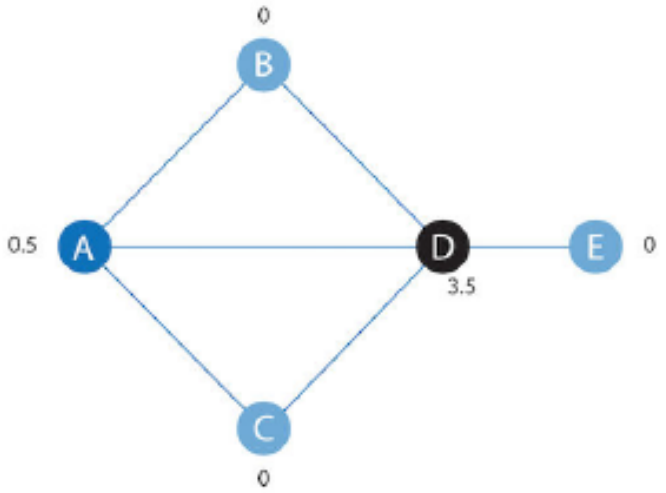
The betweenness centrality of a node is calculated by adding the results of the following formula for all shortest paths:

$$B(u) = \sum_{s \neq u \neq t} \frac{p(u)}{p}$$

where u is a node, p is the total number of shortest paths between nodes s and t, $p(u)$ is the number of shortest paths between nodes s and t that pass through node u.

Centrality Algorithms

Calculating Betweenness Centrality



Node D Calculation

Pairs with Shortest Paths Through D	Total Possible Shortest Paths for That Pair	% of Total Through D (1/Total)
A, E	1	1
B, E	1	1
C, E	1	1
B, C	2 (through D & A)	0.5
Betweenness Score		3.5

Centrality Algorithms

Calculating Betweenness Centrality - Procedure

1. For each node, find the shortest paths that go through it. a. B, C, E have no shortest paths and are assigned a value of 0.

2. For each shortest path in step 1, calculate its percentage of the total possible shortest paths for that pair.

3. Add together all the values in step 2 to find a node's betweenness centrality score. The table in Figure (previous slide) illustrates steps 2 and 3 for node D.

4. Repeat the process for each node.

Centrality Algorithms

When to Use Betweenness Centrality

Betweenness Centrality applies to a wide range of problems in real-world networks. We use it to find bottlenecks, control points, and vulnerabilities.

Note:-
Betweenness Centrality makes the assumption that all communication between nodes happens along the shortest path and with the same frequency, which isn't always the case in real life. Therefore, it doesn't give us a perfect view of the most influential nodes in a graph, but rather a good representation. Mark Newman explains this in more detail on p. 186 of *Networks: An Introduction* (Oxford University Press).

Centrality Algorithms

Betweenness Centrality – Example Use Cases

- Identifying influencers in various organizations. Powerful individuals are not necessarily in management positions, but can be found in “brokerage positions” using Betweenness Centrality. Removal of such influencers can seriously destabilize the organization. This might be considered a welcome disruption by law enforcement if the organization is criminal, or could be a disaster if a business loses key staff it underestimated. More details are found in “Brokerage Qualifications in Ringing Operations”, by C. Morselli and J. Roy.

- Uncovering key transfer points in networks such as electrical grids. Counterintuitively, removal of specific bridges can actually *improve* overall robustness by “islanding” disturbances. Research details are included in “Robustness of the European Power Grids Under Intentional Attack”, by R. Sole, et al.

Centrality Algorithms

Betweenness Centrality – Example Use Cases

- Helping microbloggers spread their reach on Twitter, with a recommendation engine for targeting influencers. This approach is described in a paper by S. Wu et al., "Making Recommendations in a Microblog to Improve the Impact of a Focal User".

