MOVIE.CO

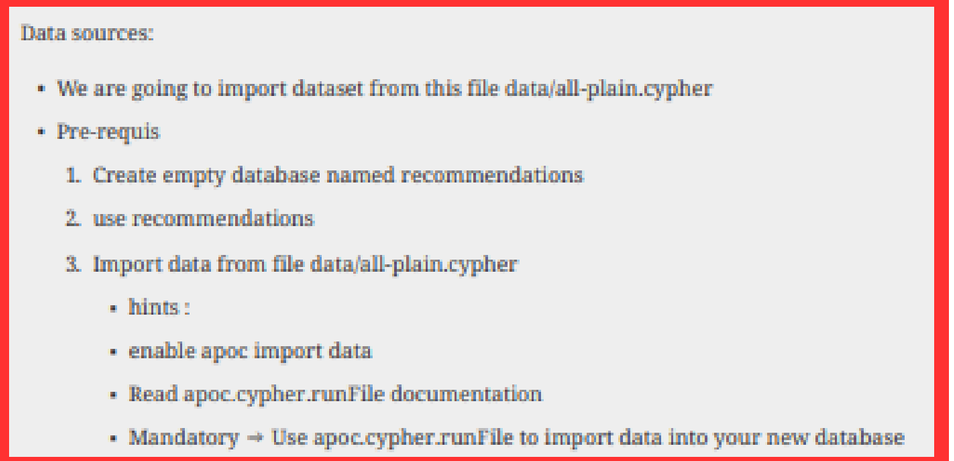
PROJET NEO4J

JUILLET 2024

Etudiants :
CAMARA Zakaria
CHAPELLON Nolwenn
SALMON Joris

INTRODUCTION

Le but du projet est de développer un système de recommandation de films en utilisant Neo4j, une base de données de graphes. Ce système de recommandation vise à proposer des films aux utilisateurs en se basant sur leurs préférences, les évaluations qu'ils ont données à différents films, ainsi que les relations entre les films, les genres, les acteurs et les réalisateurs.


Data sources:

- We are going to import dataset from this file data/all-plain.cypher
- Pre-requis
  1. Create empty database named recommendations
  2. use recommendations
  3. Import data from file data/all-plain.cypher
     - hints :
     - enable apoc import data
     - Read apoc.cypher.runFile documentation
     - Mandatory → Use apoc.cypher.runFile to import data into your new database

Méthodes utilisés :

1. Filtrage Collaboratif (Avoir des goûts similaires à l'avenir)

2. Filtrage Basé sur le Contenu (Caractéristiques des films: genres, des acteurs, des réalisateurs)

3. Algorithmes de Similarité

- Jaccard Index
- Cosine Similarity
- Pearson Similarity

DISSECTING A CYPHER STATEMENT

"How many reviews does each Matrix movie have?"

```
MATCH (m:Movie)<-[:RATED]-(u:User)
WHERE m.title CONTAINS 'Matrix'
WITH m, count(*) AS reviews
RETURN m.title AS movie, reviews
ORDER BY reviews DESC LIMIT 5;
```

	movie	reviews
1	"Matrix, The"	259
2	"Matrix Reloaded, The"	82
3	"Matrix Revolutions, The"	54

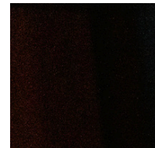
But : Trouver et lister les cinq films de la série "Matrix" et de donner nombre de critiques.
La requête recherche les films (m) dans la base de données qui contiennent le mot "Matrix" dans leur titre.

Le tableau montre les 5 films de la série "Matrix" avec le nombre de critiques respectives avec

1. "Matrix, The" avec 259 critiques.
2. "Matrix Reloaded, The" avec 82 critiques.
3. "Matrix Revolutions, The" avec 54 critiques.

Procedures

```
Call org.neo4j.example.reviewCount('Matrix') yield movieTitle, reviewCount return movieTitle,reviewCount
```



