ONLINE MOVIE RECOMMENDER SYSTEM

Final Project COMSE 6998 Cloud Computing and Big Data

Mick Lin Yin Jun Chinghui Lin Sihan Wu Yitong Wang Jingmei Zhao Dingyu Yao Jingtao Zhu Yitong Xu Yaning Yu Shao Qi

Abstract

Use the MovieLens dataset to build a movie recommender using collaborative filtering with Spark's Alternating Least Squares implementation. First, get and parse movies and ratings data into Spark RDDs. Then we build and use the recommender and persist it for later use in our on-line recommender system.

Keywords-movie recommendation; Collaborative Filtering; Alternating Least Squares; Spark.

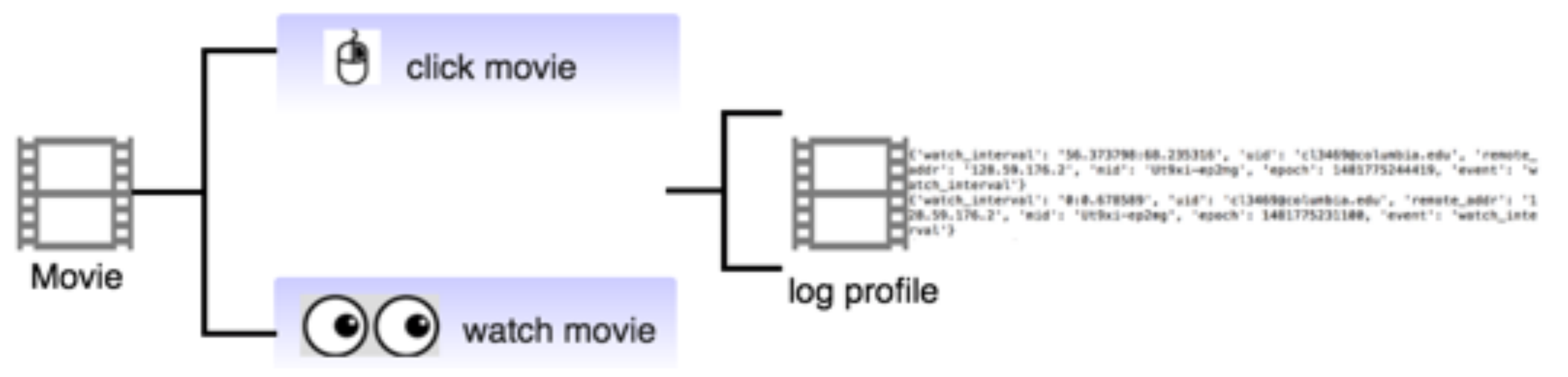
Recommendation Dataset

MovieLens 20M dataset contains 100004 ratings and 1296 tag applications across 9125 movies. These data were created by 671 users between January 09, 1995 and October 16, 2016. This dataset was generated on October 17, 2016. Users were selected at random for inclusion, and all selected users had rated at least 20 movies.

movies.csv contains movie information:

{movieId,title,genres}

ratings.csv contains rating information:

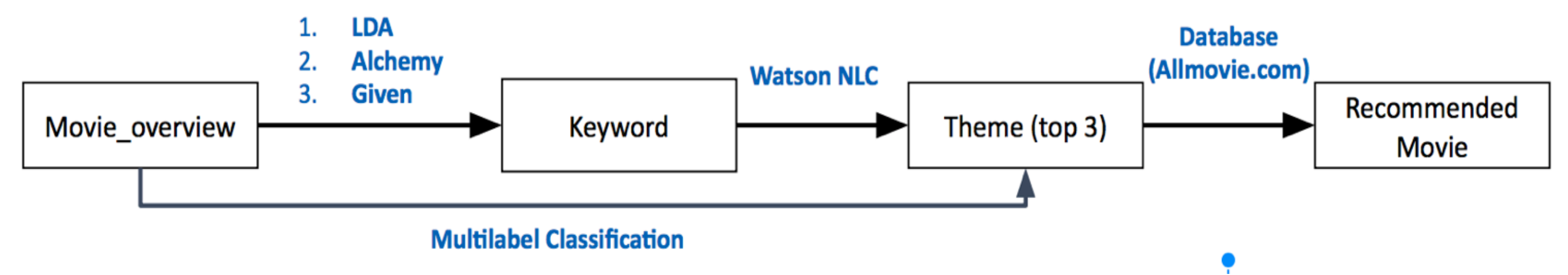


```
{'watch_interval': '56.373798:60.235316', 'uid': 'cl3469@columbia.edu', 'remote_addr': '128.59.176.2', 'mid': 'Ut9xi-ep2ng', 'epoch': 1481775244419, 'event': 'watch_interval'}
{'watch_interval': '0:0.678589', 'uid': 'cl3469@columbia.edu', 'remote_addr': '128.59.176.2', 'mid': 'Ut9xi-ep2ng', 'epoch': 1481775231100, 'event': 'watch_interval'}
```

