Similarity Algorithms in Neo4j
Arsal Munawar

The Jaccard Similarity Algorithm

This algorithm measures the similarities between sets. Similarity is defined as the size of intersection of two sets divided by the size of the union of two sets. The following formula is used

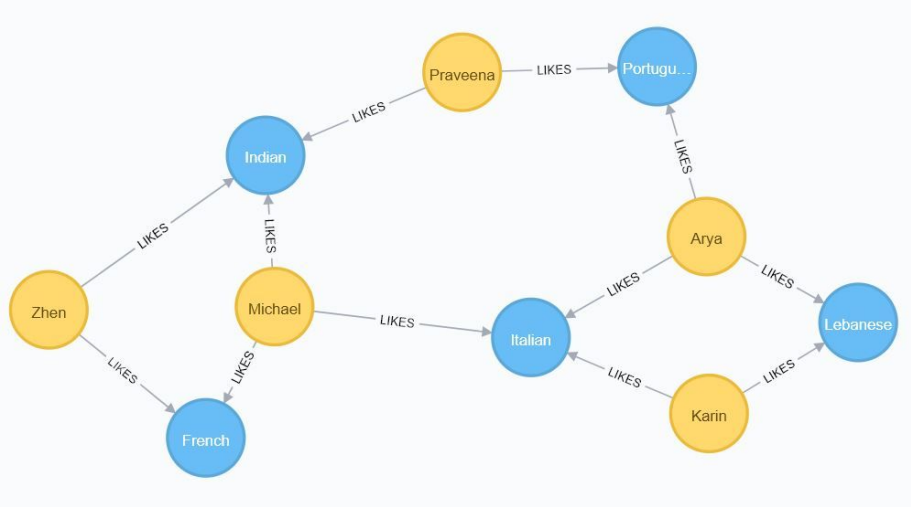
$$J(A,B) = |A \cap B| / |(A \cup B)|$$
$$= |A \cap B| / |A| + |B| - |A \cap B|$$

Where $|A \cap B|$ is the intersection of sets, and $|(A \cup B)|$ is the union of sets. For example, if set A contained 1,2,3 and set B contained 3,4,5, $|A \cap B|$ would equal 1 since there is only one overlapping element and $|(A \cup B)|$ would equal 5 since there are 5 total elements minus any repeats. As a result, J(A,B) would equal 1/5 = 0.20. This algorithm can be used to give friend recommendations. If person A is not friends with person B on Facebook but they have numerous mutual friends, then a Jaccard coefficient can be calculated between the two sets of friends to determine if they should be suggested to each other. The algorithm works best when calculating the similarity between small numbers of sets.

Below is the script to create the data and a graph. The yellow nodes are people, and the blue nodes are the types of cuisine those people like.

```
MERGE (french:Cuisine {name:'French'})
MERGE (italian:Cuisine {name:'Italian'})
MERGE (indian:Cuisine {name:'Indian'})
MERGE (lebanese:Cuisine {name:'Lebanese'})
MERGE (portuguese:Cuisine {name:'Portuguese'})
MERGE (zhen:Person {name: "Zhen"})
MERGE (praveena:Person {name: "Praveena"})
MERGE (michael:Person {name: "Michael"})
MERGE (arya:Person {name: "Arya"})
MERGE (karin:Person {name: "Karin"})
MERGE (praveena)-[:LIKES]->(indian)
MERGE (praveena)-[:LIKES]->(portuguese)
MERGE (zhen)-[:LIKES]->(french)
MERGE (zhen)-[:LIKES]->(indian)
MERGE (michael)-[:LIKES]->(french)
MERGE (michael)-[:LIKES]->(italian)
MERGE (michael)-[:LIKES]->(indian)
MERGE (arya)-[:LIKES]->(lebanese)
MERGE (arya)-[:LIKES]->(italian)
MERGE (arya)-[:LIKES]->(portuguese)
MERGE (karin)-[:LIKES]->(lebanese)
```

```
MERGE (karin)-[:LIKES]->(italian)
```



Here is the query to call the algorithm. The results will be sorted based on descending order of Jaccard coefficient. Arya and Karin, and Zhen and Michael are the most similar to each other, while Zhen and Arya, and Zhen and Karin have nothing in common. Arya likes three cuisines and Karin likes two cuisines, and they like two of the same. Thus, their similarity is ⅔ = 0.66.