PERSONALIZED RECOMMENDATIONS

Content-Based Filtering

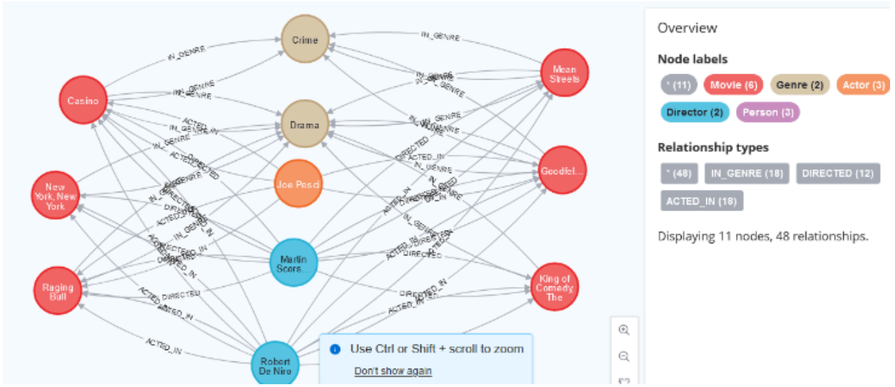
1) "Find Items similar to the item you're looking at now" TODO: Create Cypher query

```
MATCH (target:Movie {title: 'Casino'})
MATCH (target)-[:IN_GENRE]->(g:Genre)<-[:IN_GENRE]-(similar:Movie)
MATCH (target)<-[:DIRECTED]-(d:Director)-[:DIRECTED]->(similar)
MATCH (target)<-[:ACTED_IN]-(a:Actor)-[:ACTED_IN]->(similar)
WITH similar, count(g) + count(d) + count(a) AS score
RETURN similar.title AS Movie, score
ORDER BY score DESC
LIMIT 10;
```

	Movie	score
1	"Goodfellas"	12
2	"Mean Streets"	6
3	"Raging Bull"	6
4	"New York, New York"	3
5	"King of Comedy, The"	3

Procedures

CALL org.neo4j.movieRecommendations('Casino') yield movieTitle,score return movieTitle,score



Collaborative Filtering

1) " Get Users who got this item, also got that other item." TODO: Create Cypher query

```
MATCH (u1:User)-[:RATED]->(m:Movie)<-[:RATED]-(u2:User)
MATCH (u2)-[:RATED]->(other:Movie)
WHERE m.title = 'Star Wars: Episode IV - A New Hope' AND other.title <> 'Star Wars: Episode IV - A New Hope'
RETURN other.title AS RecommendedMovies, count(*) AS Frequency
ORDER BY Frequency DESC
LIMIT 10;
```

	recommendedMovie	frequency
1	"Star Wars: Episode V - The Empire Strikes Back"	58870
2	"Star Wars: Episode VI - Return of the Jedi"	54230
3	"Forrest Gump"	52780
4	"Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark)"	51330
5	"Matrix, The"	51330
6	"Pulp Fiction"	49590

Procedures

CALL org.neo4j.movieRecommendations2('Star Wars: Episode IV - A New Hope') YIELD recommendedMovie, frequency
RETURN recommendedMovie, frequency;

CONTENT-BASED FILTERING

Similarity Based on Common Genres

1) Find movies most similar to Inception based on shared genres // Find similar movies by common genres

movieTitle	commonGenres
"Patlabor: The Movie (Kidô keisatsu patorebâ: The Movie)"	6
"RoboCop"	6
"Strange Days"	6
"Watchmen"	6
"Blackhat"	5
"Cellular"	5

Procedures : CALL org.neo4j.similarMoviesByGenres('Inception') YIELD movieTitle, commonGenres RETURN movieTitle, commonGenres;

Personalized Recommendations Based on Genres

1) Recommend movies similar to those the user has already watched

	movieTitle	commonGenres
1	"Wonderful World of the Brothers Grimm, The"	8
2	"Enchanted"	7
3	"Mulan"	7
4	"Aladdin and the King of Thieves"	6
5	"All Dogs Go to Heaven 2"	6
6	"Aqua Teen Hunger Force Colon Movie Film for Theaters"	6

Procedures : CALL org.neo4j.personalizedRecommendations('Laurie Kirk') YIELD movieTitle, commonGenres RETURN movieTitle, commonGenres;

Weighted Content Algorithm

1) Compute a weighted sum based on the number and types of overlapping traits Hint: Find similar movies by common genres

	Movie	Score
1	"Toy Story 3"	38
2	"Toy Story"	36
3	"Ghostbusters II"	26
4	"Stuart Little 2"	26
5	"Austin Powers: International Man of Mystery"	25
6	"Star Wars: Episode II - Attack of the Clones"	25

CONTENT-BASED SIMILARITY METRICS

Jaccard Index

1) What movies are most similar to Inception based on Jaccard similarity of genres?

	Movie	jaccardIndex
1	"Strange Days"	0.857142857142857
2	"Watchmen"	0.857142857142857
3	"Blackhat"	0.7142857142857143
4	"Cellular"	0.7142857142857143
5	"Double, The"	0.7142857142857143
6	"Fast Five (Fast and the Furious 5, The)"	0.7142857142857143

Jaccard Index

1) Apply this same approach to all "traits" of the movie (genre, actors, directors, etc.)