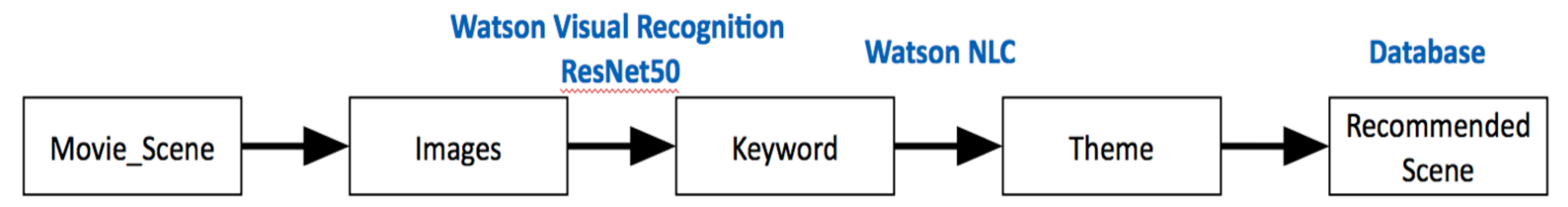
Figure 1 Data Model

Metadata for Movie and Scene

Movie Level: : the dataset we use training movie from Allmovie.com . The schema we train is as follows:



Scene Level: the data we use from 48 trailer videos. The schema we train is as follows:



Collaborative Filtering

Collaborative Filtering (CF) is a method of making automatic predictions about the interests of a user by learning its preferences (or taste) based on information of his engagements with a set of available items, along with other users' engagements with the same set of items.

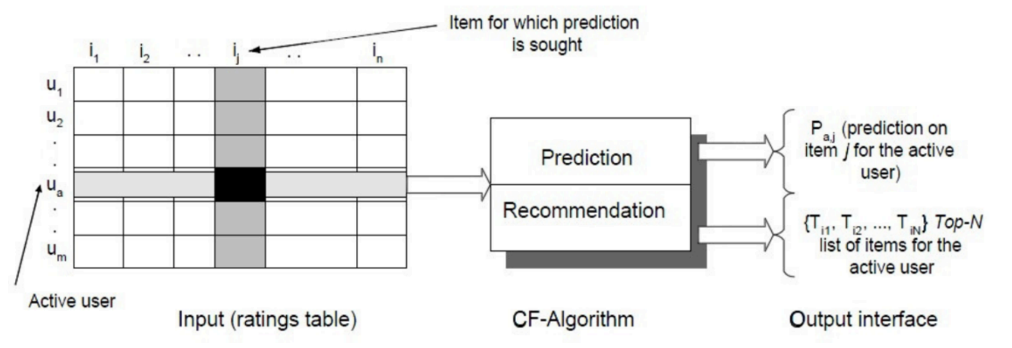

Item for which prediction is sought
i_1 i_2 .. i_j .. i_n
u_1 u_2 u_a u_m
Active user
Prediction
Recommendation
$P_{a,j}$ (prediction on item j for the active user)
$\{T_{i1}, T_{i2}, ..., T_{iN}\}$ Top-N list of items for the active user
Input (ratings table)
CF-Algorithm
Output interface

Figure 2 Overview of collaborative filtering process

We have users u for items i matrix: where r is what rating values can be.

$$Q_{ui} = \begin{cases} r & \text{if user u rate item i} \\ 0 & \text{if user u did not rate item i} \end{cases}$$

Depend on only the movies that have ratings from the users and do not make any assumption around the movies that are not rated in the recommendation.

$$w_{ui} = \begin{cases} 0 & \text{if } q_{ui} = 0 \\ 1 & \text{else} \end{cases}$$

