TECHNISCHE UNIVERSITÄT DRESDEN

KNOWLEDGE GRAPHS

Lecture 12: Centrality

Markus Krötzsch
Knowledge-Based Systems

TU Dresden, 15th Jan 2019

How to find important nodes?

Query languages provide many ways to find relevant (tuples of) vertices in a graph.

But many applications require us to order them, with the most "important" ones on top.

Possible Solution: use query to order results by some stored or computed value, e.g.:

- Wikidata's SPARQL endpoint includes the number of Wikimedia pages associated with any entity (property `wikibase:sitelinks`). This is a useful metric to estimate an item's importance across communities.
- Aggregation can compute metrics, e.g., films that won the biggest total number of awards might be most important.

Problem: This approach is highly application-dependent and limited to notions of importance that are easy enough to obtain for interactive querying.

Are there more general ways of finding "important" vertices in a graph?

(We might then pre-compute some such measures so queries can use them.)

Network analysis and Centrality

Network analysis has developed methods for finding the most important vertices in a graph (usually unlabelled; either directed or undirected).

$\rightsquigarrow$ vertex importance based on the structure of such graphs is called centrality

Example 12.1: Networks arise in many areas:

- transportation networks
- communication networks (esp. the Internet)
- link networks (esp. the Web, but also, e.g., citation networks)
- social networks (Twitter, Facebook, ...)
- networks that mirror chemical reactions or biological processes
- ...

Any network can be stored as (part of) a knowledge graph, so network analysis methods are relevant to KG analysis.

What is "important"?

Different types of networks lead to different notions of centrality.

Some considerations:

- Many networks can be considered to describe a flow of something (goods, information, readers, credibility, ...)
- A node might be "important" a lot flows from it (in a supply chain), to it (in a network of links), or through it (in a communication network)
- Flow might be modelled by (weighted) paths, possibly factoring in their length and/or number
- Only some paths might be of interest (shortest paths, vertex-simple paths, edge-simple paths, one-step paths, ...)
- Paths might be more important if they pass through important nodes (either based on a given prior notion of importance, or recursively)
- In knowledge graphs, the importance of edges and nodes may also depend on more complex features (e.g., edge or vertex labels), and some edges/vertices might be ignored altogether

Degree centrality

A primitive form of centrality restricts to incoming/outgoing paths of length one:

Definition 12.2: The in-degree centrality of a directed graph assigns is given by the in-degree of each node. The out-degree centrality and the degree centrality (for undirected graphs) are defined analogously.

Example 12.3: Degree centrality can easily be computed in queries using counting aggregates, and we have used variants of this idea in examples before (for a specific property/relationship type of our labelled graph). The count of Wikimedia sites in Wikidata's `wikibase:sitelinks` is a pre-computed degree centrality.

Advantages: easy to compute, easy to update, easy to understand

Disadvantages: considers only direct neighbourhood, easy to manipulate

The Random Walker

We can model "flow" in a network by asking where a particular "thing" that moves randomly across the network would likely end up

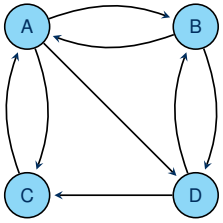
Random Walk:

- A walk may start at any node of the network with some probability (default assumption: the same uniform probability for each starting point)
- In each step, the walk may continue along any edge, in the direction of edges if the network is directed with some probability (default assumption: the same uniform probability for each choice of edge)

Idea: centrality of a node could be measured by the probability of a random walker being in that location after a large number of steps

How can we compute this?

Example



Transition matrix:

	A	B	C	D
A	0	1/2	1	0
B	1/3	0	0	1/2
C	1/3	0	0	1/2
D	1/3	1/2	0	0

Initial probability: (for being in location [A, B, C, D])

$[1/4, 1/4, 1/4, 1/4]$

Doing one step:

Probability of being in A becomes:

$$1/4 * 0 + 1/4 * 1/2 + 1/4 * 1 + 1/4 * 0 = 3/8,$$

i.e., $\sum_{N \in \{A,B,C,D\}} \Pr(\text{being in } N) * \Pr(\text{walking } N \text{ to } A)$

Using the same idea for the other cases, we get $[3/8, 5/24, 5/24, 5/24] \approx [0.38, 0.21, 0.21, 0.21]$

Further steps:

We can continue with this approach to get:
$[15/48, 11/48, 11/48, 11/48] \approx [0.31, 0.23, 0.23, 0.23]$
$[11/32, 7/32, 7/32, 7/32] \approx [0.34, 0.22, 0.22, 0.22]$
$[21/64, 43/192, 43/192, 43/192] \approx [0.33, 0.22, 0.22, 0.22]$
...