Centrality Algorithms

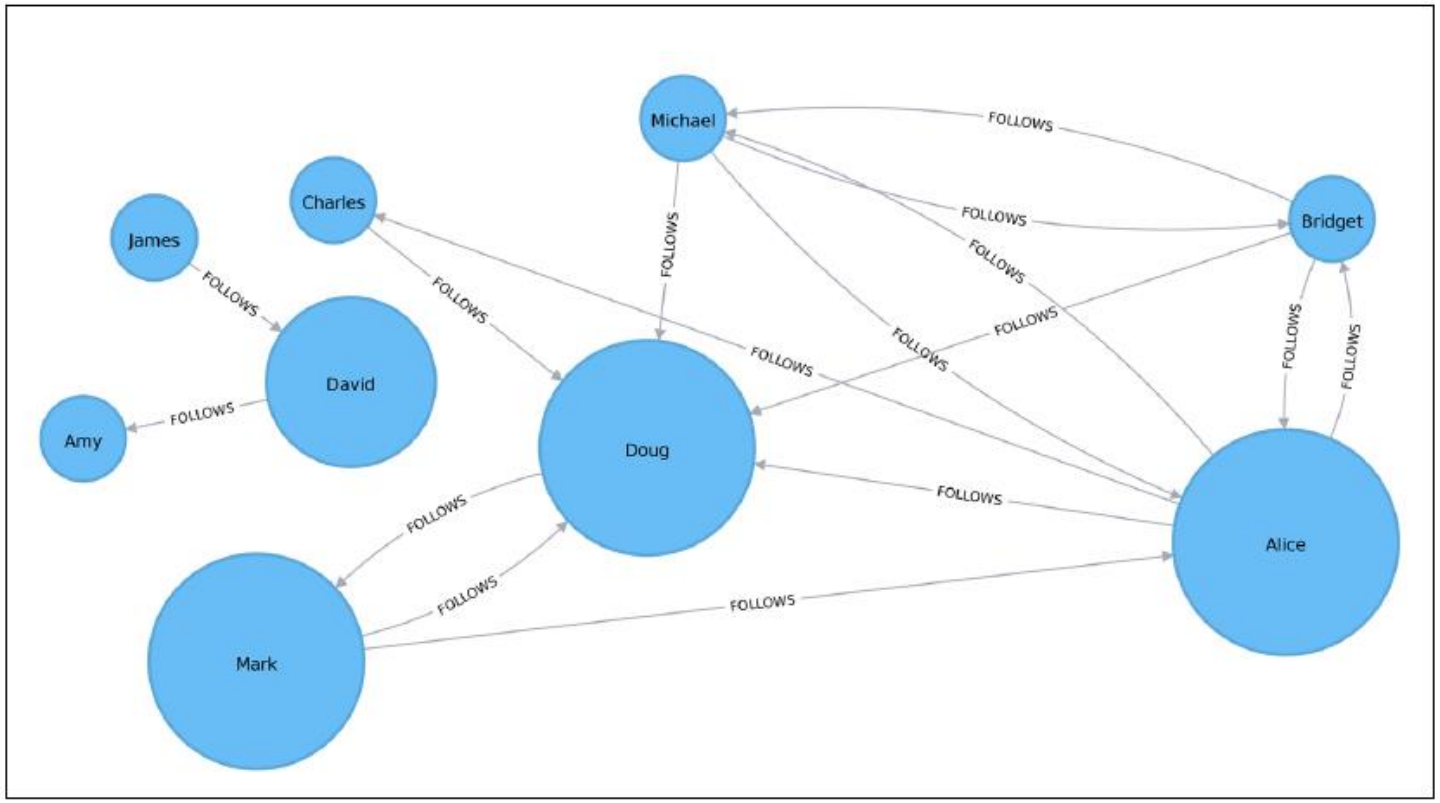
Betweenness Centrality – Neo4j

As we can see in Figure (next slide), Alice is the main broker in this network, but Mark and Doug aren't far behind. In the smaller subgraph all shortest paths go through David, so he is important for information flow among those nodes.

For large graphs, exact centrality computation isn't practical. The fastest known algorithm for exactly computing betweenness of all the nodes has a runtime proportional to the product of the number of nodes and the number of relationships. We may want to filter down to a subgraph first that works with a subset of nodes.

Centrality Algorithms

Betweenness Centrality – Neo4j



Graph Algorithms Mark Needham and Amy E. Hodler, O'Reilly 2019

Centrality Algorithms

Betweenness Centrality Variation – RA Brandes

Recall that calculating the exact betweenness centrality on large graphs can be very expensive. We could therefore choose to use an approximation algorithm that runs much faster but still provides useful (albeit imprecise) information. The Randomized-Approximate Brandes (RA-Brandes for short) algorithm is the best-known algorithm for calculating an approximate score for betweenness centrality. Rather than calculating the shortest path between every pair of nodes, the RABrandes algorithm considers only a subset of nodes.

Due to the random nature of this algorithm, we may see different results each time that we run it. On larger graphs this randomness will have less of an impact than it does on a small sample graph.

Centrality Algorithms

Betweenness Centrality Variation – RA Brandes

Two common strategies for selecting the subset of nodes are:

Random

Nodes are selected uniformly, at random, with a defined probability of selection. The default probability is: $\frac{log_{10}(N)}{e^2}$. If the probability is 1, the algorithm works the same way as the normal Betweenness Centrality algorithm, where all nodes are loaded.

Degree

Nodes are selected randomly, but those whose degree is lower than the mean are automatically excluded (i.e., only nodes with a lot of relationships have a chance of being visited). As a further optimization, you could limit the depth used by the Shortest Path algorithm, which will then provide a subset of all the shortest paths.

Centrality Algorithms

PageRank

PageRank is the best known of the centrality algorithms. It measures the transitive (or directional) influence of nodes. All the other centrality algorithms we discuss measure the direct influence of a node, whereas PageRank considers the influence of a node's neighbours, and their neighbours.

For example, having a few very powerful friends can make you more influential than having a lot of less powerful friends. PageRank is computed either by iteratively distributing one node's rank over its neighbours or by randomly traversing the graph and counting the frequency with which each node is hit during these walks.

Centrality Algorithms

PageRank

PageRank is named after Google cofounder Larry Page, who created it to rank websites in Google's search results.

The basic assumption is that a page with more incoming and more influential incoming links is more likely a credible source. PageRank measures the number and quality of incoming relationships to a node to determine an estimation of how important that node is. Nodes with more sway over a network are presumed to have more incoming relationships from other influential nodes.

Centrality Algorithms

PageRank - Influence

The intuition behind influence is that relationships to more important nodes contribute more to the influence of the node in question than equivalent connections to less important nodes. Measuring influence usually involves scoring nodes, often with weighted relationships, and then updating the scores over many iterations. Sometimes all nodes are scored, and sometimes a random selection is used as a representative distribution.

Note:

Keep in mind that centrality measures represent the importance of a node in comparison to other nodes. Centrality is a ranking of the potential impact of nodes, not a measure of actual impact. For example, you might identify the two people with the highest centrality in a network, but perhaps policies or cultural norms are in play that actually shift influence to others. Quantifying actual impact is an active research area to develop additional influence metrics.

Centrality Algorithms

PageRank Formula

PageRank is defined in the original Google paper as follows:

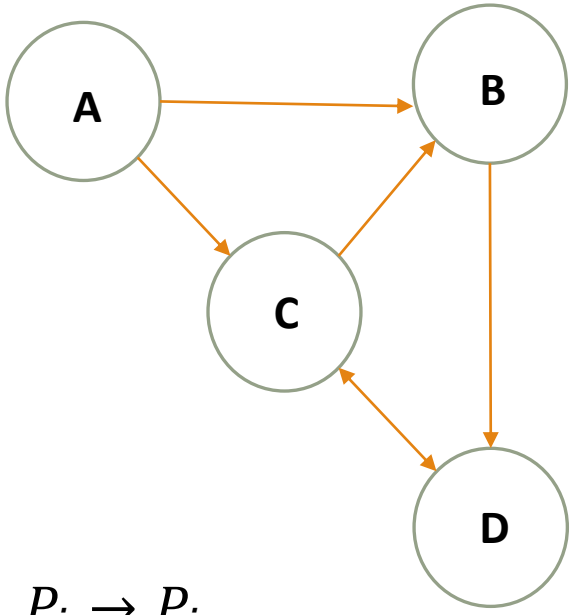
$$PR(u) = (1 - d) + d\left(\frac{PR(T1)}{C(T1)} + \cdots + \frac{PR(Tn)}{C(Tn)}\right)$$

where:

- We assume that a page u has citations from pages $T1$ to Tn
- d is a damping factor which is set between 0 and 1. It is usually set to 0.85. You
- can think of this as the probability that a user will continue clicking. This helps
- minimize rank sink, explained in the next section.
- $1 - d$ is the probability that a node is reached directly without following any relationships.
- $C(Tn)$ is defined as the out-degree of a node T.

