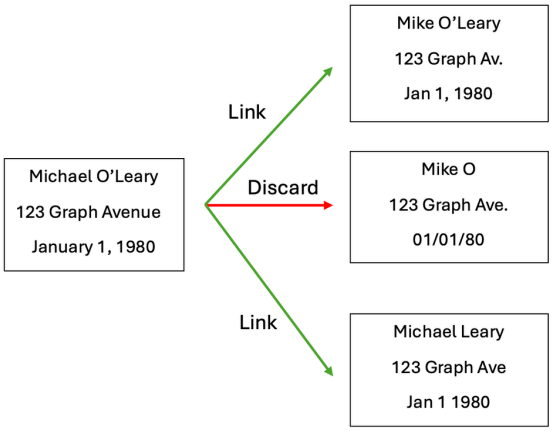
Graph Data Science for Entity Resolution, Pt. II: Betweenness Centrality

The Advantage of Using Graph Models

Graph Data Science has diverse applications in Entity Resolution, some of which I previously explored here. Essentially, graph methods will treat records for an individual (or "entity") as nodes on a graph. When framing data this way, we unlock the analytic abilities of Graph Theory, a branch of math focused on patterns between nodes.

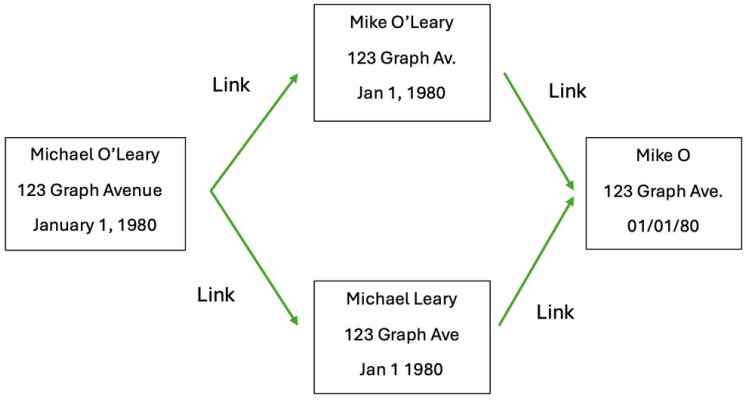
Many avoidable problems arise when an entity resolution system only compares records one-to-one (and therefore sees the similarities), but a graph-based solution looks at *the relationship of a node to all other nodes it's connected to*. In other words, a traditional entity resolution system doesn't look at the entire identity and all the records which compose it; it only matches *pairs* of records.

Let's examine a simple scenario involving a model which only compares records one-to-one. Here, an ML model using word similarity fails to match a record to an identity because it's not similar enough to the first established record in the identity.



A human can tell that the Mike O record is related to the others, but many machine learning models could not.

When merely comparing pairs of records, the ML model fails to make a connection. But a graph-based model links the Mike O record to the other candidates, sees that the other candidate records are connected to the first record, then connects them all.



Now, let's look at the reverse scenario: a traditional ML model mistakenly linking identities. My previous write-up examined an issue dubbed "transitive matching," wherein a record from one entity resembles a record from another, leading an Entity Resolution system to (incorrectly) group them together.

Mistakenly combined records cause serious headaches for companies- what if you denied Bill Johnson a loan along with William Johnson because you merged them in your database, even though Bill was a highly qualified applicant? What if you mailed a product to the wrong person because your Entity Resolution system confused their addresses? What if that product was a prescription medication?

In this guide, we'll explore how to refine an Entity Resolution system using a simple algorithm called Betweenness Centrality. We'll be using the Neo4j implementation of betweenness centrality found here.