From	To	Intersection	Similarity
Arya	Karin	2	0.66
Zhen	Michael	2	0.66
Zhen	Praveena	1	0.33
Michael	Karin	1	0.25
Praveena	Michael	1	0.25
Praveena	Arya	1	0.25
Michael	Arya	1	0.2
Praveena	Karin	0	0
Zhen	Arya	0	0
Zhen	Karin	0	0

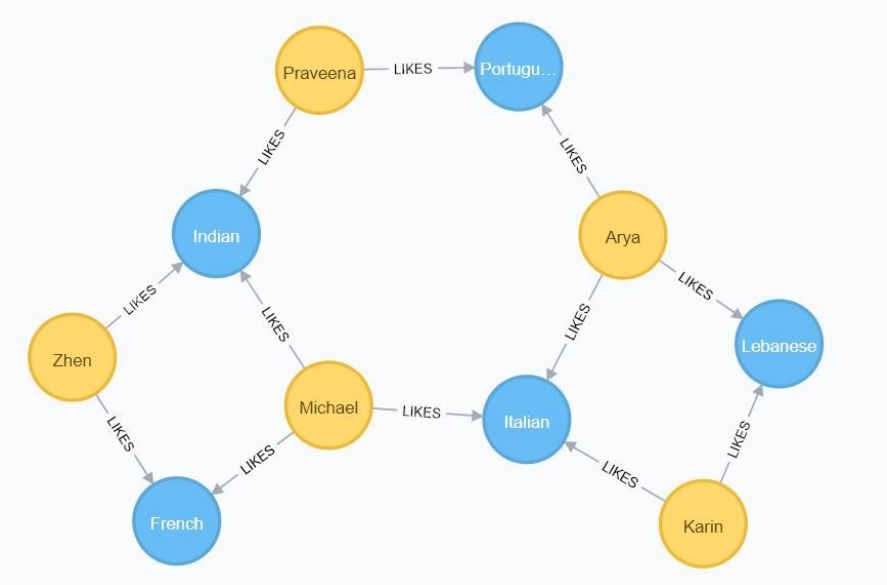
The Cosine Similarity Algorithm
This algorithm calculates the cosine of the angle between two vectors, or the dot product divided by the cross product as given by the following formula.

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|}$$

Here, the number of common attributes is divided by the number of possible attributes between two sets. The similarity values will range between -1 and 1, with -1 being absolutely dissimilar and 1 being absolutely similar. An interesting application of this algorithm would be in identifying plagiarism since it takes into account all possible attributes in two sets. If the number of common words between two papers if high, then a higher similarity score will result. The algorithm works best when calculating the similarity between small numbers of sets.

Below is the script to create the data and graph.

```
MERGE (french:Cuisine {name:'French'})
MERGE (italian:Cuisine {name:'Italian'})
MERGE (indian:Cuisine {name:'Indian'})
MERGE (lebanese:Cuisine {name:'Lebanese'})
MERGE (portuguese:Cuisine {name:'Portuguese'})
MERGE (zhen:Person {name: "Zhen"})
MERGE (praveena:Person {name: "Praveena"})
MERGE (michael:Person {name: "Michael"})
MERGE (arya:Person {name: "Arya"})
MERGE (karin:Person {name: "Karin"})
MERGE (praveena)-[:LIKES {score: 9}]->(indian)
MERGE (praveena)-[:LIKES {score: 7}]->(portuguese)
MERGE (zhen)-[:LIKES {score: 10}]->(french)
MERGE (zhen)-[:LIKES {score: 6}]->(indian)
MERGE (michael)-[:LIKES {score: 8}]->(french)
MERGE (michael)-[:LIKES {score: 7}]->(italian)
MERGE (michael)-[:LIKES {score: 9}]->(indian)
MERGE (arya)-[:LIKES {score: 10}]->(lebanese)
MERGE (arya)-[:LIKES {score: 10}]->(italian)
MERGE (arya)-[:LIKES {score: 7}]->(portuguese)
MERGE (karin)-[:LIKES {score: 9}]->(lebanese)
```



Here is the query to call the algorithm, and the results. Arya and Karin are the most similar as was proved by the previous algorithm.

```
MATCH (p:Person), (c:Cuisine)
OPTIONAL MATCH (p)-[likes:LIKES]->(c)
WITH {item:id(p), weights: collect(coalesce(likes.score, 0))} as userData
WITH collect(userData) as data
CALL algo.similarity.cosine.stream(data)
YIELD item1, item2, count1, count2, similarity
RETURN algo.getNodeById(item1).name AS from,
algo.getNodeById(item2).name AS to, similarity
ORDER BY similarity DESC
```

from	to	similarity
Arya	Karin	0.8893006975229283
Zhen	Michael	0.8249630162429022
Praveena	Michael	0.5100496780395022
Zhen	Praveena	0.4061183653774261
Michael	Arya	0.3184912471845722
Michael	Karin	0.3085485706658717
Praveena	Arya	0.2723483386163968

