FINAL YEAR PROJECT

SECOND EVALUATION (SEM 8)

DEVAL MUDIA BT16CSE060
AYUSH SINGH BT16CSE098
CHAITYA CHHEDA BT16CSE016
SADNEYA PUSALKAR BT16CSE076

UNDER THE GUIDANCE OF
Dr. S.R. SATHE

PROBLEM STATEMENT

Recommendation System for Amazon and Netflix

USING BIG DATA AND BIG COMPUTE

INFRASTRUCTURE

WE ARE HERE

Raw Rating Data (~100 GB)

amazon

NETFLIX

movielens

Big Data

Preprocess

APACHE Spark

Utility Matrix

HYBRID FILTERING

Collaborative Filtering

SVD

Content Filtering

Cosine Similarity

python™

GRAPH FILTERING

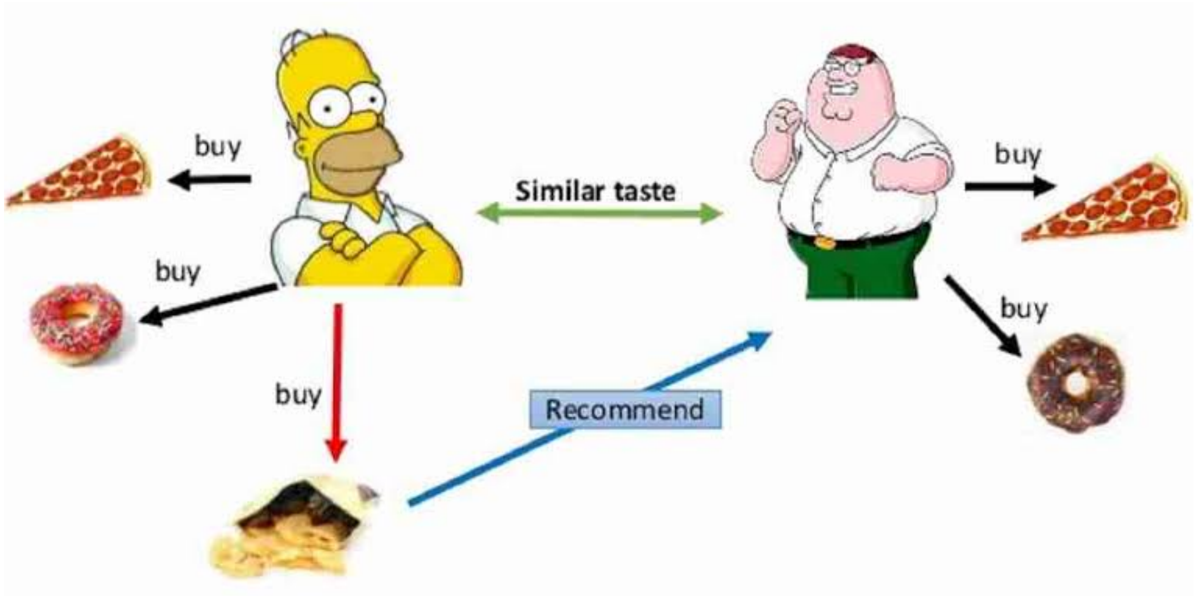
Bootstrapping Recommendations with

neo4j

Platform Development

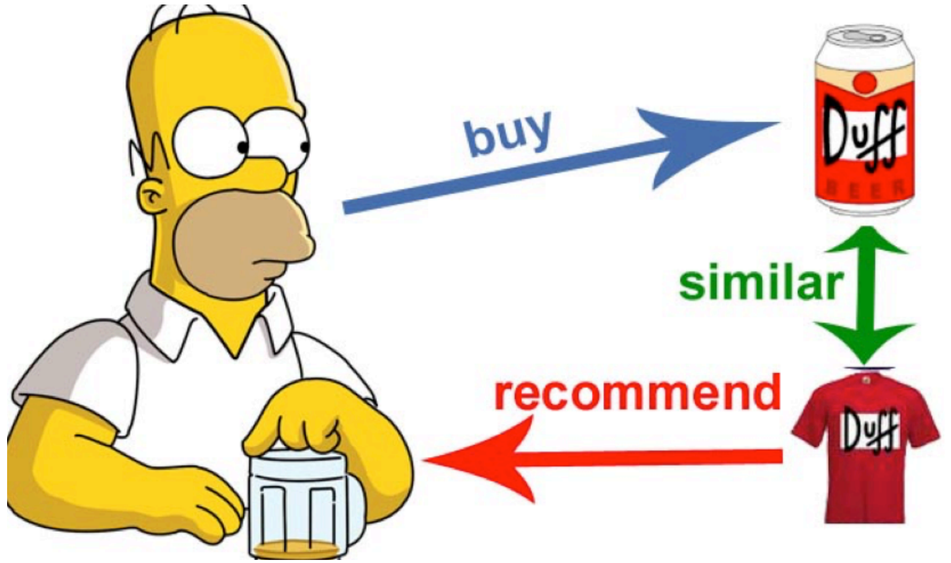
LAST EVALUATION

Collaborative Filtering



+

Content Filtering



PROBLEMS HANDLED

Sparsity: If the movie is not popular or just released, then it will have few or no ratings all all.