Centrality Algorithms

PageRank Formula – Worked Example – Iteration 1



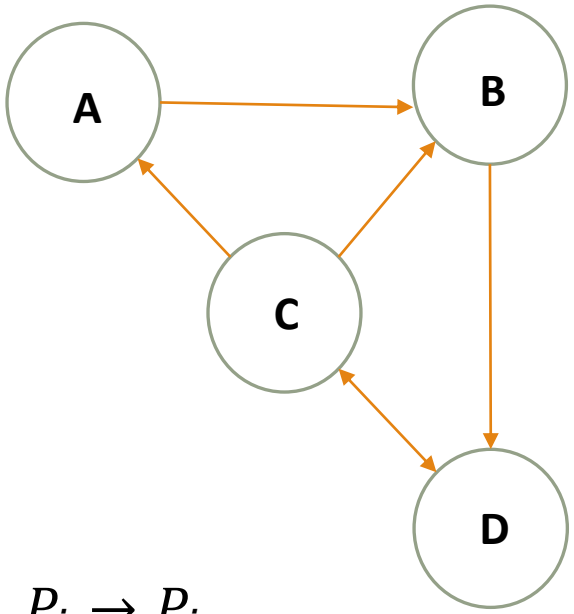
$P_i \rightarrow P_j$

	Iteration 0	Iteration 1	Iteration 2	PageRank
A	1/4			
B	1/4			
C	1/4			
D	1/4			

$$PR_{t+1}(P_i) = \frac{\sum_j PR_t(P_j)}{C(P_j)}$$

Centrality Algorithms

PageRank Formula – Worked Example – Iteration 1



$P_i \rightarrow P_j$

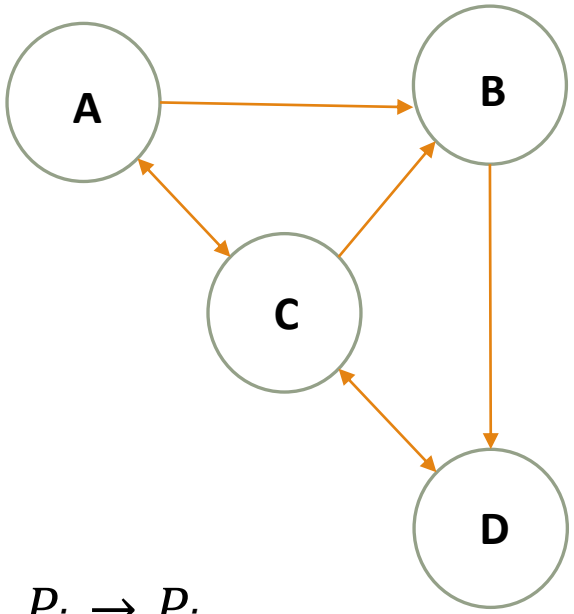
	Iteration 0	Iteration 1	Iteration 2	PageRank
A	1/4	1/12		
B	1/4			
C	1/4			
D	1/4			

$$PR_{t+1}(P_i) = \frac{\sum_j PR_t(P_j)}{C(P_j)}$$

$$\boldsymbol{PR(A) = \frac{1/4}{3} = \frac{1}{12}}$$

Centrality Algorithms

PageRank Formula – Worked Example – Iteration 1



	Iteration 0	Iteration 1	Iteration 2	PageRank
A	1/4	1/12		
B	1/4	2.5/12		
C	1/4			
D	1/4			

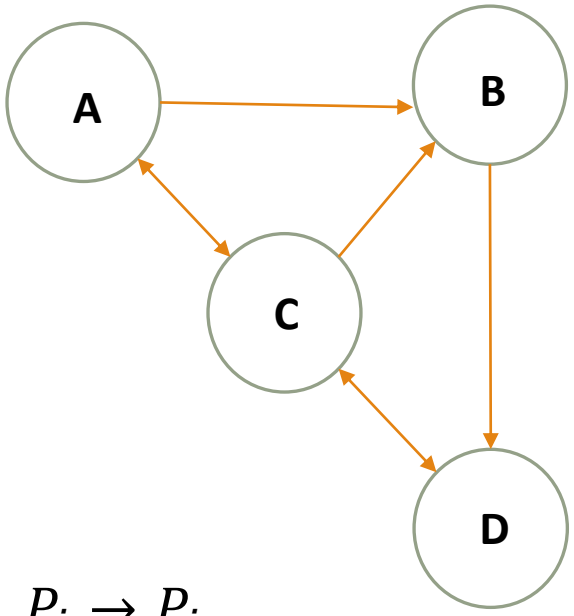
$P_i \rightarrow P_j$

$$PR_{t+1}(P_i) = \frac{\sum_j PR_t(P_j)}{C(P_j)}$$

$$\boldsymbol{PR(B) = \frac{1/4}{2} + \frac{1/4}{3} = \frac{1}{12}}$$

Centrality Algorithms

PageRank Formula – Worked Example – Iteration 1



	Iteration 0	Iteration 1	Iteration 2	PageRank
A	1/4	1/12		
B	1/4	2.5/12		
C	1/4	4.5/12		
D	1/4			

