Recommendation System

Zhenghui Li
Group Leader
155 Arbor Glen Dr
Lizheng9@msu.edu

Zengze Du
Group Member
2950 Mount Hope Rd
duzengze@msu.edu

Zhiqi Fu
Group Member
155 Arbor Glen Dr
fuzhiqi@msu.edu

ABSTRACT

This project aims to build an integrated recommender system with versatile features based on the Amazon reviews dataset.

Keywords
Amazon Ratings Analysis, Collaborative Filtering, Recommender System, Web Interface

1. INTRODUCTION

Recommendation systems play a vital role in our daily life. Each time Instagram suggests a friend, Amazon shows new product, Netflix recommends a new movie, etc. Automated systems help people filter irrelevant the information and provide suggestions to users. This project tries to solve problems confronting present recommender systems such as how to improve the degree of automation, how to make algorithms run faster and more robust with parallel computation.

In the Lecture 4, We learnt how to store data into a relational database. In our project, we use database to store the user information, product information review and recommend items. In the Lecture 16, We have learnt Collaborative Filtering and Recommender System.

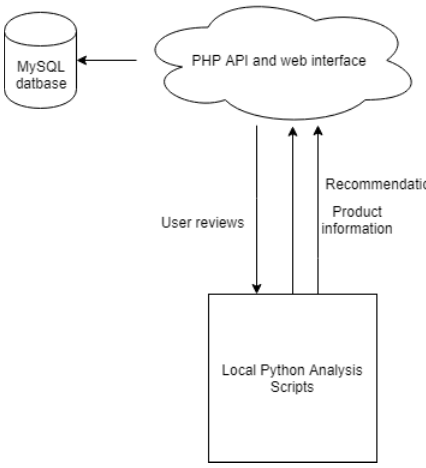
When developing web interface of amazon product recommendation system, we could not find a way to run python script code dynamically in PHP to get the updated recommendation result in our website.

Mean Absolute Error (MAE) and Root mean squared error (RMSE) are two of the most common metrics used to measure accuracy for continuous variables. We choose to use matrix factorization SVD because it generates the lowest MAE.

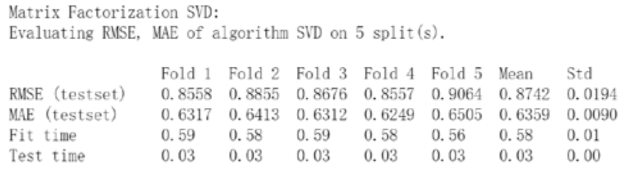
2. PRELIMINARIES

The team has collected the Amazon product data from the website provided in the instruction. The original data, which is json formatted, covers selected reviews from certain product category. The team picks musical instruments as selected experiment data because the shorter data size makes the process time shorter and the data is significantly easier to manage in personal computers. The original data contains Amazon product id (asin), helpful count, overall rating, review text, review time, reviewer ID, reviewer name, review summary, and unix formatted review time per review. The team plans to divide the analysis into two parts including rating-based and review-based recommendation system. The team picks Python Surprise toolkit to perform calculations and analysis. Surprise is a Python scikit building and analyzing recommender systems. The team plans to apply KNN with ratings of each user, Baseline only method, Matrix Factorization SVD, and Matrix Factorization NMF to the experiment dataset and compare the performance between different algorism in term of error rate (RMSE and MAE) and time consumption.

3. METHODOLOGY



For preprocessing part, the team keeps the reviewerID, asin, and overall rating for the rating-based recommendation part. Based on the dataset the team choose, there is no need of limiting the analysis to users who have rated sufficiently many products in the dataset because the team did not encounter performance problem while processing the dataset. Loading data to surprise Dataset using Reader is easy after the preprocessing because the ease of use of Surprise Tookit. After preprocessing, the team needs to compare between different algorisms using cross validation with 5 folds.



```
Matrix Factorization SVD:
Evaluating RMSE, MAE of algorithm SVD on 5 split(s).

                 Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)   0.8558  0.8855  0.8676  0.8557  0.9064  0.8742  0.0194
MAE (testset)    0.6317  0.6413  0.6312  0.6249  0.6505  0.6359  0.0090
Fit time         0.59    0.58    0.59    0.58    0.56    0.58    0.01
Test time        0.03    0.03    0.03    0.03    0.03    0.03    0.00
```

Matrix Factorization SVD is decided to be the best algorithm performance wise.

The team encounters a problem of how to get the name of the product from its Amazon asin ID. The team figured it out after some coding of the get_name() function. The team then starts cross development of web interface and data analysis. Figuring out how they communicate each other took a bit of work, and it turns out PHP + MySQL provides both web interface and API access for data processing because the network restriction on connecting to the MySQL server outside the campus network. Then the problem comes to the method of saving all the existing reviews to database for web interface accessing. The team figured out to host a local MySQL database and run SQL query through python to import all the products along with its names. In the method, the team saves a long time by exporting the table from local database and import though phpMyAdmin to MSU MySQL server. The team then develop a web API for local script to acquire user inputs of user reviews. After all, using SVD algorithm to make prediction and saving the recommendation back to the database is doable through another PHP API. In order to build a web-based interface, team used php to implement the website. To avoid unauthorized user, team decided not to implement user registration page but registered username and password manually on database, so user can log in only with username and password provided by team.

Users is a php class representing user table. User is a prototype of users in Users. To get information about user who logged in. Team saved user information in the session so when user reviews a product, we can store specific user information with what he rated together into database.

