# COSINE SIMILARITY

There are many methods to determine similarity between users or movies. For our use case, we have used Cosine Similarity to find the element of Similarity between two users and suggest movies to one of them accordingly.

**Cosine Similarity:** Cosine similarity is the cosine of the angle between two *n*-dimensional vectors in an *n*-dimensional space. It is the dot product of the two vectors divided by the product of the two vectors' lengths (or magnitudes). For two vectors A and B in an *n*-dimensional space:

$$similarity(A,B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Cosine similarity ranges between -1 and 1, where -1 is perfectly dissimilar and 1 is perfectly similar. Each person should be thought of as a vector where their coordinates are defined by their movie ratings. Thus:

$$\overrightarrow{M.Sherman} = \langle 3,8,7,5,2,9 \rangle$$

$$\overrightarrow{M.Hunger} = \langle 10,8,6,6,4,5 \rangle$$

$$similarity(M.Sherman, M.Hunger) = \frac{3 \cdot 10 + 8 \cdot 8 + 7 \cdot 6 + 5 \cdot 6 + 2 \cdot 4 + 9 \cdot 5}{\sqrt{3^2+8^2+7^2+5^2+2^2+9^2} \times \sqrt{10^2+8^2+6^2+6^2+4^2+5^2}} = \frac{219}{15.2315 \times 16.6433} = 0.8639$$

In our project, we added the “SIMILARITY” relationship between users so that we can find cosine similarity along with it. For that we fired the following query:

```
MATCH (p1:User)-[x:RATED]->(m:Movie)<-[y:RATED]-(p2:User)
WITH  SUM(x.rating * y.rating) AS xyDotProduct,
    SQRT(REDUCE(xDot = 0.0, a IN COLLECT(x.rating) | xDot + a^2)) AS xLength,
    SQRT(REDUCE(yDot = 0.0, b IN COLLECT(y.rating) | yDot + b^2)) AS yLength,
    p1, p2
MERGE (p1)-[s:SIMILARITY]-(p2)
SET   s.similarity = xyDotProduct / (xLength * yLength)
```

Here, as we can see the formula of the cosine similarity is used to compute the term and then the relationship is formed for each of the user. After that, “similarity” property is set to the “SIMILARITY” relationship. It is important to note that there is only one [:SIMILARITY] relationship between each person.

After creating the relationship and setting properties, we will check that the cosine similarity was calculated correctly or not. For that we fire following query:

```
MATCH  (p1:User {name:'Rudolph'})-[s:SIMILARITY]-(p2:User {name:'Forest'})
RETURN s.similarity AS `Cosine Similarity`
```

Here, for the users named “Rudolph” and “Forest” we find what the cosine similarity is.

With the similarities added to the graph, it is easy to view your *k*-nearest neighbours. Let’s view it for James’s 5-nearest neighbours with the following query:

```
MATCH    (p1:User {name:'James'})-[s:SIMILARITY]-(p2:User)
WITH     p2, s.similarity AS sim
ORDER BY sim DESC
LIMIT    5
RETURN   p2.name AS Neighbor, sim AS Similarity
```

Here, list of five people along with the cosine similarity are displayed in descending order, rated movies most similarly to James.

Finally, we wanted to provide recommendations for movies that a person hasn’t rated (which can be interpreted that they haven’t seen the movie). For this, we decided to find the average of the movie ratings from that person’s *k*-nearest neighbours (out of the neighbours who rated the relevant movie). We decided to use $k = 3$ for the movie recommendations; these recommendations should be thought of as estimates of how much the person would like (or how the person would rate) the movies they haven’t seen. This query was written in python so that we can ask the name of the user. To accomplish this, we fired the following query for any user you wish to find the movies for:

```
MATCH    (b:User)-[r:RATED]->(m:Movie), (b)-[s:SIMILARITY]-(a:User {username:'"""+x+"""'})
WHERE    NOT((a)-[:RATED]->(m))
WITH     m, s.similarity AS similarity, r.rating AS rating
ORDER BY m.title, similarity DESC
WITH     m.title AS movie, COLLECT(rating)[0..3] AS ratings
WITH     movie, REDUCE(s = 0, i IN ratings | s + i)*1.0 / SIZE(ratings) AS reco
ORDER BY reco DESC
RETURN   movie AS Movie, reco AS Recommendation
```