DS 4300 Homework 5

Spotify Recommendation System

Avril Mauro & Katelyn Donn

Sampling

When testing the computing power of Neo4j, we quickly agreed that we need to take a smaller sample of the full Spotify dataset to build our recommendation system. Our approach in doing so was inspired by our goal of taking the **top 20 songs from each genre** to include a robust variety of songs for our recommendation system to work off of. Next, we decided to pick some features to make the basis of our algorithm. After some exploratory data analysis, we decided on **valence, tempo, popularity, and genre** as our features and extracted these relevant columns from the original dataframe. We decided on these features because they did not show high correlation with each other and other features (lower risk of collinearity).

Preprocessing

We then encoded our categorical variable "genre" and scaled our numeric variables so that all values were between 0 and 1, allowing us to accurately compare these features across songs. Then, we dropped rows with null values and duplicates where track_name and artists were repeated together. Finally, we computed similarity scores between songs and created a dataframe of the similarity scores for each unique song pair, merged with the original dataframe to also include each song's artists, album titles, and genres. This resulting dataframe was then exported as a csv called **"spotify_test.csv"**. We sifted out the irrelevant columns for our sample data so every column can be utilized fully in our neo4j code.

Full Graph Model

Our full graph model has **3184 nodes** and **1,642,148 edges**. Each node represents a song and each edge represents the similarity between two songs with similarity score (simscore) as a relationship property. The node properties for each song include song title, artist, album name, and genre. We have 2 types of song nodes: Song1 and Song2, each corresponding to a unique pairing from our sample dataset. There are no repeat relationships (i.e. if Song1 is "Last Nite" and Song2 is "The Adults Are Talking", there is one edge to represent this and there does not exist a pairing where Song1 is "The Adults Are Talking" and Song2 is "Last Nite").

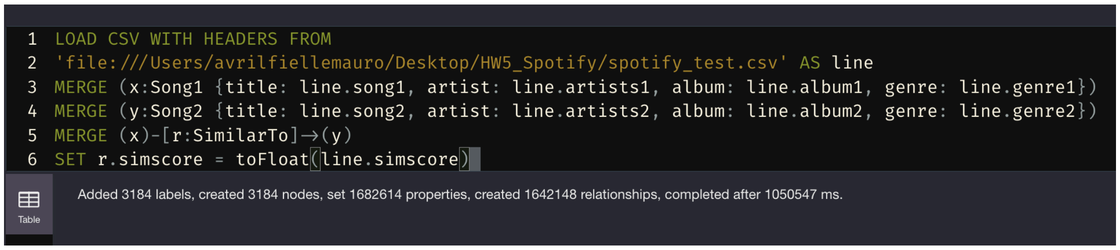
Algorithm

Our recommender system is inspired by **Content-Based Filtering** algorithms of tech leaders like Netflix who pride themselves on personalizing their recommendations for each user based on the content they interact with on the platform. Thus, leads us to the development of our similarity score.

We decided to take the **pairwise cosine similarity** of all numerical features in our sample, which, after preprocessing included valence, tempo, popularity, and genre encoded. We then assembled the songs and scores in a dataframe containing all unique pairs of songs in the sample. Finally, we used cypher queries to (1) output all pairs that included the input song ("Last Nite" by The Strokes), (2) sort the songs in descending order by similarity score, and (3) limit the results to 5 songs.