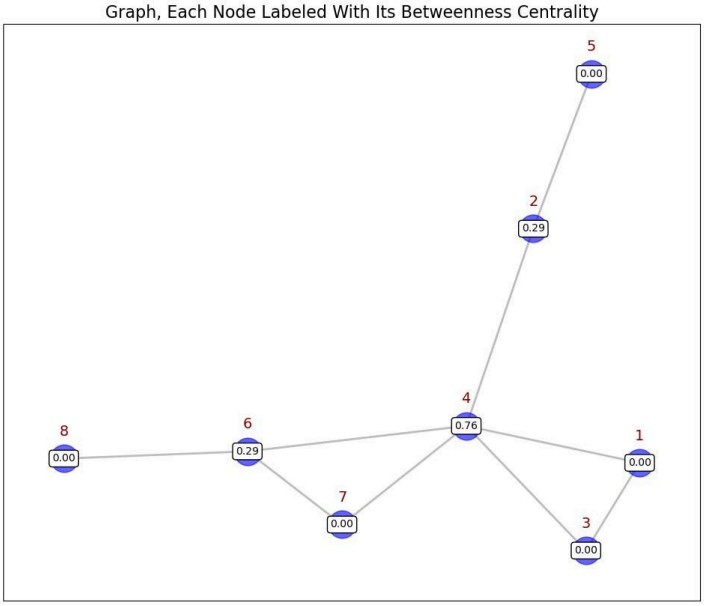
What is Betweenness Centrality?

There are several fundamental centrality metrics to find the most "important" or "relevant" nodes in a graph. The idea of "importance" for a node is, obviously, subjective so researchers fashioned different methods to capture it. And, just like in traditional machine learning, one can employ multiple performance metrics to get a better idea of how well a model does its job.

The popular centrality metrics are widely available in programming packages (and most are easy to understand and interpret), but betweenness is best suited for our use case.

To explain what Betweenness Centrality is, briefly: if a node X is part of the shortest path between two other nodes, node X has a higher Betweenness Centrality score. Being *connected* to many other nodes may increase a Betweenness Centrality score, but being on the shortest route between two others is what Betweenness really detects. Being the *only* node between many others would yield a very high centrality score.

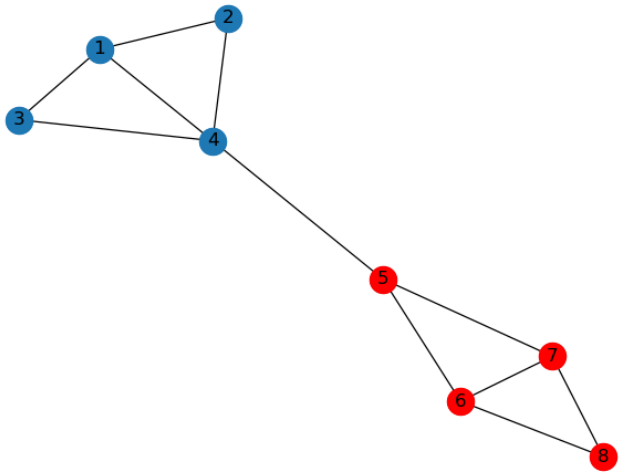
For example, a central train station in a city would have a high Betweenness score because you need to stop there to change subway lines. If it were deleted, you could not travel from one line to the other, and the city's subway map would look like some disconnected collections of stops. Here's an example on a graph:



Node 4 has the highest Betweenness Centrality since it's between the most nodes. Nodes 2 and 6 are in the shortest path between a few other nodes, so they have some small centrality scores as well. The rest are peripheral nodes with a centrality of 0.

If you removed Node 4, you would be left with 3 disconnected collections of nodes! Ergo, Node 4 has the highest Betweenness Centrality.

Now, recall that an erroneous "transitive match" usually resembles a barbell, with the nodes for one identity on one side and another identity's nodes on the other, connected by an erroneous link in the middle.



The red and blue entities here have been mistakenly connected because records 4 and 5 have similar data.

For our use case, we want to find a node which builds a “bridge” between two groups of nodes, meaning it is connected to several nodes on one side and several on the other. That node may not be *directly* connected to the nodes on the other side, but there is a *path* between these two groups and a transitive node is surely in the middle of it. **To get from, say, Node 6 to Node 1, you *need* to pass through nodes 4 and 5; therefore, 4 and 5 have the highest betweenness centrality score.**

Betweenness Centrality Compared to Other Graph Methods

When the Transitive Match problem arises, we could run a graph algorithm which builds entire identities on its own (i.e. Community Detection, discussed previously) to try and avoid false pairings. But that could create new kinds of errors (perhaps splitting records into separate identities too aggressively). Betweenness Centrality simply identifies the erroneous link.

