A Simple Movie Recommender System using Collaborative Filtering

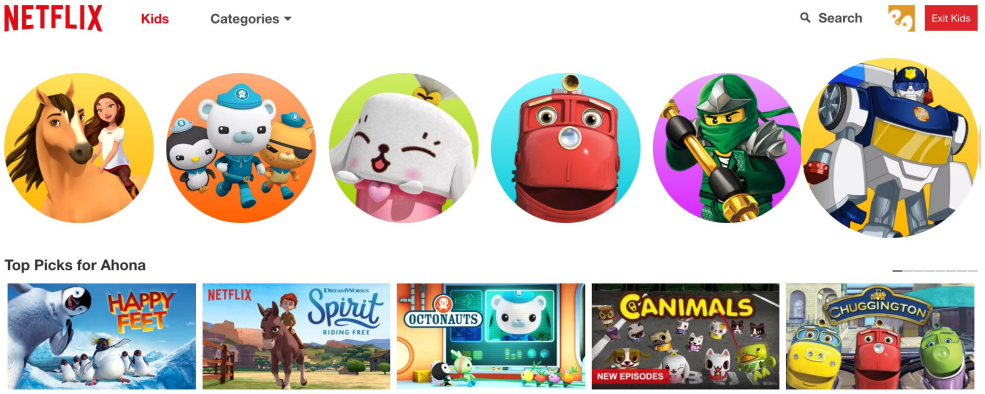
Tazmina Sharmin
Yangcha K. Ho

Overview

- Introduction to Recommender Systems and Collaborative Filtering
- Dataset
- Flowchart
- Implementation Basics
- Model Evaluation
- Lessons Learned

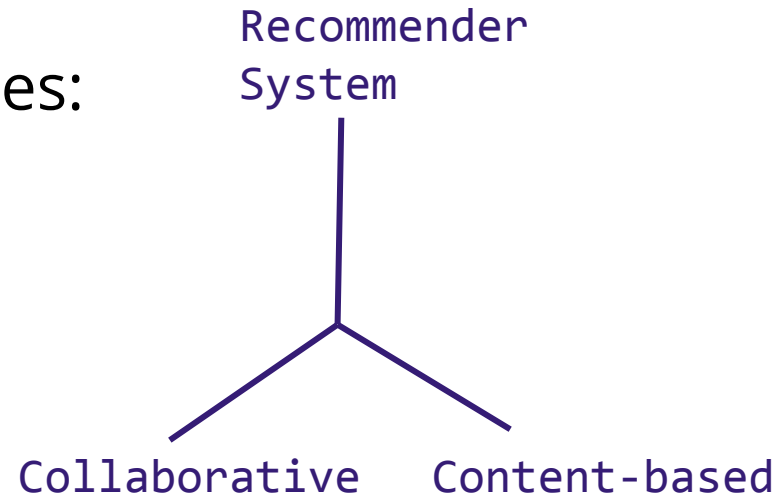
What is a Recommender System?

- Subclass of Information Filtering System.
- Helps users to decide: what item to buy/ what stock to purchase etc.
- Applied in various applications.



Approaches

Two approaches:



Predicts individual's preference based on their peer ratings

Predicts individual's preference based on the information on the content of items

Introduction to Collaborative Filtering

Recommendation systems based on collaborative filtering use information about a user's preferences to make personalized predictions about content, such as topics, people.

Algorithms that CF systems employ are:

- How types of ratings in a CF system affect choices
- How to evaluate and compare recommenders
- Trends in the development of interactive and social interfaces

Core concepts of Collaborative Filtering

We introduce the core concepts of CF, its primary uses for users of the adaptive web, the theory and practice of CF algorithms.

As the volume of accessible information and active users on the Internet continues to grow, it becomes difficult to compute recommendations quickly and accurately over a large dataset.

The speed of computers allows us to process these opinions and develop a personalized view of that item using the opinions most appropriate for a given user/users.

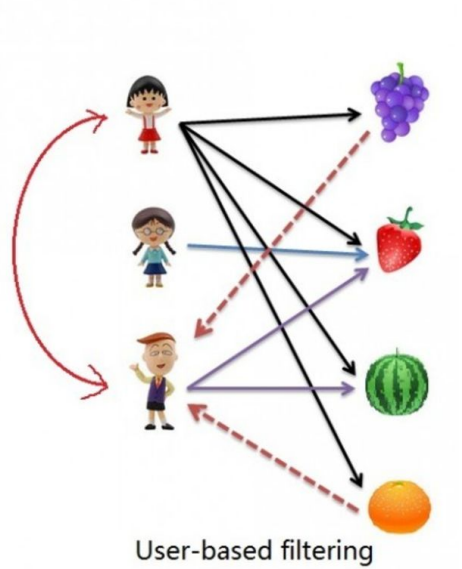
Examples

- Help me find new items I might like
- Help me to find a specialty school
- Help me find a user/some users to share – find a person with a rare disease such as pre-aged person syndrome