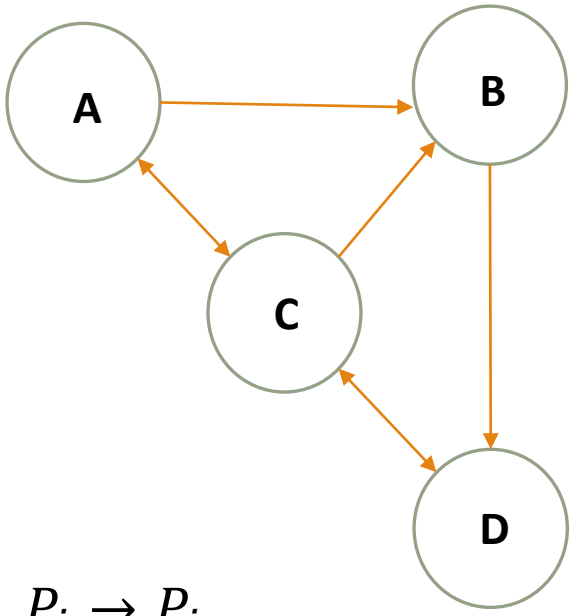
$P_i \rightarrow P_j$

$$PR_{t+1}(P_i) = \frac{\sum_j PR_t(P_j)}{C(P_j)}$$

$$\boldsymbol{PR(C) = \frac{1/4}{2} + \frac{1/4}{1} = \frac{4.5}{12}}$$

Centrality Algorithms

PageRank Formula – Worked Example – Iteration 1



$P_i \rightarrow P_j$

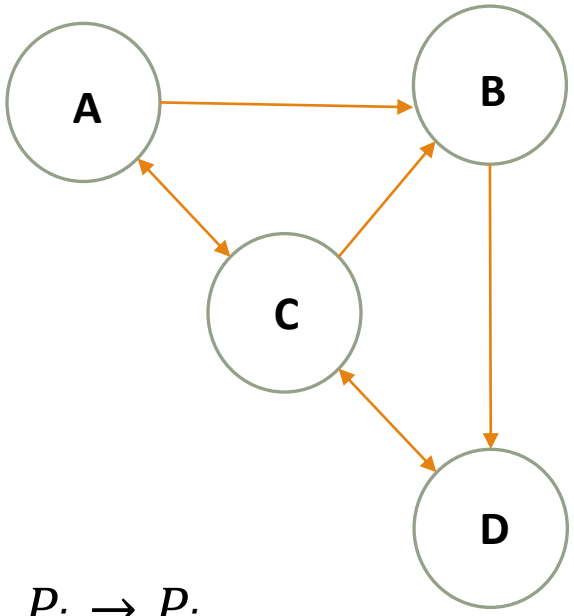
	Iteration 0	Iteration 1	Iteration 2	PageRank
A	1/4	1/12		
B	1/4	2.5/12		
C	1/4	4.5/12		
D	1/4	4/12		

$$PR_{t+1}(P_i) = \frac{\sum_j PR_t(P_j)}{C(P_j)}$$

$$\boldsymbol{PR(D) = \frac{1/4}{3} + \frac{1/4}{1} = \frac{4}{12}}$$

Centrality Algorithms

PageRank Formula – Worked Example – Iteration 2



	Iteration 0	Iteration 1	Iteration 2	PageRank
A	1/4	1/12	1.5/12	
B	1/4	2.5/12		
C	1/4	4.5/12		
D	1/4	4/12		

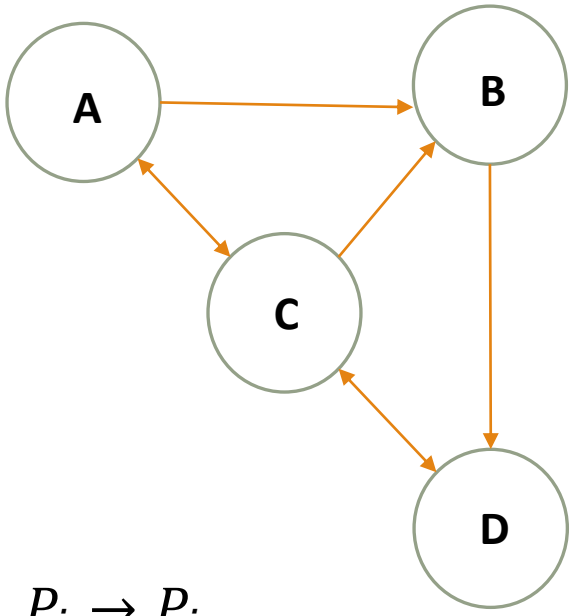
$P_i \rightarrow P_j$

$$PR_{t+1}(P_i) = \frac{\sum_j PR_t(P_j)}{C(P_j)}$$

$$\boldsymbol{PR(A) = \frac{4.5/12}{3} = \frac{1.5}{12}}$$

Centrality Algorithms

PageRank Formula – Worked Example – Iteration 2



	Iteration 0	Iteration 1	Iteration 2	PageRank
A	1/4	1/12	1.5/12	
B	1/4	2.5/12	2/12	
C	1/4	4.5/12		
D	1/4	4/12		

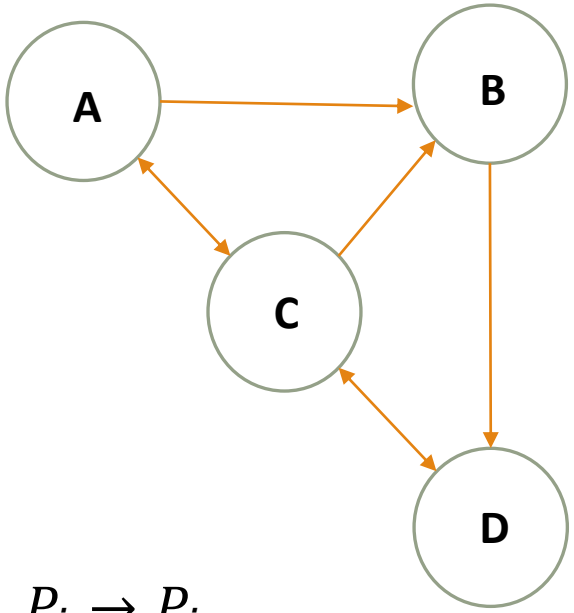
$P_i \rightarrow P_j$

$$PR_{t+1}(P_i) = \frac{\sum_j PR_t(P_j)}{C(P_j)}$$

$$\boldsymbol{PR(B) = \frac{1/12}{2} + \frac{4.5/12}{3} = \frac{2}{12}}$$

Centrality Algorithms

PageRank Formula – Worked Example – Iteration 2



	Iteration 0	Iteration 1	Iteration 2	PageRank
A	1/4	1/12	1.5/12	
B	1/4	2.5/12	2/12	
C	1/4	4.5/12	4.5/12	
D	1/4	4/12		