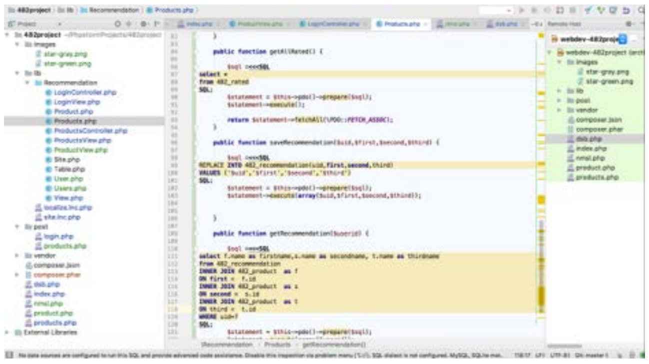
Products is a php class representing product table. Product is a prototype of elements in Products. Team implemented member functions in Products class with SQL query to get or update data in database. In order to display correct information of products and ratings, queries in functions need to cross tables to display specific names with given product id.

Team also implemented login controller and products controller which are determining if the login and rating is valid.

Team was unable to set up tables for products and reviews appropriately. Solved this problem by add CSS files referred from bootstrap.com.

View classes team implemented will generate HTML for pages, loops over database tables and if user has reviewed any products, the review and recommendation will be displayed.

Team implemented error checking for invalid login and rating, too. If password and username are not match, user will not get permission to access product page or any page of recommendation system. If user give a rating out of range 1-5, an error message will show up. The review will not be stored in database, and user gets redirected back to product selecting page to select product and review again.



```
    }

    public function getAllRated() {

        $sql =<<<SQL
select *
from 482_rated
SQL;
        $statement = $this->pdo()->prepare($sql);
        $statement->execute();

        return $statement->fetchAll(\PDO::FETCH_ASSOC);
    }

    public function saveRecommendation($uid,$first,$second,$third) {

        $sql =<<<SQL
REPLACE INTO 482_recommendation(uid,first,second,third)
VALUES ('$uid','$first','$second','$third')
SQL;
        $statement = $this->pdo()->prepare($sql);
        $statement->execute(array($uid,$first,$second,$third));


    }

    public function getRecommendation($userid) {

        $sql =<<<SQL
select f.name as firstname,s.name as secondname, t.name as thirdname
from 482_recommendation
INNER JOIN 482_product  as f
ON first =  f.id
INNER JOIN 482_product  as s
ON second =  s.id
INNER JOIN 482_product  as t
ON third =  t.id
WHERE uid=?
SQL;
        $statement = $this->pdo()->prepare($sql);
```

4. EXPERIMENTAL EVALUATION

4.1 Experimental Setup

Team collected 10261 reviews for 900 products from Amazon. There are 3 attributes for the dataset: user id, product id, and rating. Team tried to throw random reviews into the system. The more random reviews are, the result's RMSE becomes higher. Which means reviewing randomly will distract this recommendation system. Team used phpStorm to implement the website and Jupyter notebook the setup the algorithm.

Timeline:

March 18, 2018 Intermediate report.

March 20, 2018 Set up the git repository.

March 24, 2018 Set up python surprise.

March 30, 2018 Implementing database and web interface

April 4, 2018 Set up php API

April 10, 2018 Implementing algorithm

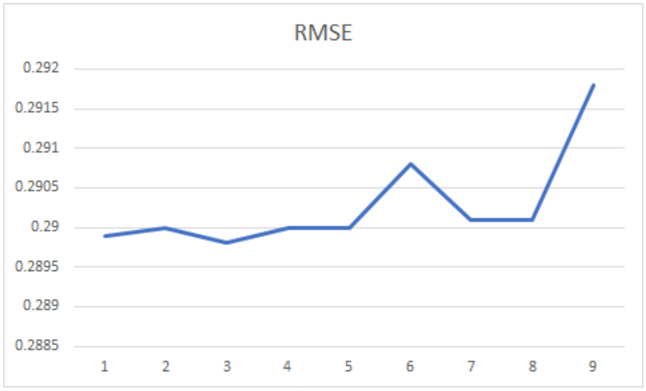
April 24, 2018 Finishing up project.

4.2 Experimental Results

Team designed a website(http://webdev.cse.msu.edu/~duzengze/recommendation/) to let user review products. After logging in, user can choose products to review by giving a 1-5 stars rating and a review for the product. After reviews are updated, system will generate 3 items that are recommended for user based on the ratings or review text.

4.3 Discussion

Team found out that when user keeps giving reviews, the recommend items will be different. After times of experiment, team figured that the more reviews user gives the more accurate the recommendation is.



What's more, other users ratings and reviews can affect results too.

5. CONCLUSIONS

For future work, team suggests that companies focus more on details like reviews rather than just rating score. Since although a score can immediately describe the rating of an item, for example:4.5 out of 5 average rating, but it is still very abstract for customers. It can only tell user whether other customers like this product or not, but not why is it good or the product has what kind of flaw.

For example, based on a customer's review, company gives a list of recommendations of products. However, the first rated item is a super good-rated item but has some kind of flaw that user cannot accept. The second rated item, on the other hand, has lower ratings than the first one, but this customer does not care about the flaw it has. This user likes the second-rated product more than the first but it does not mean that the first-rated item is not a good product. Users can only tell company how much they like the product by rating it, but in order to get their preferences, companies need to analysis their reviews.

6. REFERENCES

[1] bootstrap.com for stylesheet of web interface.

[2] Amazon product data: http://jmcauley.ucsd.edu/data/amazon/

[3] Surprise Toolkit: http://surpriselib.com

System instruction:

1. Web interface is accessible through http://webdev.cse.msu.edu/~duzengze/recommendation/.

   Username: admin

   Password: pass

2. Review the products by clicking on the product number and enter rating and review text for the products.
3. Run the refresh script “refresh.py” to re-calculate the recommendations
4. Refresh the webpage to view the new result