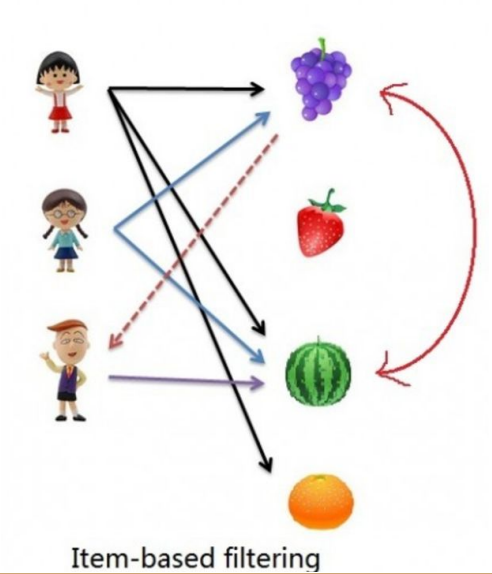
Collaborative Approaches

Collaborative filtering algorithms are divided into two groups:

1) User-based CF:



2) Item-based CF:



User-based Collaborative Filtering

- User-based CF tries to mimics word-of- mouth by analyzing rating data from many individuals.

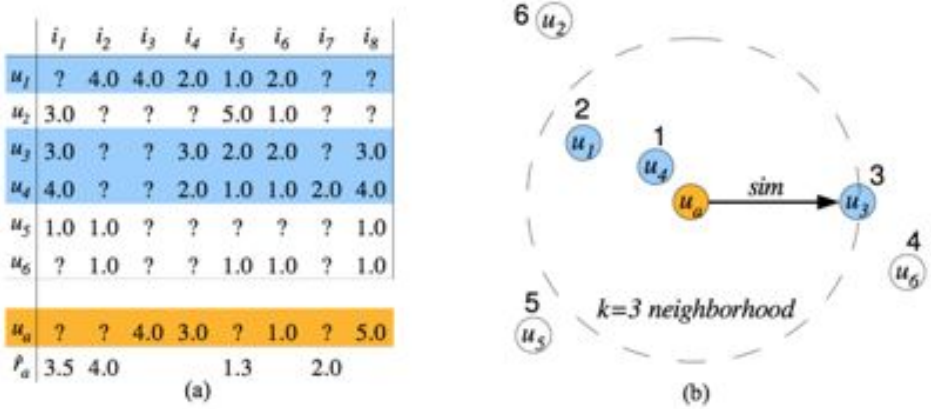
- The idea is that users with similar preferences will rate items similarly. Thus missing ratings for a user can be predicted by first finding a neighborhood of similar users and then aggregate the ratings of these users to form a prediction.

User-based Collaborative Filtering

The easiest form is to just average the ratings in the neighborhood.

$$\hat{r}_{aj} = \frac{1}{|\mathcal{N}(a)|} \sum_{i \in \mathcal{N}(a)} r_{ij}$$

User based Collaborative Filtering



	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8
u_1	?	4.0	4.0	2.0	1.0	2.0	?	?
u_2	3.0	?	?	?	5.0	1.0	?	?
u_3	3.0	?	?	3.0	2.0	2.0	?	3.0
u_4	4.0	?	?	2.0	1.0	1.0	2.0	4.0
u_5	1.0	1.0	?	?	?	?	?	1.0
u_6	?	1.0	?	?	1.0	1.0	?	1.0
u_a	?	?	4.0	3.0	?	1.0	?	5.0
$\hat{r}_a$	3.5	4.0			1.3		2.0	

(a)

(b)

Figure: User-based collaborative filtering example with (a) rating matrix and estimated ratings for the active user, and (b) user neighborhood formation.

Item-based Collaborative Filtering

This is a model-based approach which produces recommendations based on the relationship between items inferred from the rating matrix.

The idea is that users will prefer items that are similar to other items they like.

It builds similarity matrix containing all item-to- item similarities and are stored in a n×n similarity matrix S.

To reduce the model size to n×k with $k \ll n$, only a list of the k most similar items and their similarity values are stored.