Eigenvectors

Does the iterative process in the example converge to some limit?

Yes, it actually does, with the limit being $[1/3, 2/9, 2/9, 2/9] = [0.333\ldots, 0.222\ldots, 0.222\ldots, 0.222\ldots]$

What characterises this limit?

Written as a column vector, it is the unique vector $\mathbf{v}$ such that

$$\mathbf{M}\mathbf{v} = \mathbf{v}$$

where $\mathbf{M}$ is the transition matrix. Indeed, matrix multiplication is exactly the operation we iteratively applied in the example.

Definition 12.4: Given a square matrix $\mathbf{M}$ of real values, a (right) eigenvector of $\mathbf{M}$ is a (column) vector $\mathbf{v}$ that is not the zero vector and for which there is a real number λ such that $\mathbf{M}\mathbf{v} = \lambda\mathbf{v}$. In this case, λ is called an eigenvalue of $\mathbf{M}$.

The emerging centrality criterion is called eigenvector centrality.

Eigenvector centrality

Note: If we multiply an eigenvector by a non-zero scalar, we get another eigenvector for the same eigenvalue. A common approach is to normalise (positive) eigenvectors so that their sum is 1, and components can be interpreted as probabilities.

Of course, there can also be completely different eigenvectors for different eigenvalues.

To define a centrality measure, we still need some guarantees:

- a well-defined specific eigenvector among all possible ones (scaling is acceptable, since we only need the relative importance of nodes)
- where all components of the vector should be non-negative (interpreting mixed sign components as "importance" would seem strange)
- and which can be computed in a practical fashion (maybe using iteration as in the example)

Thankfully, we will find that all of these issues are resolved in many common cases.

Well-behaved eigenvalues

Definition 12.5: Every $n \times n$ matrix $\mathbf{M}$ of non-negative real numbers gives rise to a directed graph $G_{\mathbf{M}}$ that has n vertices and an edge $v_i \to v_j$ if and only if $\mathbf{M}_{ij} > 0$. $\mathbf{M}$ is irreducible if $G_{\mathbf{M}}$ is strongly connected.

Example 12.6: Any stochastic matrix where columns express non-zero (but not necessarily uniform) transition probabilities in some strongly connected graph is irreducible.

Theorem 12.7 (Perron–Frobenius Theorem): Consider square matrix $\mathbf{M}$ of non-negative real numbers. If $\mathbf{M}$ is irreducible then

- $\mathbf{M}$ has an eigenvalue r that has the highest absolute value among all eigenvalues, and that is positive ($r > 0$),
- $\mathbf{M}$ has a right eigenvector $\mathbf{v}$ with eigenvalue r whose components are all positive.

Computing eigenvalue centrality

In many more general cases, the eigenvalue with the largest absolute value can be found by iterated matrix multiplication starting from any initial vector.

For common networks and stochastic matrices, this computation converges very quickly. It also works well using floating point arithmetic and allows for effective parallelisation.

Restriction: Eigenvector centrality requires the graph to be strongly connected, i.e.,

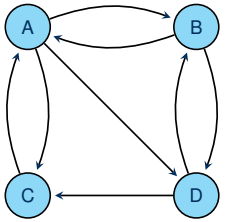
- contain no dead ends (vertices without outgoing edges), and
- contain no spider traps (cyclic structures with no path back to all nodes)

Intuitive explanation:

- Dead ends don't let a random walker continue. The weight of the node is not further propagated and therefore lost. Vertices with a path to a dead end likewise loose weight (in a stochastic matrix converging to 0).
- Spider traps will catch random walkers. Again, weight is drained from vertices with a path into the trap.

In both cases, iterative multiplication does not lead to a meaningful centrality measure.

