MOVIE RECOMMENDATION ENGINE USING SPARK MLLIB

Ramgopal Reddy Putta

WHAT IS MOVIE RECOMMENDATION ENGINE ?

- This tool makes personalized movie recommendations tailor for the user.
- Movie is recommended based on his/her past viewed movies and also based on other users with similar interest.
- This engine is build using Spark MLlib tool.

ABOUT THE DATASET:

- The dataset contains 1 million ratings from 6000 users on 4000 movies.
- 3 datasets are used in this analysis.

- **MOVIES:**

MovieID::Title::Genres

Eg: (1::Toy Story (1995)::Animation|Children's|Comedy)

- **RATINGS:**

UserID::MovieID::Rating::Timestamp

Eg: (1::1193::5::978300760)

- **USERS:**

UserID::Gender::Age::Occupation::Zip-code

Eg: (2::M::56::16::70072)

SPARK MLLIB

- MLlib is the spark’s scalable machine learning library consisting of common Machine learning algorithms.
- In this project, ‘Collaborative Filtering’ technique is implemented for creating recommendations.

COLLABORATIVE FILTERING:

- Also known as Social filtering, this technique aims to fill the missing entries of a user-item association matrix (user-movie rating matrix) in which users and products (movies) are described by a set of latent factors.

- To implement this technique in this project, Alternate Least Square (ALS) algorithm is used to learn these factors.

- ALS is an iterative algorithm where in each iteration, the algorithm fixes one factor matrix and solves for other. This process continues till it converges.

(This alternation between which matrix to optimize is where the "alternating" in the name comes from.)

BASIC IDEA OF COLLABORATIVE FILTERING:

	USER 1	USER 2	USER 3
MOVIE 1	4	2	5
MOVIE 2	2	5	1
MOVIE 3	5	1	4
MOVIE 4	4	5	4
MOVIE 5	5	4	

Here, the interests of USER 3 seem similar to USER 1
Hence, MOVIE 5 can be recommended for USER 3

CHALLENGES OF COLABORATIVE FILTERING:

- **Data Sparsity:**

	LARGE DATASET	SMALL DATADET
SPEED	👎	👍
ACCURACY	👍	👎

- **Scalability**

Increase in volume of users will decrease the performance of the algorithm.

CHALLENGES OF COLABORATIVE FILTERING:

- **Synonym:**

The recommendation system treats a same product differently if the names are different, even thought functionally they are similar to each other.

LIBRARIES IMPORTED:

- SQLContext which is an entry point for working with structured data.

```
val sqlContext = new org.apache.spark.sql.SQLContext(sc)
```

- This library implicitly converts an RDD to a DataFrame.

```
import sqlContext.implicits._
```

- Importing Spark SQL data types

```
import org.apache.spark.sql._
```

- Importing MLLIB data types

```
import org.apache.spark.mllib.recommendation.{ALS, MatrixFactorizationModel,
Rating}
```

IMPLEMENTING THE ALS ALGORITHM:

- We run ALS on the input trainingRDD of Rating(user, product, rating) objects with the rank and Iterations parameters.

- The ALS run(trainingRDD) method will build and return a MatrixFactorizationModel, which can be used to make product predictions for users.

OTHER FAMOUS RECOMMENDATION ENGINES:

Content based filtering:

- This Recommender relies on the similarity of the items being recommended.
- It generally works well when it’s easy to determine the context/properties of each item.

REFERENCES:

- Intellipat.com
- https://youtu.be/wTZUHiLUY94
- The 4 Recommendation Engines That Can Predict Your Movie Tastes | by James Le | Towards Data Science
- Movie Recommendation with Spark MLlib - Databricks

<u>THANK YOU</u>