Cold Start Problem: If a user just joins the platform and hasn't reviewed any movies, there is no way to create a taste of the user to recommend movies to him/her.

Popularity Bias Problem: The above method ends up recommending the most popular movies, which does not add an extra value to all the users.

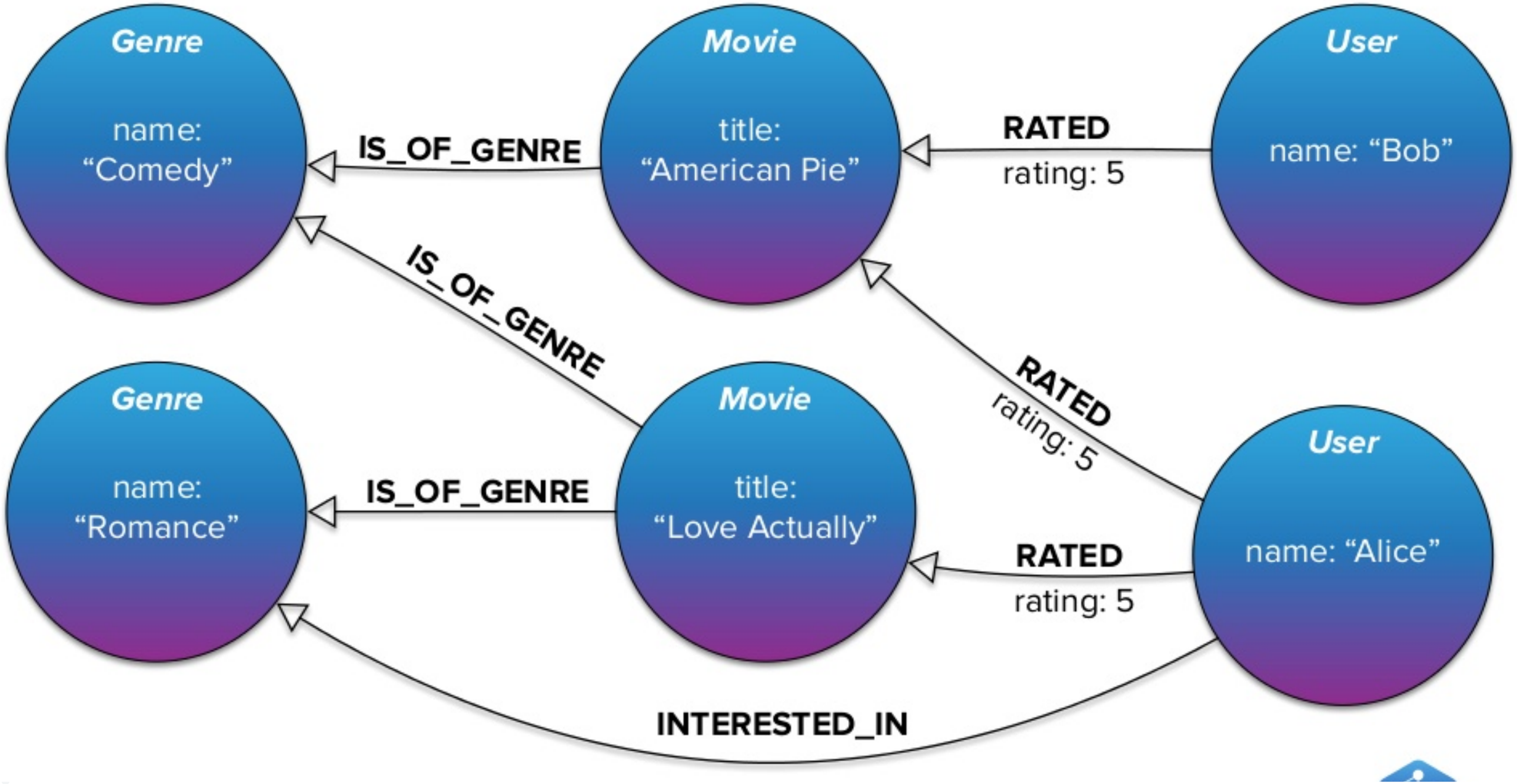
RECOMMENDATIONS WITH NEO4J

Neo4j is a graph database management system and an ACID-compliant transactional database with native graph storage and processing.

Uses **Cypher**, a declarative graph query language that allows for expressive and efficient data querying in a property graph.

```
1 MATCH
2 (u:User)-[:LIKED]->(m:Movie),
3 (m)<-[:LIKED]-(another:User),
4 (another)-[:LIKED]->(reco:Movie)
5 WHERE NOT (u)-[:LIKED|DISLIKED]->(reco)
6 RETURN reco;
```

Features (Movies, Users, Genres) as well as **relationships** can be naturally represented as a graph.