Item-based Collaborative Filtering

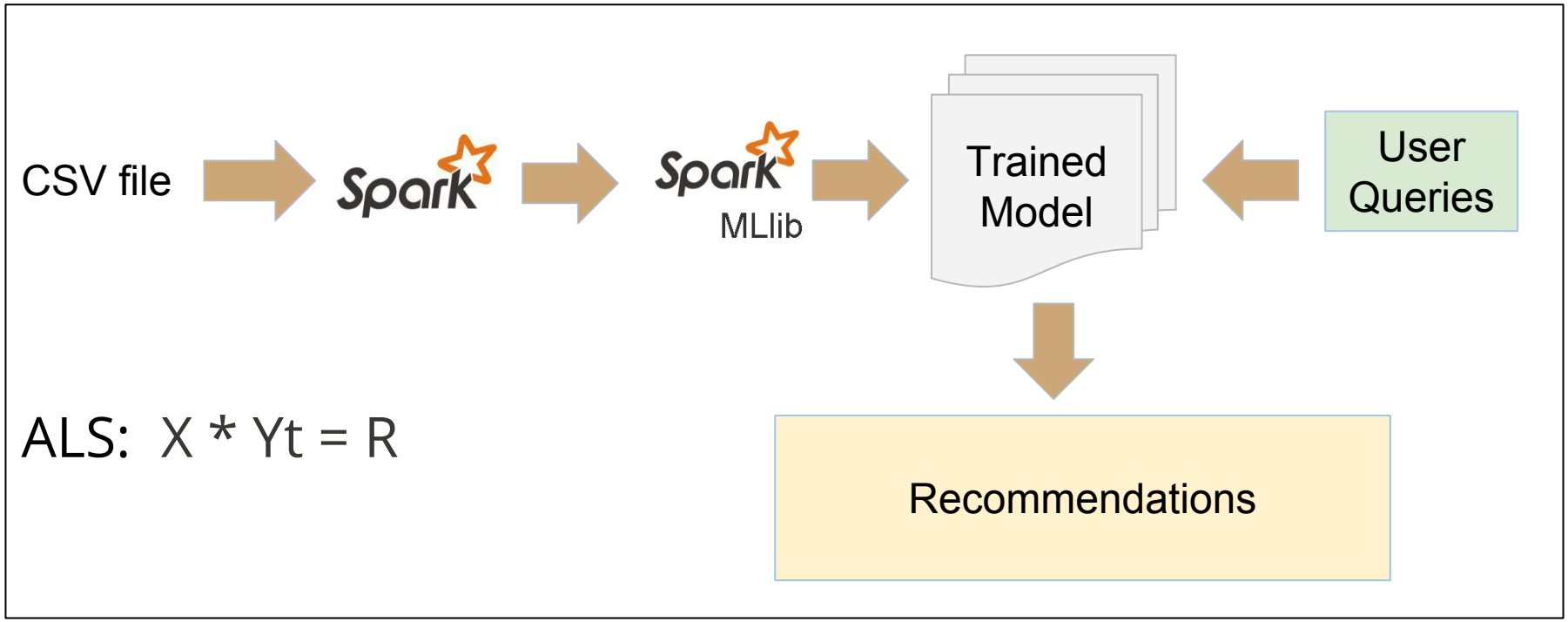
S	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	$\hat{r}_a$	$k=3$
i_1	-	0.1	0	**0.3**	**0.2**	**0.4**	0	0.1	-	
i_2	0.1	-	**0.8**	**0.9**	0	**0.2**	0.1	0	0.0	
i_3	0	**0.8**	-	0	**0.4**	0.1	0.3	**0.5**	4.6	
i_4	**0.3**	**0.9**	0	-	0	0.1	0	**0.2**	3.2	
i_5	**0.2**	0	**0.4**	0	-	0.1	**0.2**	0.1	-	
i_6	**0.4**	**0.2**	0.1	**0.3**	0.1	-	0	0.1	2.0	
i_7	0	**0.1**	**0.3**	0	**0.2**	0	-	0	4.0	
i_8	**0.1**	0	**0.5**	**0.2**	0.1	0.1	0	-	-	
u_a	2	?	?	?	4	?	?	5		

Figure: Item-based collaborative filtering

Dataset

- Movielens Dataset with 100000 ratings
- Downloaded from grouplens.org
- csv format

High Level Architecture



Implementation

- Loading and Parsing the dataset

```
JavaRDD<String> ratingRdd = jsc.textFile(ratingPath);  // load rating file
JavaRDD<Rating> ratings = ratingRdd.map(s -> {
    String[] sarray = s.split( regex: ",");
    return new Rating(Integer.parseInt(sarray[0]),  // user
            Integer.parseInt(sarray[1]),            // movie
            Double.parseDouble(sarray[2]));         // rating
});
```