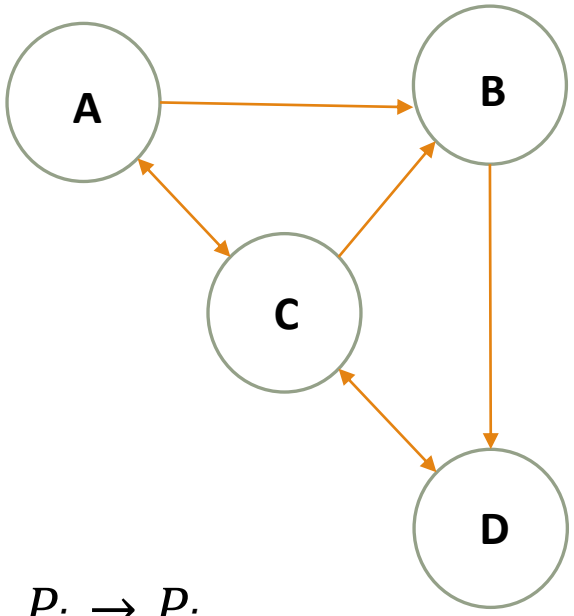
$P_i \rightarrow P_j$

$$PR_{t+1}(P_i) = \frac{\sum_j PR_t(P_j)}{C(P_j)}$$

$$\boldsymbol{PR(C) = \frac{1/12}{2} + \frac{4/12}{1} = \frac{4.5}{12}}$$

Centrality Algorithms

PageRank Formula – Worked Example – Iteration 2



	Iteration 0	Iteration 1	Iteration 2	PageRank
A	1/4	1/12	1.5/12	
B	1/4	2.5/12	2/12	
C	1/4	4.5/12	4.5/12	
D	1/4	4/12	4/12	

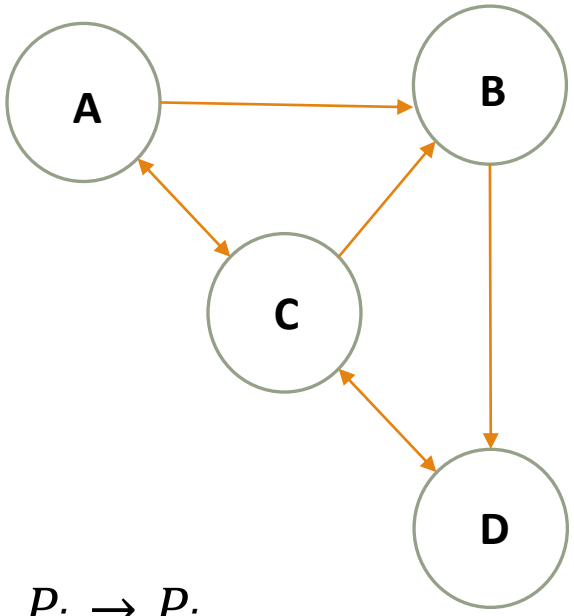
$P_i \rightarrow P_j$

$$PR_{t+1}(P_i) = \frac{\sum_j PR_t(P_j)}{C(P_j)}$$

$$\boldsymbol{PR(D) = \frac{4.5/12}{3} + \frac{2.5/12}{1} = \frac{4}{12}}$$

Centrality Algorithms

PageRank Formula – Worked Example – PageRank



$P_i \rightarrow P_j$

	Iteration 0	Iteration 1	Iteration 2	PageRank
A	1/4	1/12	1.5/12	1
B	1/4	2.5/12	2/12	2
C	1/4	4.5/12	4.5/12	4
D	1/4	4/12	4/12	3

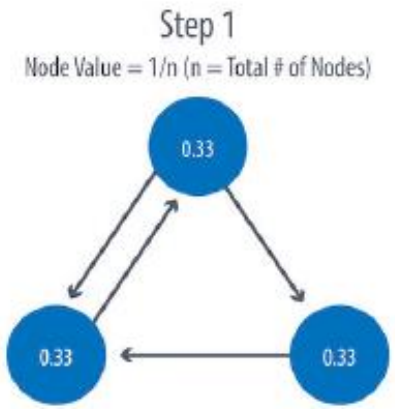
$$PR_{t+1}(P_i) = \frac{\sum_j PR_t(P_j)}{C(P_j)}$$

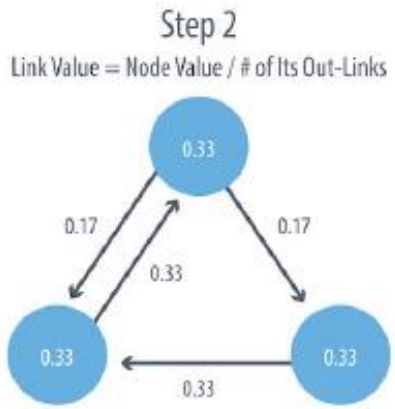
Centrality Algorithms

PageRank Walk Through

The following slides walk through a small example of how PageRank will continue to update the rank of a node until it converges or meets the set number of iterations.

Each iteration of PageRank has two calculation steps: one to update node values and one to update link values.