Among centrality metrics, Betweenness provides the best conceptual framework for finding an erroneous link. Degree centrality is the simplest centrality metric- it just counts the number of edges connected to a node. A transitive node may only be connected to three other nodes, which is the same as a moderately-complete record placed in the right identity. So degree doesn’t quite capture the nuance of a transitive node.

You may be wondering, “what use is an algorithm which simply finds erroneous links?” Without such a tool, you may not know how many ill-composed identities are hiding in your data. To find out, you would need to go through all your identities manually to see if they make sense! Adding Betweenness Centrality to your répertoire can help clean your databases automatically.

How does centrality find the erroneous links?

If a record connects two identities, it’s connected to two sets of records, and will therefore have an abnormally high Betweenness Centrality score. The method here is to **find nodes with an unusually high betweenness centrality score because they are probably the records responsible for transitive matches**.

Once erroneous links are found, there are several possible next steps:

1. Human Handoff to analyze the data and let an employee decide which identities the records belong to. This route is what Data Surge’s client preferred, because preventing erroneous links was the highest priority. Erroneous links were also rare enough that it did not incur much cost for their business’s scale.
2. Join the record to the identity it has the highest matching score with. In other words, split the false connection and automatically join the “transitive” record with the record it is otherwise closest-related to.
3. Run a graph-based method for composing the identities again (i.e. community detection), and use that identity. In this scenario, you have employees who are familiar with Graph Data Science and can capably build a new Entity Resolution system based on graph methods.

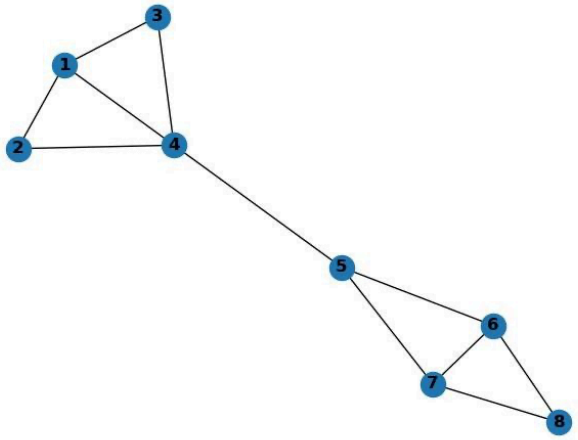
Looking at an Example

We previously discussed the case of Stephen Graphmann and Stephanie Garrison, unwittingly merged by non-graph-based Entity Resolution. Their records in our database are shown below.

	First Name	Last Name	Home State	Phone #	AssignedSystemID
1	Stephen	Graphman	Maryland	(555) 555-5555	12345
2	Stephen	Graphman	Maryland	555-5555	12345
3	Steve	Graphman	MD	(555) 555-5555	12345
4	Stephen	G.	MD	(blank)	12345
5	Steph	G.	MN	(blank)	12345
6	Steph	Garrison	MN	(blank)	12345
7	Stephanie	Garrison	MN	(444) 444-4444	12345
8	Stephanie	Garrison	Minnesota	(444) 444-4444	12345

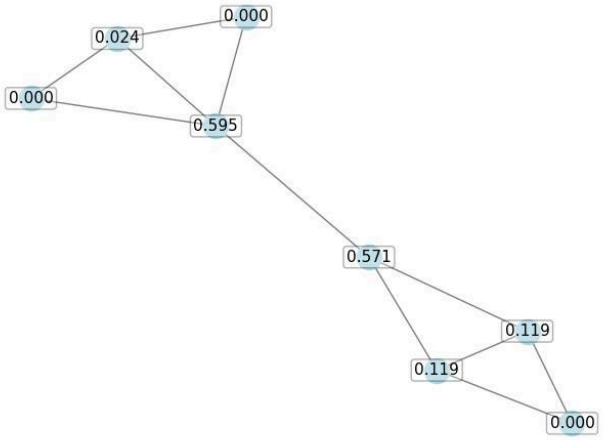
Records 4 and 5 are the culprits; their similarity confused the entity resolution system into thinking Stephanie and Steve are the same person.