Cost functions we are trying to minimize:

$$J(x_u) = (q_u - x_u Y) W_u (q_u - x_u Y)^T + \lambda x_u x_u^T$$
$$J(y_i) = (q_i - X y_i) W_i (q_i - X y_i)^T + \lambda y_i y_i^T$$

Finally, solutions for factor vectors are given:

$$x_u = (Y W_u Y^T + \lambda I)^{-1} Y W_u q_u$$
$$y_i = (X^T W i X + \lambda I)^{-1} X^T W_i q_i$$

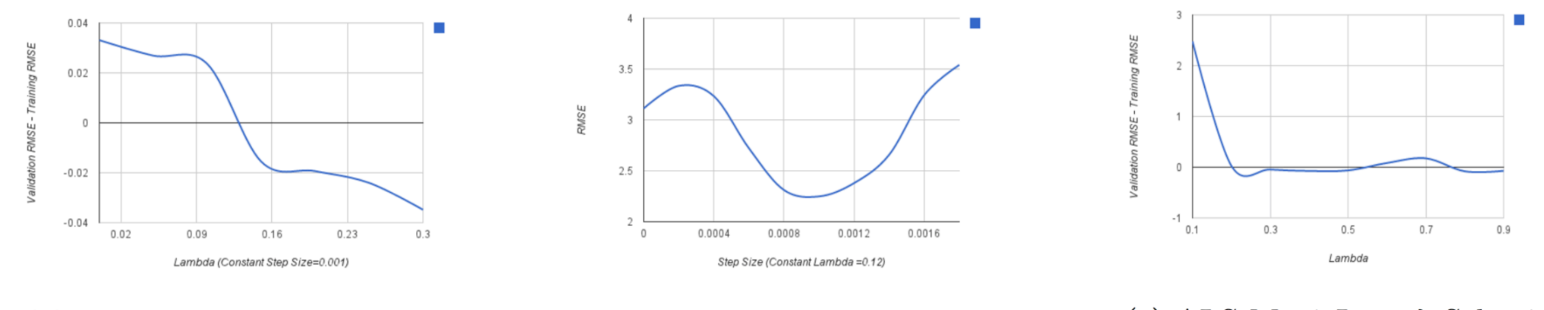
In the regularization, we may want to incorporate both factor matrices in the update rules as well if we want to be more restrictive.

Rating Algorithm

Parse user logs to ratings:
We initially assign 0 to each video for an user. There are 36 videos in our case.
Given a video m1 length S seconds and Q for watched seconds, we get a unit of ¼ if the user i watched ¼ S seconds on this video. We emphasize its weight by multiplying 3, simply 3Q/S.
When an user clicked on a video, we add one unit for the video for that user.
Finally, for each user, we fetched the user logs from SQS and stored all the parsed ratings in memory. Then, we feed the ratings to movie recommendation pyspark.

Algorithm Evaluation

In order to select valid λ and SGD gradient step sizes, we run a parameter sweep to find which values gave me the lowest RMSE on dataset. The parameter sweeps is shown in Figure 3. Table 1 shows the final error of all algorithms while using the best observed parameters.



(a) SGD MovieLens λ Selection (b) SGD MovieLens Step Size Selection (c) ALS MovieLens λ Selection

Figure 3 (Validation of parameter choices for MovieLens dataset)

SGD		B-SGD		ALS	
Training	Validation	Training	Validation	Training	Validation
2.20	2.23	1.01	1.02	0.66	0.92

Table 1 Training and Validation on different Algorithms

Alternating least squares performs better than other algorithms on the MovieLens dataset which is extremely sparse, providing with higher accuracy and better scaling.

User-based Online Recommendation

Motivation: Online recommendation need to solve the 'cold start' problem, and we cannot assume movies had been rated before!