- Building the recommendation model using ALS

```
int rank = 8;
int numIterations = 10;

MatrixFactorizationModel model = ALS.train(
    JavaRDD.toRDD(ratings),
    rank,
    numIterations,
     v: 0.01
);

System.out.println("Model has been trained" +
        ", num ratings: " + ratings.collect().size() +
        ", num movies: " + movies.collect().size());
```

Implementation

- Finding unseen movies with the model predicted ratings for the selected users

```
List<Tuple2<Tuple2<Integer, Integer>, Double>> predictions =
    model.predict(JavaRDD.toRDD(userUnseenMovies)).toJavaRDD()
            .map(r -> new Tuple2<>(new Tuple2<>(r.user(), r.product()), r.rating()))
            .sortBy(Tuple2::_2,  ascending: false,  numPartitions: 10)
            .take( num: 10);
```

- Sort and find the best picks

```
predictions.forEach(r -> {
    int movieId = r._1._2;
    String movieName = movies.filter(m -> m._1 == movieId).first()._2();  // find movie name
    System.out.println("User: " + r._1._1 + ", Movie: " + movieName + ", Rating: " + r._2);
});
```

Sample Output

```
User: 10, Movie: Breaking the Waves (1996), Rating: 7.598940328859921
[mapr@maprdemo recommender]$ /opt/mapr/spark/spark-2.1.0/bin/spark-submit --class recommender.MovieRecommender CS185-jar-with-dependencies.jar data/ml-latest-small/ratings.csv data/
ml-latest-small/movies.csv data/ml-latest-small/users_query.csv
17/05/10 22:06:34 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeSystemBLAS
17/05/10 22:06:34 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeRefBLAS
17/05/10 22:06:34 WARN LAPACK: Failed to load implementation from: com.github.fommil.netlib.NativeSystemLAPACK
17/05/10 22:06:34 WARN LAPACK: Failed to load implementation from: com.github.fommil.netlib.NativeRefLAPACK
17/05/10 22:06:36 WARN Executor: 1 block locks were not released by TID = 56:
[rdd_214_0]
17/05/10 22:06:36 WARN Executor: 1 block locks were not released by TID = 57:
[rdd_215_0]
Model has been trained, num ratings: 100004, num movies: 9125
User: 10, Movie: Kung Pow: Enter the Fist (2002), Rating: 8.888005073352936
User: 10, Movie: "Secret in Their Eyes, Rating: 7.468318363483336
User: 10, Movie: Homegrown (1998), Rating: 7.461458065897339
User: 10, Movie: Coma (1978), Rating: 7.354712778282636
User: 10, Movie: "Sorrow and the Pity, Rating: 7.042441900089509
User: 10, Movie: Clash of the Titans (2010), Rating: 6.950717345505156
User: 10, Movie: "Believers, Rating: 6.855650788154164
User: 10, Movie: State and Main (2000), Rating: 6.698940131997768
User: 10, Movie: Hannah and Her Sisters (1986), Rating: 6.694676040310749
User: 10, Movie: Earthquake (1974), Rating: 6.450161262623798
```

How good is our Recommender System?

Evaluation of predicted Ratings -

- Mean Absolute Error: $\frac{1}{n}\sum_{u,i}\left|p_{u,i} - r_{u,i}\right|$
- Root Mean Squared Error: $\sqrt{\frac{1}{n}\sum_{u,i}(p_{u,i} - r_{u,i})^2}$

How good is our Recommender System?

We tried different tuning parameters:

- Number of Iterations
- Lambda
- Rank

We found out that for our dataset, using number of iterations = 10, Lambda = 10 (default) and Rank = 8, we get the best Root Mean Squared Error of 0.880732.

Lessons Learned

- There needs to be enough other users already in the system to find a match.
- Cannot recommend an item that has not been previously rated (New items, esoteric items)
- Cannot recommend items to someone with unique tastes. Tends to recommend popular items.

Future Works

- The challenges to privacy and trust within CF systems
- The evaluation on different types of social network is needed.

Questions?

References

Extracted Data from “MovieLens Latest Dataset”https://grouplens.org/datasets/movielens/

Pham, Manh Cuong, et al. "A clustering approach for collaborative filtering recommendation using social network analysis." *J. UCS* 17.4 (2011): 58

Schafer, J. H. J. B., et al. "Collaborative filtering recommender systems." *The adaptive web* (2007): 291-324.3-604.

https://dzone.com/articles/improved-r-implementation-of-collaborative-filteri-1

https://dzone.com/articles/improved-r-implementation-of-collaborative-filteri-1

https://www.google.com/patents/US5704017

Thank You!