Movie	toutJaccard	genreJaccard	acteurJaccard	directeurJa
'Insomnia"	0.5714285714285715	0.7142857142857143	0.0	1.0
'Dark Knight Rises, The"	0.5694444444444444	0.375	0.3333333333333333	1.0
'Dark Knight, The"	0.5238095238095238	0.5714285714285714	0.0	1.0
'Prestige, The"	0.5238095238095238	0.5714285714285714	0.0	1.0

COLLABORATIVE FILTERING – LEVERAGING MOVIE RATINGS

Misty Williams

1) Show all ratings by Misty Williams

	Movie	Rating
1	"Bad Boys"	4.0
2	"Blue Angel, The (Blaue Engel, Der)"	4.0
3	"Beavis and Butt-Head Do America"	3.0
4	"Seven Samurai (Shichinin no samurai)"	3.0
5	"Jerry Maguire"	3.0
6	"Benny & Joon"	3.0

2) Find Misty's average rating

AverageRating
3.5342789598108713

3) What are the movies that Misty liked more than average?

	Movie	Rating
1	"Raising Arizona"	5.0
2	"Jaws"	5.0
3	"Breaking the Waves"	5.0
4	"Nosferatu (Nosferatu, eine Symphonie des Grauens)"	5.0
5	"Cape Fear"	5.0
6	"Back to the Future"	5.0

COLLABORATIVE FILTERING – THE WISDOM OF CROWDS

Pour un utilisateur donné, quels sont les genres dont l'évaluation est supérieure à la moyenne ?
Utiliser ces données pour classer des films similaires

	Title	Genre	genreAvgRating	movieCount
1	"House of Games"	"Film-Noir"	4.333333333333334	9
2	"Bullet to the Head"	"Film-Noir"	4.333333333333334	9
3	"Tinker Tailor Soldier Spy"	"Film-Noir"	4.333333333333334	9
4	"Drive"	"Film-Noir"	4.333333333333334	9
5	"He Ran All the Way"	"Film-Noir"	4.333333333333334	9
6	"Nightfall"	"Film-Noir"	4.333333333333334	9

COSINE SIMILARITY

Find the users with the most similar preferences to Cynthia Freeman, according to cosine similarity

A la main

```
MATCH (u1:User {name: "Cynthia Freeman"})-[x:RATED]->(m:Movie)<-[y:RATED]-(u2:User)
WITH u1, u2, SUM(x.rating * y.rating) AS somme,
SQRT(SUM(x.rating * x.rating)) AS xtaille,
SQRT(SUM(y.rating * y.rating)) AS ytaille
RETURN u1.name AS User1, u2.name AS User2, somme / (xtaille * ytaille) AS sim
ORDER BY sim DESC;
```

User1	User2	sim
"Cynthia Freeman"	"Samuel Nguyen"	1.0000000000000002
"Cynthia Freeman"	"Mathew Smith"	1.0000000000000002
"Cynthia Freeman"	"Kimberly Edwards"	1.0000000000000002
"Cynthia Freeman"	"Michele Garcia"	1.0000000000000002

Fonction

```
// Récupérer les notes des films pour Cynthia Freeman
MATCH (u:User {name: 'Cynthia Freeman'})-[r:RATED]->(m:Movie)
WITH collect({movieId: m.movieId, rating: r.rating}) AS cynthiaRatings
// Récupérer les notes des films pour chaque autre utilisateur
MATCH (u2:User)-[r2:RATED]->(m2:Movie)
WITH u2, cynthiaRatings, collect({movieId: m2.movieId, rating: r2.rating}) AS otherRatings
WHERE u2.name <> 'Cynthia Freeman'
// Créer des vecteurs qui fonctionne
UNWIND cynthiaRatings AS cRating
UNWIND otherRatings AS oRating
WITH u2, cRating, oRating
WHERE cRating.movieId = oRating.movieId
WITH u2, collect(cRating.rating) AS cynthiaVector, collect(oRating.rating) AS otherVector
```

```
// Utiliser gds.similarity.cosine
WITH u2, cynthiaVector, otherVector
WHERE size(cynthiaVector) = size(otherVector) AND size(cynthiaVector) > 0
RETURN 'Cynthia Freeman' AS User1, u2.name AS User2,
gds.similarity.cosine(cynthiaVector, otherVector) AS cosineSimilarity
ORDER BY cosineSimilarity DESC
LIMIT 10;
```

User1	User2	cosineSimilarity
"Cynthia Freeman"	"Patricia Johnson"	1.0
"Cynthia Freeman"	"Linda Whitaker"	1.0
"Cynthia Freeman"	"Michael Stone"	1.0
"Cynthia Freeman"	"Jessica Wilson"	1.0