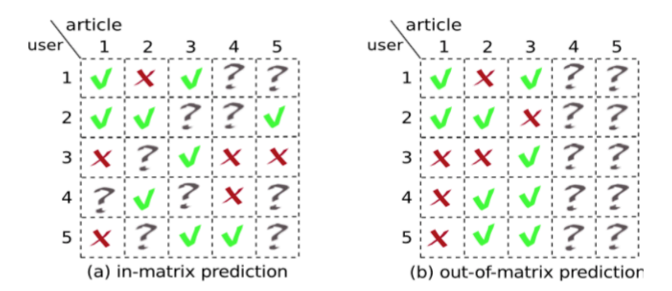
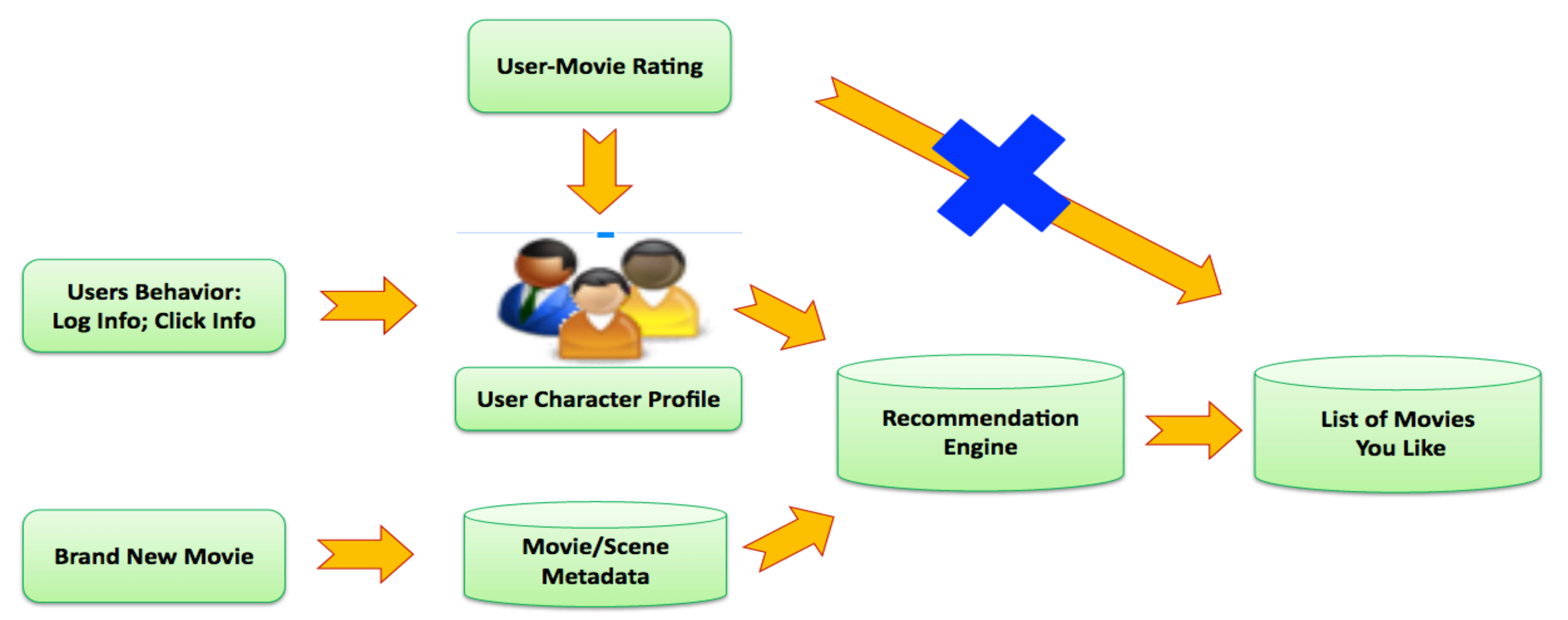


Figure 1: Illustration of the two tasks for scientific article recommendation systems, where √ indicates "like", × "dislike" and ? "unknown".

Instead of factorizing the user-rating matrix by SVD, and directly recommend user the movie, we decide to use the user-movie rating data to generalize user's character.