When we plot the connections between records made by the non-graph-based Entity Resolution system, the network of records makes the “barbell” from earlier:



Each record from the sample above is a node on the graph. Each “edge” (connection between nodes) indicates where a machine learning model decided two records are part of the same identity. Records 4 and 5 link the “barbell”.

Now, according to our hypothesis, 4 and 5 will have a much higher betweenness centrality score and can be identified as the records responsible for the transitive match.

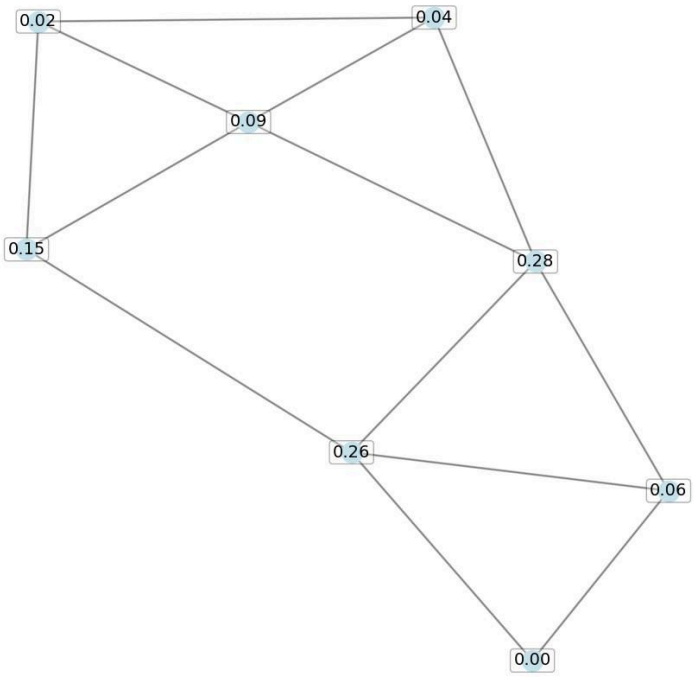


Graph with nodes labeled by their centrality score. The nodes responsible for the transitive match have the highest score.

We see that nodes 4 and 5 have the highest Betweenness Centrality, and are the culprits responsible for the erroneous link.

Caveats and Pitfalls of Betweenness Centrality

- Because it keeps track of relationships between all nodes in the graph, computing Betweenness can require a lot of memory and processing power; such is the price of accuracy and reliability.
- It’s also wise to only look at nodes with a significantly higher centrality score than the others; a score only slightly higher may just indicate a nice, complete record which linked several disparate records (perhaps from different data sources), but are truly part of the same identity.
- Betweenness Centrality is most effective when you only have one record from each identity that looks like the other identity. But if you have multiple records from one identity connected to multiple records from the other, centrality gets diluted.



Signal reduced to noise: adding 2 more edges between nodes has rendered Betweenness Centrality an uninformative metric since no node's score is significantly higher than the others.

Applying Betweenness to Your Data

Although it's a simple concept, centrality is quite a powerful metric for data analysis. A strong case is made for centrality metrics because of their explainability and their specificity in attacking the heart of this problem. Betweenness centrality has been used in cities to identify where is most beneficial to build new pedestrian and cycling infrastructure and make cities more sustainable. PageRank, the algorithm made famous by Google for determining the relevance of search results, is actually a centrality measure (a variant of eigenvector centrality). While centrality was traditionally used to find the most relevant or important nodes in a network, we've found a novel use for it reliably identifying flaws in our data!

At Data Surge, we have a team of consultants highly skilled in graph data science, along with a solutions accelerator tailored specifically for graph work.