Cypher Queries

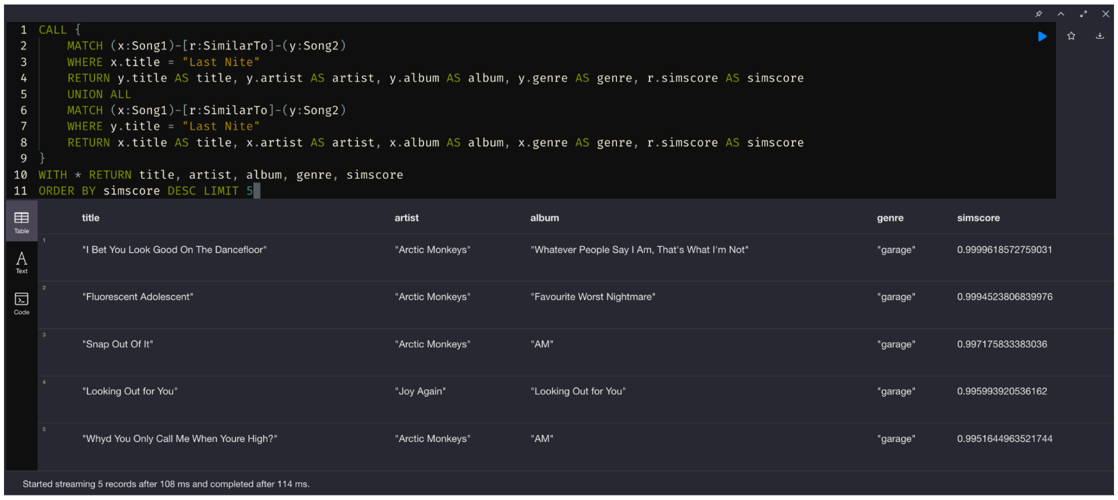
1. The first query, as seen below, builds our full graph database from our sample data.



```
LOAD CSV WITH HEADERS FROM
'file:///Users/avrilfiellemauro/Desktop/HW5_Spotify/spotify_test.csv' AS line
MERGE (x:Song1 {title: line.song1, artist: line.artists1, album: line.album1, genre: line.genre1})
MERGE (y:Song2 {title: line.song2, artist: line.artists2, album: line.album2, genre: line.genre2})
MERGE (x)-[r:SimilarTo]→(y)
SET r.simscore = toFloat(line.simscore)
```

```
Added 3184 labels, created 3184 nodes, set 1682614 properties, created 1642148 relationships, completed after 1050547 ms.
```

2. The second query focuses on creating relevant recommendations based on a user-inputted song of choice. It goes through the database, assesses the relationships that the song has, orders the relationships (the similarity scores) so the most prominent pairs are at the top, and returns the information of the top 5 pairings so the user can identify the proper recommendations.

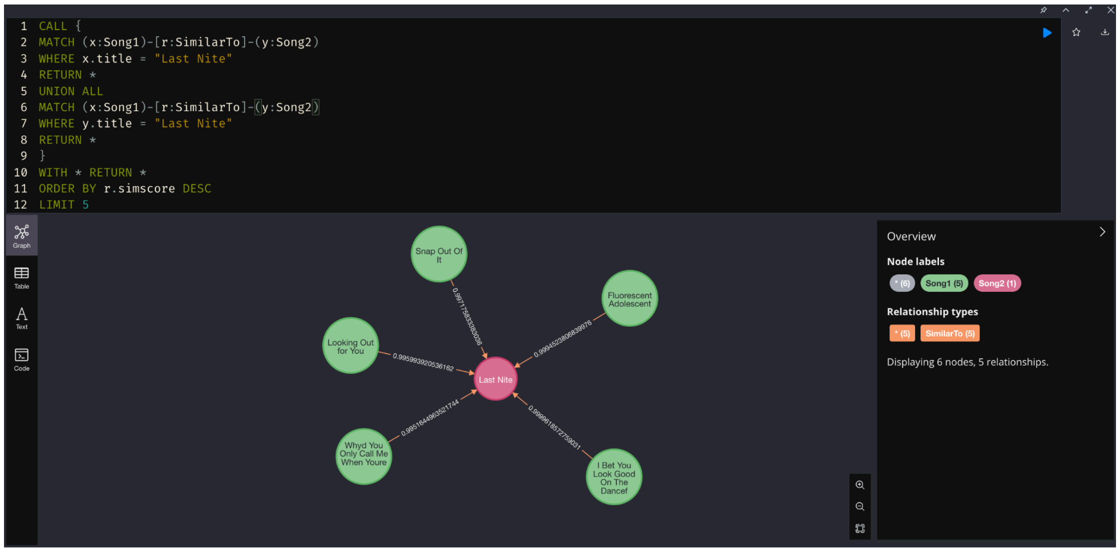


```
CALL {
    MATCH (x:Song1)-[r:SimilarTo]-(y:Song2)
    WHERE x.title = "Last Nite"
    RETURN y.title AS title, y.artist AS artist, y.album AS album, y.genre AS genre, r.simscore AS simscore
    UNION ALL
    MATCH (x:Song1)-[r:SimilarTo]-(y:Song2)
    WHERE y.title = "Last Nite"
    RETURN x.title AS title, x.artist AS artist, x.album AS album, x.genre AS genre, r.simscore AS simscore
}
WITH * RETURN title, artist, album, genre, simscore
ORDER BY simscore DESC LIMIT 5
```