DESIGN

NODES:

1- **users** (userId): Created with using “ratings.csv” data. We create “Users” node and it will has relations with “Movies” and “Genres” nodes

2- **movies** (movieId, title, rating_mean): Created with using “movies.csv” data. “Movies” node will has relations with “Users” and “Genres” nodes and it has also relationship to itself based on similarity

3- **genres** (genres): It is small data it has 19 rows and it keeps genres

DESIGN

RELATIONSHIPS:

1- **users_movies** (userId, movieId, rating): Between “Users” and “Movies” nodes.

2- **movies_genres** (movieId, genres): Between “Movies” and “Genres” nodes.

3- **users_genres** (userId, genres): Between “Users” and “Genres” nodes. “genres” is a calculated field.

4- **movies_similarity** (movieId, sim_movieId, relevance): It is the most critical dataset in this pipeline. It includes 5 rows for each movies and they are similar movies ids.

CONTEXT WALK

ContextWalk, a recommendation algorithm based on Markov **random walks** that can combine ratings information with contextual information, such as tags, movie genre, users information to recommend movies.

In contrast to algorithms like, for instance, **PageRank** and **FolkRank**, we are not interested in the background probability of all nodes in the contextual graph by taking a random walk of infinite length. Instead, we **limit our random walks** to a finite length to keep the user in the vicinity of his original movie-related ‘recommendation need.

Use modified Graph Traversal Algorithms like Union Colours Algorithm or Modified Dijkstra’s Algorithm.

Combine the results of Content, Collaborative and Graph Filter.

Web Platform for end product development.

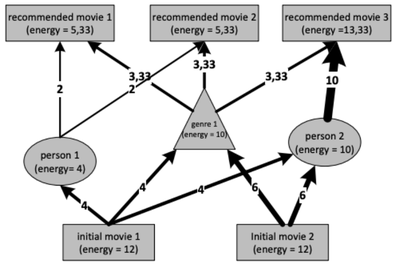


Figure 4. Visualization of Energy Spreading algorithm.

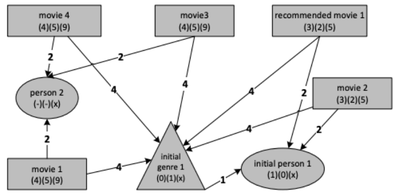


Figure 5. Visualization of Modified Dijkstra’s algorithm.

Visualisation of the algorithm is illustrated in Figure 4. The numbers on the arrows and their width represent the amount of energy which is being received by the receiving nodes. In the case shown in the figure, the movie number 3 is the first in the list of recommended nodes.

D. Modified Dijkstra’s Algorithm

The Modified Dijkstra’s Algorithm is, as the name suggests, based on the well known Dijkstra’s algorithms for finding the shortest path in a graph. We implement a variation on this algorithm, which works for multiple initial nodes.

1) Run the Dijkstra’s shortest path algorithm from each node to (some constant) maximal depth - calculate the shortest path from the starting node to each visited node.
   a) If the algorithm is ran from the first initial node, put all visited nodes with the value of the shortest path as their total value into a results set.
   b) If the algorithm is ran from other initial nodes, check if the node is in the results set. If yes, add the shortest path value to its total value.
2) Check if each node in the results set was visited from all initial nodes.
3) Order the nodes in the results set by their total values of the shortest paths.
4) Return the required number of nodes from the ordered results set.

V. Experimental Evaluation

The goal of the experimental evaluation we performed was to determine the best algorithm of the four algorithms we designed and to determine the accuracy of each algorithm. Based on the results of the experiment we intend to work on the most successful algorithms in the future and further modify them to make their results even better.

We filled the databases with a dataset consisting of information about a large amount of real movies and TV programs. In the end, the graph database consisted of roughly 165 000 nodes and 870 000 relations.

The experiment we used to test our methods was based on collecting explicit feedback from users. The user’s task was to pick some initial nodes – movies, people or genres. The system generated 4 different sets of recommendations based on the initial nodes using the four recommendation algorithms we designed. We presented each user with 7 different scenarios, in each scenario the user was supposed to pick initial nodes in a different way, for example in the first scenario we asked him to pick two animated movies. The scenarios were implemented into the experiment to ensure the recommendation algorithms were tested on their versatility. After selection of the initial nodes each recommendation algorithm recommended five movies to the user; the user proceeded to order the algorithms based on their accuracy and was also asked to rate the most accurate algorithm on a scale from 1 to 5. In total, 30 users participated in the experiment. However, not all of the users who began the experiment completed every scenario. In total, 168 scenarios were completed by the users and recorded to our database, which, on average, works out to

References

Course : Recommender Systems (University of Minnesota)
https://www.coursera.org/specializations/recommender-systems#courses

http://ids.csom.umn.edu/faculty/gedas/cars2010/Bogers-CARS-2010.pdf

http://www2.fiit.stuba.sk/~bielik/publ/abstracts/2013/televido-dexa2013.pdf

https://github.com/beckernick/matrix_factorization_recommenders

https://hackernoon.com/introduction-to-recommender-system-part-1-collaborative-filtering-singular-value-decomposition-44c9659c5e75