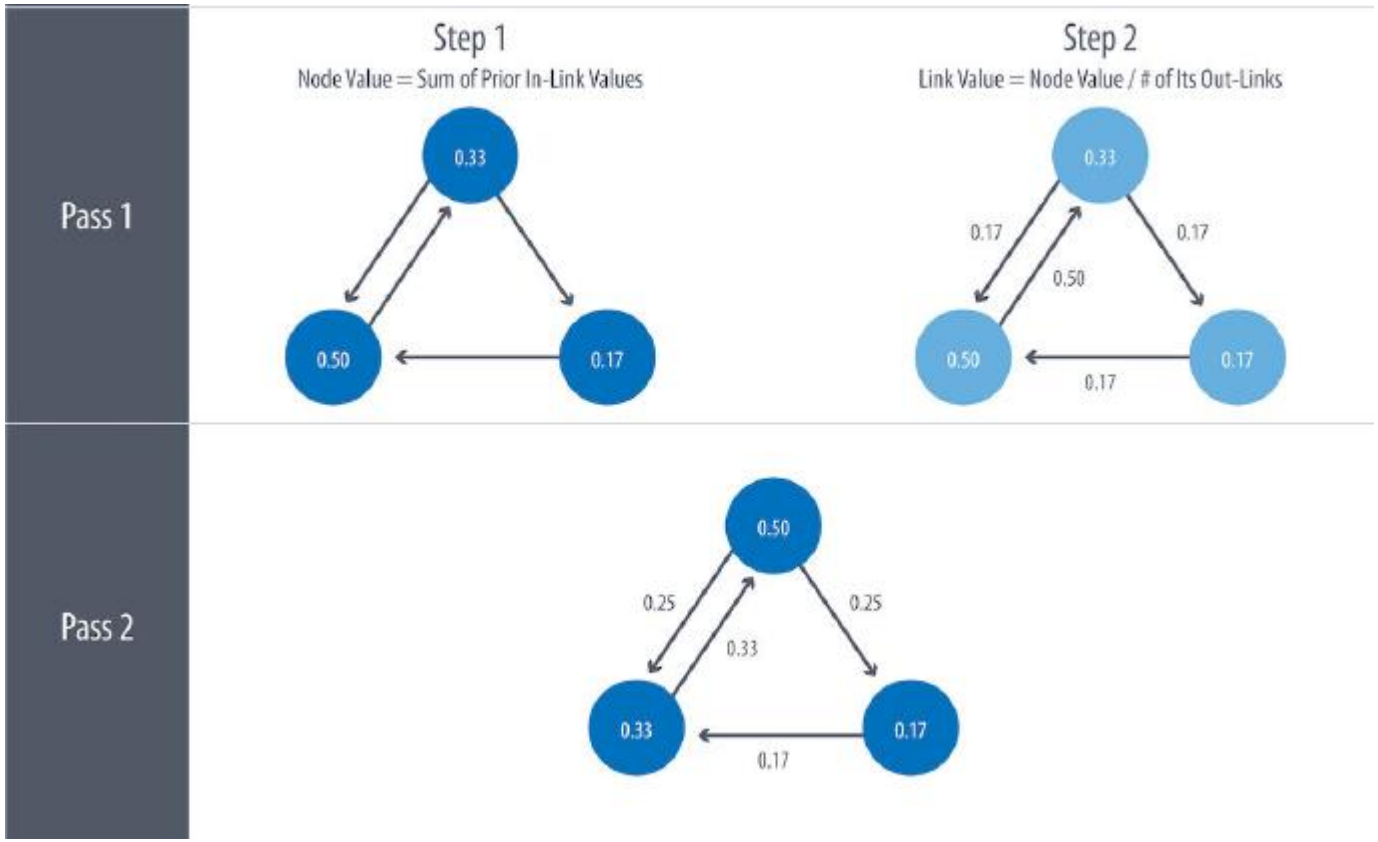




Graph Algorithms Mark Needham and Amy E. Hodler, O'Reilly 2019

Centrality Algorithms

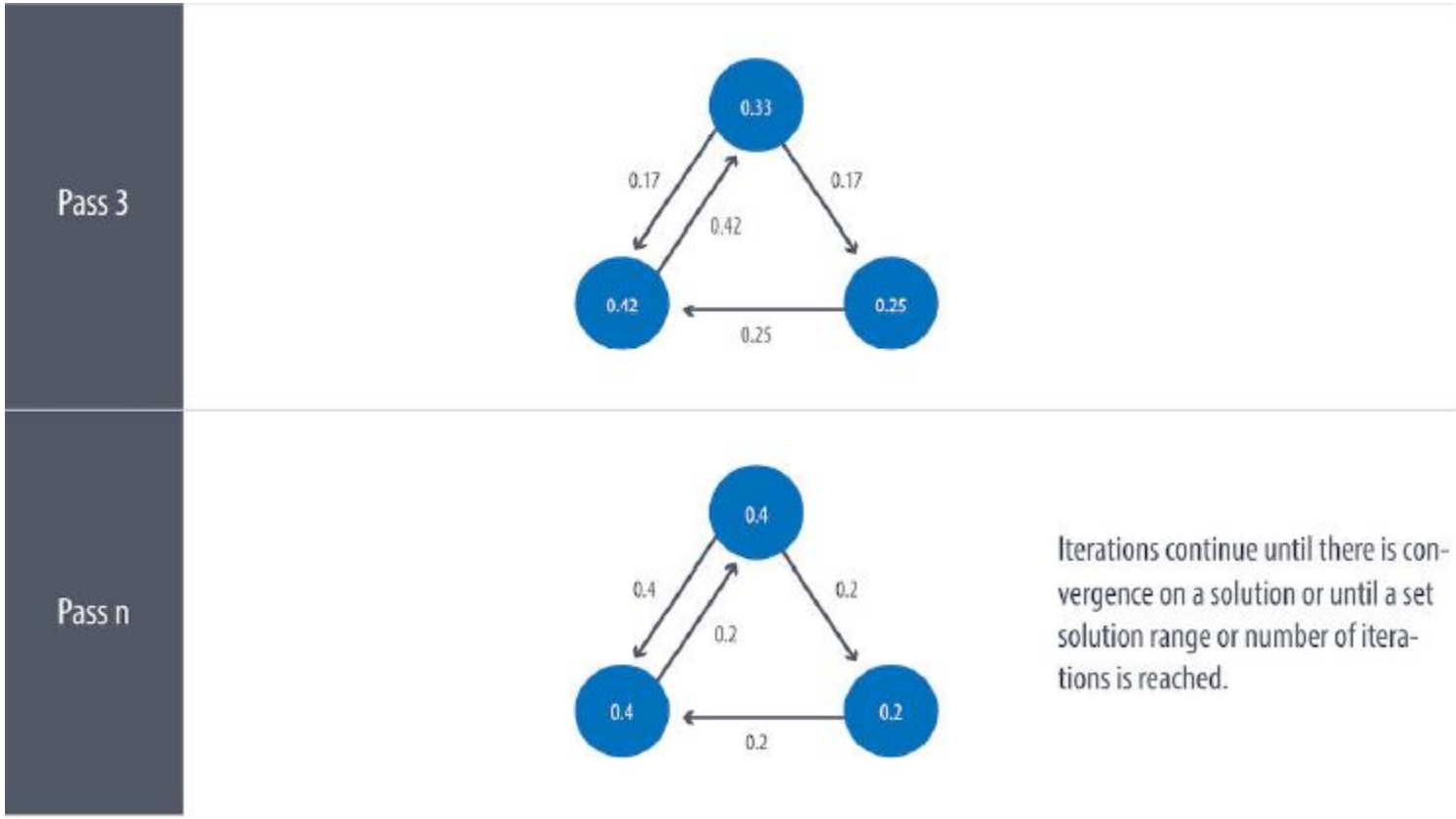
PageRank Walk Through



Graph Algorithms Mark Needham and Amy E. Hodler, O'Reilly 2019

Centrality Algorithms

PageRank Walk Through



Iterations continue until there is convergence on a solution or until a set solution range or number of iterations is reached.

Graph Algorithms Mark Needham and Amy E. Hodler, O'Reilly 2019

Centrality Algorithms

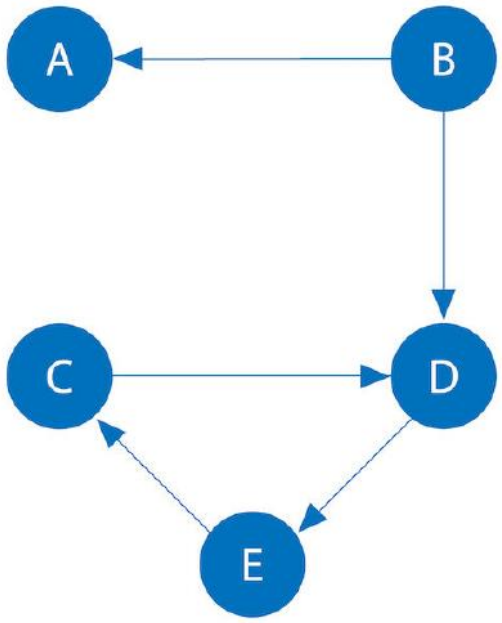
Iteration, Random Surfers, and Rank Sinks

PageRank is an iterative algorithm that runs either until scores converge or until a set number of iterations is reached. Conceptually, PageRank assumes there is a web surfer visiting pages by following links or by using a random URL. A damping factor _d _ defines the probability that the next click will be through a link. You can think of it as the probability that a surfer will become bored and randomly switch to another page. A PageRank score represents the likelihood that a page is visited through an incoming link and not randomly. A node, or group of nodes, without outgoing relationships (also called a *dangling node*) can monopolize the PageRank score by refusing to share. This is known as a *rank sink*. You can imagine this as a surfer that gets stuck on a page, or a subset of pages, with no way out. Another difficulty is created by nodes that point only to each other in a group. Circular references cause an increase in their ranks as the surfer bounces back and forth among the nodes.

Centrality Algorithms

Iteration, Random Surfers, and Rank Sinks

Rank Sinks Monopolize Rank Scores



A is a dangling node with no outgoing relationships. Teleportation is used to overcome dead ends.

C, D, and E are circular references with no way out of the group. A dampening factor is used to introduce random node visits.

Centrality Algorithms

Iteration, Random Surfers, and Rank Sinks

There are two strategies used to avoid rank sinks. First, when a node is reached that has no outgoing relationships, PageRank assumes outgoing relationships to all nodes. traversing these invisible links is sometimes called *teleportation*. Second, the damping factor provides another opportunity to avoid sinks by introducing a probability for direct link versus random node visitation. When you set *d* to 0.85, a completely random node is visited 15% of the time.