	title	artist	album	genre	simscore
1	"I Bet You Look Good On The Dancefloor"	"Arctic Monkeys"	"Whatever People Say I Am, That's What I'm Not"	"garage"	0.9999618572759031
2	"Fluorescent Adolescent"	"Arctic Monkeys"	"Favourite Worst Nightmare"	"garage"	0.9994523806839976
3	"Snap Out Of It"	"Arctic Monkeys"	"AM"	"garage"	0.997175833383036
4	"Looking Out for You"	"Joy Again"	"Looking Out for You"	"garage"	0.995993920536162
5	"Whyd You Only Call Me When Youre High?"	"Arctic Monkeys"	"AM"	"garage"	0.9951644963521744

```
Started streaming 5 records after 108 ms and completed after 114 ms.
```

3. The third query displays the graph relationship between the input song and its recommended songs.



```
CALL {
MATCH (x:Song1)-[r:SimilarTo]-(y:Song2)
WHERE x.title = "Last Nite"
RETURN *
UNION ALL
MATCH (x:Song1)-[r:SimilarTo]-(y:Song2)
WHERE y.title = "Last Nite"
RETURN *
}
WITH * RETURN *
ORDER BY r.simscore DESC
LIMIT 5
```

Overview

Node labels

* (6) Song1 (5) Song2 (1)

Relationship types

* (5) SimilarTo (5)

Displaying 6 nodes, 5 relationships.

Cypher Queries (Text)

1. Build Graph Database

```
LOAD CSV WITH HEADERS FROM
'file:///Users/avrilfiellemauro/Desktop/HW5_Spotify/spotify_test.csv' AS line
MERGE (x:Song1 {title: line.song1,
                artist: line.artists1,
                album: line.album1,
                genre: line.genre1})
MERGE (y:Song2 {title: line.song2,
                artist: line.artists2,
                album: line.album2,
                genre: line.genre2})
MERGE (x)-[r:SimilarTo]->(y)
SET r.simscore = toFloat(line.simscore)
```

2. Top 5 Recommendations

```
CALL {
MATCH (x:Song1)-[r:SimilarTo]-(y:Song2)
      WHERE x.title = "Last Nite"
      RETURN y.title AS title, y.artist AS artist, y.album AS album, y.genre AS
      genre, r.simscore AS simscore
      UNION ALL
      MATCH (x:Song1)-[r:SimilarTo]-(y:Song2)
      WHERE y.title = "Last Nite"
      RETURN x.title AS title, x.artist AS artist, x.album AS album, x.genre AS
      genre, r.simscore AS simscore
}
WITH * RETURN title, artist, album, genre, simscore
ORDER BY simscore DESC LIMIT 5
```

3. Graph Visualization

```
CALL {
MATCH (x:Song1)-[r:SimilarTo]-(y:Song2)
      WHERE x.title = "Last Nite"
      RETURN *
      UNION ALL
      MATCH (x:Song1)-[r:SimilarTo]-(y:Song2)
      WHERE y.title = "Last Nite"
      RETURN *
}
WITH * RETURN *
ORDER BY simscore DESC LIMIT 5
```