Zhen	Arya	0.0
Zhen	Karin	0.0
Praveena	Karin	0.0

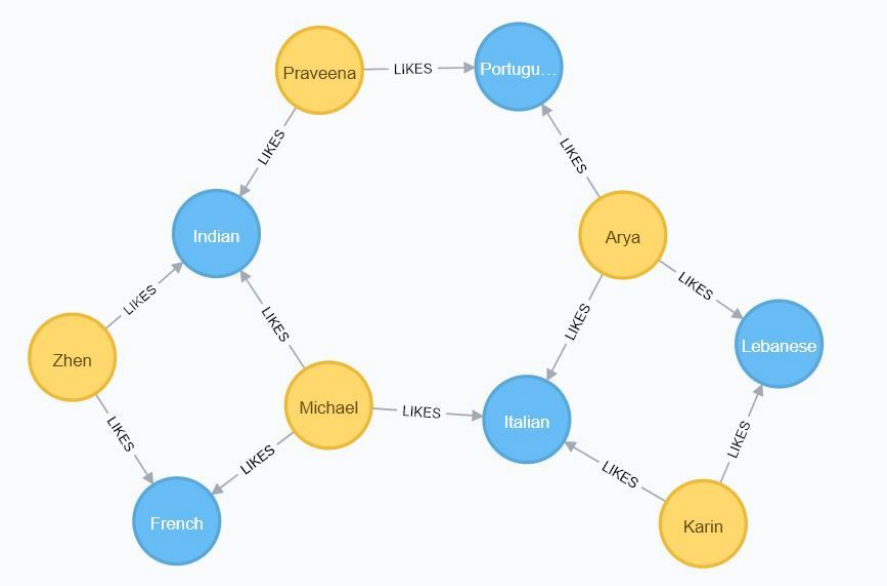
The Euclidean Distance Algorithm
This algorithm measures the straight-line distance between two points in space, in any amount of dimensions. The following formula is used to calculate this distance

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_i - q_i)^2 + \cdots + (p_n - q_n)^2}.$$

Where p_1,p_2,p_n are the x, y, n coordinates of one point and q_1,q_2,q_n the x, y, n coordinates for the second point. The algorithm works best when calculating the similarity between small numbers of sets. This algorithm can be used to measure straight line distance between two points on a map, so that an estimate of distance can be made. Some algorithms use the Euclidean measurement to estimate distance in order to traverse the shortest.

Below is the script to create the data and graph.

```
MERGE (french:Cuisine {name:'French'})
MERGE (italian:Cuisine {name:'Italian'})
MERGE (indian:Cuisine {name:'Indian'})
MERGE (lebanese:Cuisine {name:'Lebanese'})
MERGE (portuguese:Cuisine {name:'Portuguese'})
MERGE (zhen:Person {name: "Zhen"})
MERGE (praveena:Person {name: "Praveena"})
MERGE (michael:Person {name: "Michael"})
MERGE (arya:Person {name: "Arya"})
MERGE (karin:Person {name: "Karin"})
MERGE (praveena)-[:LIKES {score: 9}]->(indian)
MERGE (praveena)-[:LIKES {score: 7}]->(portuguese)
MERGE (zhen)-[:LIKES {score: 10}]->(french)
MERGE (zhen)-[:LIKES {score: 6}]->(indian)
MERGE (michael)-[:LIKES {score: 8}]->(french)
MERGE (michael)-[:LIKES {score: 7}]->(italian)
MERGE (michael)-[:LIKES {score: 9}]->(indian)
MERGE (arya)-[:LIKES {score: 10}]->(lebanese)
MERGE (arya)-[:LIKES {score: 10}]->(italian)
MERGE (arya)-[:LIKES {score: 7}]->(portuguese)
MERGE (karin)-[:LIKES {score: 9}]->(lebanese)
```



Here is the query to call the algorithm and the results. The score which is given to each link is used to calculate the similarity. A lower similarity means a lower straight-line distance. By using the numbers (10,10,7) for Arya and (9,7,0) for Karin, the similarity does indeed equal sqroot(59) = 7.68.

```
MATCH (p:Person), (c:Cuisine)
OPTIONAL MATCH (p)-[likes:LIKES]->(c)
WITH {item:id(p), weights: collect(coalesce(likes.score, 0))} as userData
WITH collect(userData) as data
CALL algo.similarity.euclidean.stream(data)
YIELD item1, item2, count1, count2, similarity
RETURN algo.getNodeById(item1).name AS from,
algo.getNodeById(item2).name AS to, similarity
ORDER BY similarity
```

from	to	similarity
Arya	Karin	7.681145747868608
Zhen	Michael	7.874007874011811
Zhen	Praveena	12.569805089976535
Praveena	Michael	12.727922061357855
Michael	Karin	15.033296378372908

Praveena	Karin	16.1245154965971
Zhen	Karin	16.30950643030009
Praveena	Arya	16.76305461424021
Michael	Arya	17.406895185529212
Zhen	Arya	19.621416870348583