Centrality Algorithms

Iteration, Random Surfers, and Rank Sinks

Although the original formula recommends a damping factor of 0.85, its initial use was on the World Wide Web with a power-law distribution of links (most pages have very few links and a few pages have many). Lowering the damping factor decreases the likelihood of following long relationship paths before taking a random jump. In turn, this increases the contribution of a node's immediate predecessors to its score and rank. If you see unexpected results from PageRank, it is worth doing some exploratory analysis of the graph to see if any of these problems are the cause.

Read Ian Rogers's article, "The Google PageRank Algorithm and How It Works" to learn more http://ianrogers.uk/google-page-rank/

Centrality Algorithms

When to Use PageRank

PageRank is now used in many domains outside web indexing. Use this algorithm whenever you're looking for broad influence over a network. For instance, if you're looking to target a gene that has the highest overall impact to a biological function, it may not be the most connected one. It may, in fact, be the gene with the most relationships with other, more significant functions.

Centrality Algorithms

PageRank - Example Use Cases

- Presenting users with recommendations of other accounts that they may wish to follow (Twitter uses Personalized PageRank for this). The algorithm is run over a graph that contains shared interests and common connections. The approach is described in more detail in the paper "WTF: The Who to Follow Service at Twitter", by P. Gupta et al.

- Predicting traffic flow and human movement in public spaces or streets. The algorithm is run over a graph of road intersections, where the PageRank score reflects the tendency of people to park, or end their journey, on each street. This is described in more detail in "Self-Organized Natural Roads for Predicting Traffic Flow: A Sensitivity Study", a paper by B. Jiang, S. Zhao, and J. Yin.

Centrality Algorithms

PageRank - Example Use Cases

- As part of anomaly and fraud detection systems in the healthcare and insurance industries. PageRank helps reveal doctors or providers that are behaving in an unusual manner, and the scores are then fed into a machine learning algorithm.

David Gleich describes many more uses for the algorithm in his paper, "PageRank Beyond the Web".