More fun with matrix multiplication



Adjacency matrix:

$$\mathbf{M} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

(number of paths of length 1)

Matrix multiplication can be used to compute the number of paths between two nodes:

$$\mathbf{MM} = \begin{bmatrix} 2 & 0 & 0 & 2 \\ 1 & 2 & 1 & 0 \\ 1 & 2 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$ (number of length-2 paths)

$$\mathbf{MMM} = \begin{bmatrix} 2 & 4 & 2 & 0 \\ 3 & 1 & 1 & 3 \\ 3 & 1 & 1 & 3 \\ 3 & 2 & 1 & 2 \end{bmatrix}$$ (number of length-3 paths)

Katz centrality

Can we use path counting to compute centrality?

Idea: A node is central if many paths lead to it.

Problem: There can be an infinite number of (directed) paths in a graph with cycles.

Solution: (Leo Katz, 1953) Penalize longer paths using (rapidly shrinking) powers of some attenuation factor $\alpha < 1$.

Definition 12.8: Given an $n \times n$ column-based adjacency matrix $\mathbf{M}$ and a real number $0 < \alpha < 1$, the (in-bound/receiving) Katz centrality of a node $i \leq n$ is the result of the limit

$$\sum_{k \geq 1} \sum_{j=1}^{n} \alpha^k (\mathbf{M}^k)_{ij},$$

i.e., sum of the α^k-weighted sums of the i-th row in matrix $\mathbf{M}^k$.

Note: For this to work, the α must not be too large. Precisely, we should have $\alpha < 1/|r|$, where r is the eigenvalue of $\mathbf{M}$ with the largest absolute value.

Katz centrality: discussion

Katz overcomes some problems of pure eigenvector approaches:

- Dead ends do not drain all weight from the remaining graph (Katz centrality is meaningful on acyclic graphs)
- Spider traps do not drain all weight from the remaining graph (but they retain some centrality advantage)

The method is sensitive to the choice of the attenuation factor:

- Large values lead us to consider long paths, making results somewhat similar to eigenvector centrality.
- Small values focus on (very) short paths, making results more similar to in-degree centrality.

PageRank

A slightly different modification of eigenvector centrality was proposed by Sergey Brin and Larry Page when investigating centrality criteria for ranking Web pages in search results in 1996.

Goal: Address the limits of eigenvector centrality while keeping the idea of a random walker

Approach:

- Continue to consider random walks based on a stochastic transition matrix
- But also allow walkers to teleport to a random new location with some small probability $1 - \beta$.

Iterative computing scheme (PageRank with taxation):

$$\mathbf{v} := \beta \mathbf{M} \mathbf{v} + (1 - \beta)/N$$

where β is the damping factor used for taxation, and N is the number of nodes.

PageRank in practice

Dampening factor: For Web search and similar applications, β is usually close to 0.85, but other domains have reported rather different factors to be optimal.

- A large dampening factor emphasise gains due to long paths, but suffers more strongly from the problems with pure eigenvector centrality (dead ends, spider traps)
- A small dampening factor reduces these problems, but leads to a more local perspective that ignores long paths.

Computation: The iterative computation often converges quickly (50 to 75 iterations suffice to compute PageRanks for the Web to the limits of double precision!).

The limit of the computation is the vector $\mathbf{v}$ that satisfies the equation:

$$\beta \mathbf{M} \mathbf{v} + (1 - \beta)/N = \mathbf{v}$$

However, it is not feasible to solve this equation algebraically for large networks, so implementations mostly rely on (optimised, parallelised) iteration.

Page vs. Katz

We can see strong parallels between the two algorithms:

- Both are all-paths criteria based on the number of paths leading to a node
- Both use a parameter to reduce the impact of long paths

The main difference is that Katz propagates the full (discounted) weight to all successors of a node, whereas PageRank (like eigenvector centrality) splits the weight among all successors:

- Katz centrality: "A node is important if it is highly linked or if it is linked from other important nodes."
- PageRank: "A node is important if it is highly linked or if it is linked from other important nodes that do not link many other pages."

But maybe the most important difference is that one of the two has been famed as the secret of Google's success ...

Summary

Centrality criteria aim at identifying the most important vertices in a graph, based on graph structures

Eigenvalue centrality and related measures have been applied in a wide range of areas, notably Web search

What's next?

- More network analysis
- Summary
- Examinations