PEARSON SIMILARITY

Find the users with the most similar preferences to Cynthia Freeman, according to cosine similarity

A la main

```
MATCH (u1:User {name: "Cynthia Freeman"})-[x:RATED]->(m:Movie)<-[y:RATED]-
(u2:User)
WITH u1, u2, x.rating AS u1Rating, y.rating AS u2Rating
WITH u1, u2, AVG(u1Rating) AS avgU1, AVG(u2Rating) AS avgU2
MATCH (u1)-[x:RATED]->(m:Movie)<-[y:RATED]-(u2)
WITH u1, u2, x.rating AS u1Rating, y.rating AS u2Rating, avgU1, avgU2
WITH u1, u2, SUM((u1Rating - avgU1) * (u2Rating - avgU2)) AS haut,
SQRT(SUM((u1Rating - avgU1) ^ 2) * SUM((u2Rating - avgU2) ^ 2)) AS bas
RETURN u1.name AS User1, u2.name AS User2, haut / bas AS sim
ORDER BY sim DESC;
```

User1	User2	sim
"Cynthia Freeman"	"Leslie Brady"	1.0000000000000004
"Cynthia Freeman"	"Ian Brewer"	1.0000000000000002
"Cynthia Freeman"	"Ian Brewer"	1.0000000000000002
"Cynthia Freeman"	"Dawn Wood"	1.0000000000000002
"Cynthia Freeman"	"Hannah Armstrong"	1.0000000000000002

Fonction

```
// Récupérer les notes des films pour Cynthia Freeman
MATCH (u:User {name: 'Cynthia Freeman'})-[r:RATED]->(m:Movie)
WITH collect({movieId: m.movieId, rating: r.rating}) AS cynthiaRatings
// Récupérer les notes des films pour chaque autre utilisateur
MATCH (u2:User)-[r2:RATED]->(m2:Movie)
WITH u2, cynthiaRatings, collect({movieId: m2.movieId, rating: r2.rating}) AS otherRatings
WHERE u2.name <> 'Cynthia Freeman'
// Créer des vecteurs de notes compatibles
UNWIND cynthiaRatings AS cRating
UNWIND otherRatings AS oRating
WITH u2, cRating, oRating
WHERE cRating.movieId = oRating.movieId
WITH u2, collect(cRating.rating) AS cynthiaVector, collect(oRating.rating) AS otherVector
// Utiliser gds.similarity.pearson
WITH u2, cynthiaVector, otherVector
WHERE size(cynthiaVector) = size(otherVector) AND size(cynthiaVector) > 0
RETURN 'Cynthia Freeman' AS User1, u2.name AS User2,
gds.similarity.pearson(cynthiaVector, otherVector) AS pearsonSimilarity
ORDER BY cosineSimilarity DESC
LIMIT 10;
```

User1	User2	cosineSimilarity
"Cynthia Freeman"	"Ian Brewer"	1.0000000000000004
"Cynthia Freeman"	"Jessica Herring"	1.0
"Cynthia Freeman"	"Hannah Armstrong"	1.0
"Cynthia Freeman"	"Jake Mathews"	1.0
"Cynthia Freeman"	"Crystal Strong"	1.0

K-NEAREST NEIGHBORS

Who are the 10 users with tastes in movies most similar to mine? What movies have they rated highly that I haven't seen yet?

```
// Trouver les 10 utilisateurs les plus similaires à Cynthia Freeman
MATCH (u1:User {name: "Cynthia Freeman"})-[cRating:RATED]->(m:Movie)<-[oRating:RATED]-(u2:User)
WHERE u1 <> u2
WITH u2, collect(cRating.rating) AS cynthiaVector, collect(oRating.rating) AS otherVector
WHERE size(cynthiaVector) = size(otherVector) AND size(cynthiaVector) > 0
WITH u2, gds.similarity.pearson(cynthiaVector, otherVector) AS pearsonSimilarity
ORDER BY pearsonSimilarity DESC
LIMIT 10
// Recommander des films
MATCH (u1:User {name: "Cynthia Freeman"})
WITH u1, collect(u2) AS similarUsers
UNWIND similarUsers AS similarUser
MATCH (similarUser)-[r2:RATED]->(m:Movie)
WHERE NOT EXISTS((u1)-[:RATED]->(m))
WITH m, AVG(r2.rating) AS avgRating, COUNT(r2) AS numRatings
WHERE numRatings > 1
RETURN m.title AS Movie, avgRating
ORDER BY avgRating DESC
LIMIT 10;
```

Movie	avgRating
"Love and Death"	5.0
"Serpico"	5.0
"Blood Simple"	5.0
"Conversation, The"	5.0
"Gold Rush, The"	5.0