Centrality Algorithms

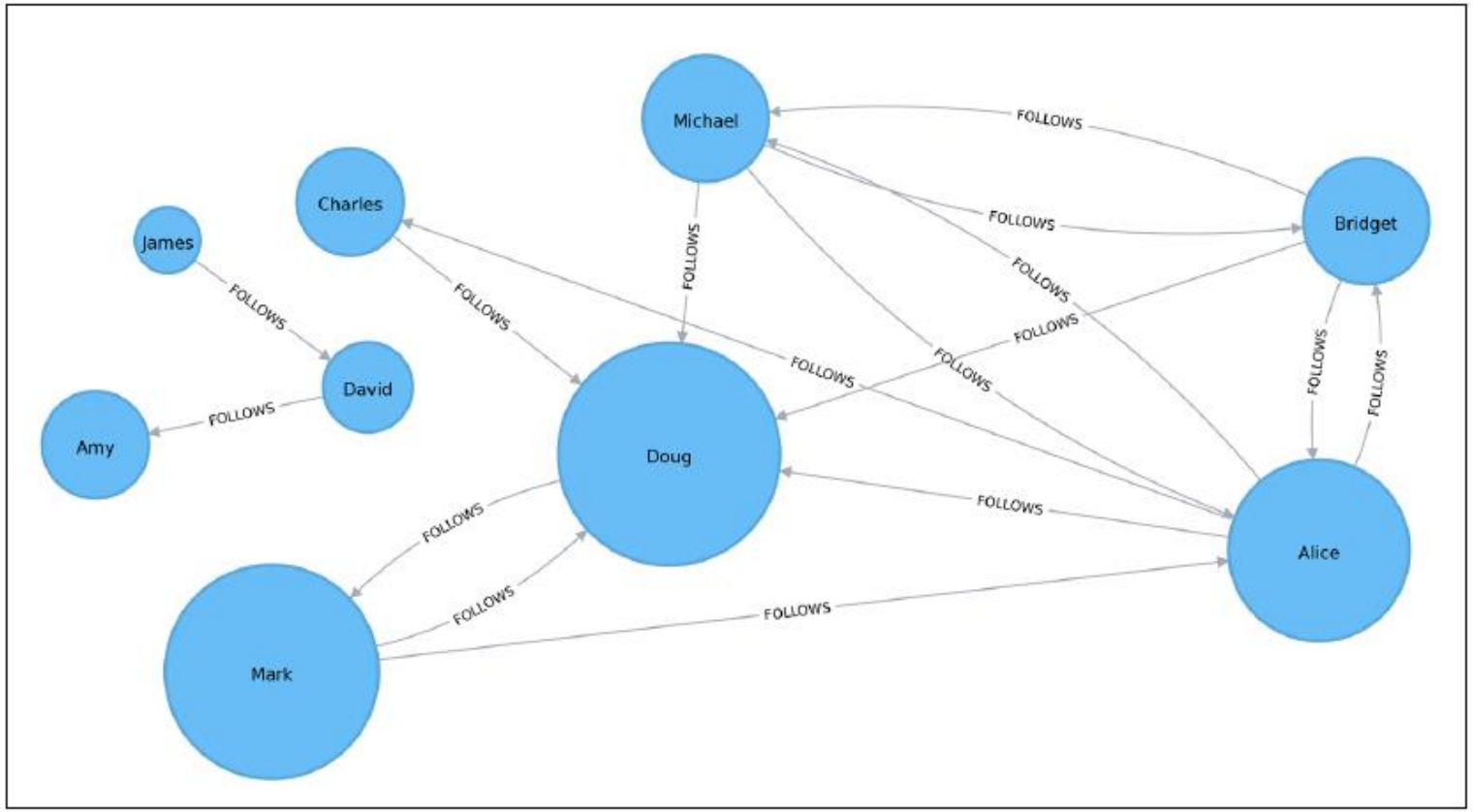
PageRank - Neo4j

Note: PageRank implementations vary, so they can produce different scoring even when the ordering is the same. Neo4j initializes nodes using a value of 1 minus the dampening factor whereas Spark uses a value of 1. In this case, the relative rankings (the goal of PageRank) are identical but the underlying score values used to reach those results are different.

From Figure (next slide). As we might expect, Doug has the highest PageRank because he is followed by all other users in his subgraph. Although Mark only has one follower, that follower is Doug, so Mark is also considered important in this graph. It's not only the number of followers that is important, but also the importance of those followers.

Centrality Algorithms

PageRank - Neo4j



Graph Algorithms Mark Needham and Amy E. Hodler, O'Reilly 2019

Centrality Algorithms

PageRank Variation – Personalised PageRank

Personalized PageRank (PPR) is a variant of the PageRank algorithm that calculates the importance of nodes in a graph from the perspective of a specific node. For PPR, random jumps refer back to a given set of starting nodes. This biases results toward, or personalizes for, the start node. This bias and localization make PPR useful for highly targeted recommendations.

id	pageRank
Alice	0.1650183746272782
Michael	0.048842467744891996
Bridget	0.048842467744891996
Charles	0.03497796119878669
David	0.0
James	0.0
Amy	0.0

In the demonstration example, the results of this query could be used to make recommendations for people who for example Doug should follow. Alice is the best suggestion for somebody that Doug should follow, but we might suggest Michael and Bridget as well.

Graph Algorithms Mark Needham and Amy E. Hodler, O'Reilly 2019

Centrality Algorithms

Summary

There are many wide-ranging uses for centrality algorithms and exploration is encouraged for a variety of analyses. You can apply centrality algorithms to locate optimal touch points for disseminating information, find the hidden brokers that control the flow of resources and uncover the indirect power players lurking in the shadows.

Centrality Algorithms

Continuous Assessment Exercise 1

Padgett's data on marriage alliances among leading Florentine families in the latter Renaissance is given in the Figure. Each node is a family, and each edge denotes a connection by marriage. Load the data as a Neo4j Knowledge Graph. For each node, compute and interpret the following measures of centrality:

Degree centrality
Closeness centrality
Betweenness centrality
Eigenvector centrality
Pagerank

Neo4j Bloom or Python NetworkX can be used to visualise your insights.