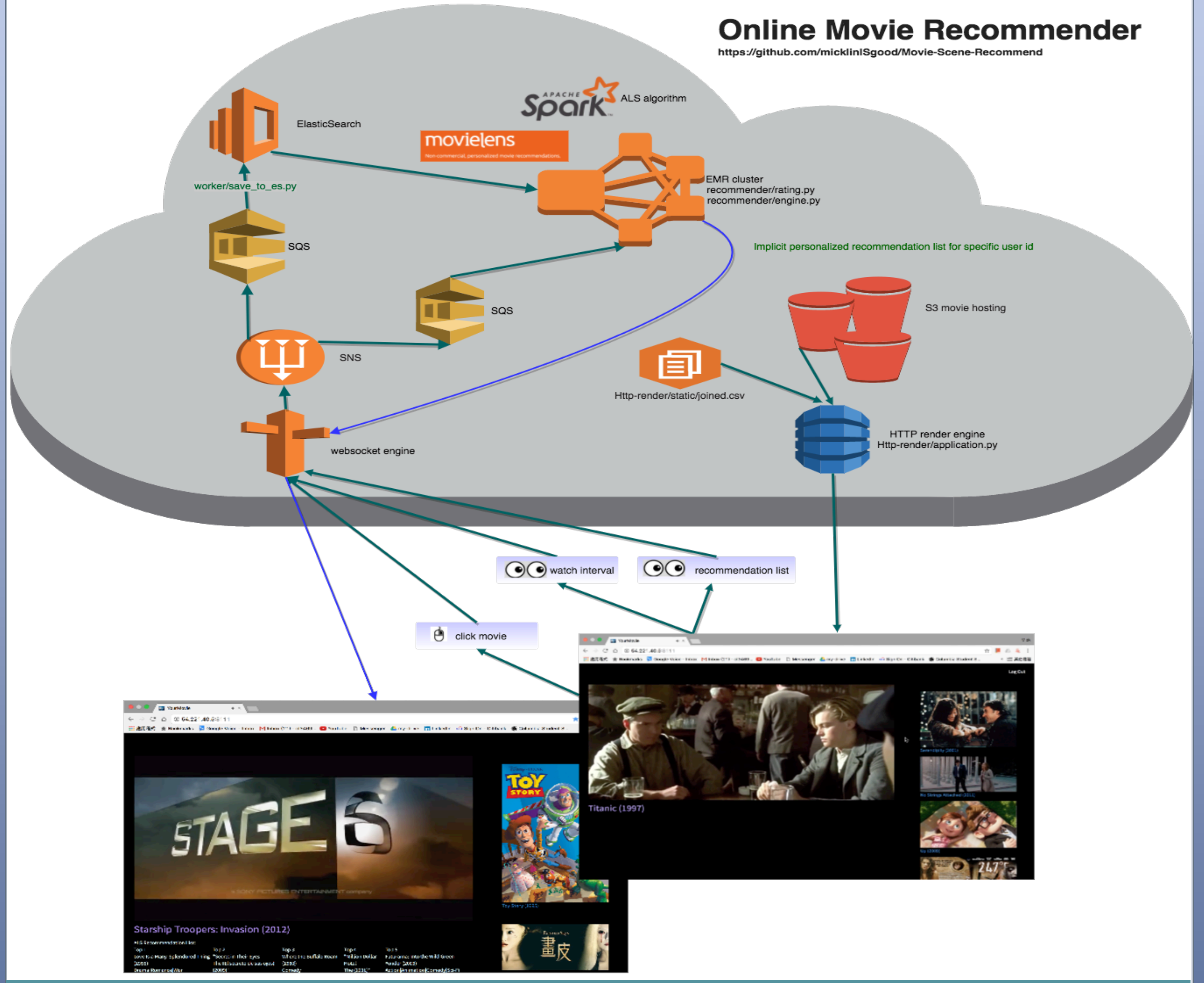
Online Scene-Movie Recommendation Model



Conclusion

In this project, we implement collaborative filtering for movie recommendation. Besides that, we create metadata for both user and movie/scene and design a more robust recommender engine to solve the 'cold start' problem. It allows a user to select his choices from a list of movies and then recommend him movies/scenes based on our recommender algorithm. By the nature of our system, it is not an easy task to evaluate the performance since there is no right or wrong recommendation; it is just a matter of opinions. Based on informal evaluations that we carried out over a small set of users we got a positive response from them. We would like to have a larger data set that will enable more meaningful results using our system.

Cloud Technology



Reference

[1] Movielens 10m network dataset - KONECT, Nov. 2014.
[2] Z. Huang, D. Zeng, and H. Chen. A comparison of collaborative-filtering recommendation algorithms for e-commerce. IEEE Intelligent Systems, 22(5):68–78, Sept. 2007.
[3].C. Wang and D. M. Blei. Collaborative topic modeling for recommending scientific articles. In KDD, pages 448–456, 2011.
[4].K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385, 2015.