Assignment 4

Recommender System using Apache Mahout

November 12, 2014

1 Introduction

If you have used Amazon, Netflix, or Pandora, you must have used their "Recommender Systems". Several of the modern web services use some kind of recommendation engine to suggest their products. There are two types of Recommender Systems. When a user's information like ratings are used to predict the products he/she may like, it is called 'Collaborative Filtering'. Whereas when the product's properties are used to determine the items you may like, it is called as 'Content-based Filtering'. For this assignment we will be doing only Collaborative Filtering. Based on the products liked by group of users, you will recommend other products which the user may not have bought but are likely to buy.

2 Collaborative Filtering

Collaborative Filtering works by analyzing the common products liked by users and recommending the products based on likeness of other users with similar product ratings. This can either be calcuated with a 'Probabilistic Model', like Bayesian Networks or Non-probabilistic models like Nearest Neighbor. In this assignment we will be using only the non-probabilistic models. When the Collaborative Filtering is done based on the similarity of Users, it is called as User Based Recommender. When the Recommendation is done based on the similarity of the product (the 'item'), it is called as Item Based Recommender. Although it is easy to see that both are similar problems looked from different angles, there are critical differences which makes them different. First, items do not change, thus their similarity with another item remains constant. Second, in most cases 'items' do not increase unlike 'user'. This is helpful because now these values could be pre-computed and stored for faster online-recommendation. Because of these reasons, in Apache Mahout, while Item Based Collaborative Filtering is implemented with a distributed algorithm which can be run in MapReduce, User-Based Collaborative Filtering is not. Therefore, for this assignment we will be writing **Item-Based Collaborative Filtering.**

Non-probabilistic Collaborative Filtering can be essentially reduced into a Neighrest Neighbour function, which is computed using the similarity of the items. Consider the following equation, which gives Prediction Score for item i by an user u:

$$Predict(u,i) = \frac{\sum_{j \in items\_\_rated\_\_by\_\_u} similarity(i,j) \times r_{uj}}{\sum_{j \in items\_\_rated\_\_by\_\_u} similarity(i,j)}$$

where, $r_{uj}$is the rating of the item j by user u

and $similarity(i,j)$ is the function which gives the similarity measure between two items i and j. Where the *similarity* could be either any of the existing similarity measures implemented in Apache Mahout like Pearson, Spearman, Euclidean, Tanimoto and Loglikelihood or it could be user defined.

3 Dataset

You are provided with the same big Amazon Review file you used for the Assignment 0.

File "**all.txt**" is provied here

/home/o/class/cs129a/assignment4/all.txt

This path is accesible from any of the Public CS Machines

http://www.cs.brandeis.edu/~guru/public_work_stations.html

Extract the fields you think are relevant *(see Assignment 0 for the details of the fields)* for the assignment.

I would suggest the following but feel free to experiment.

product/productId, review/userId, review/score

You are free to apply any preprocessing that you think may benefit the test score. In some cases it has been found that removing the items with only one review might help get better scores.

Do **not** copy the "**all.txt**" to your home directory or on server. Please do the preprocessing by reading it from that source, as you did for the assignment 0.

While "**all.txt**" is a 34 GB file, the extracted and preprossed file is expected to be **around 1 GB**. If you have too large a file, you should check your preprocessing.

You may also have to convert the alphanumberic IDs into numeric, based on what mahout application or library you use. Simplest way would be to hash it or write your own mapping code. Feel free to do what seems simplest to you.

4 Implementation

Mahout or No Mahout

You are encouraged to use MAHOUT but it is not a mandatory requirement. Since the collaborative filtering algorithm is easy to understand, you may write your own code using MapReduce. If you can write Python code and make it run in Hadoop environment (Hadoop Streaming), you are free to do so.

If Mahout

Command Line or Code

If you decide to use Mahout, you will be benefitted by looking at "mahout recommenditembased" application, which runs from terminal. All the algorithms for the recommendation can be run from terminal. It is easy to run the terminal commands but they give you limited access to tweaking the parameters. You should start with these applications and if you have time, write your own code to improve your score.

If Code

It is important to understand the **DataModel** (FileDataModel), for the Recommender Systems. Kindly refer the Apache Mahout manual for the details. Please run your recommender with atleast two different similarity measures. Please note to keep the same seed [**RandomUtils.useTestSeed();**] with randomization so that you get the same results. It is because the traning/test split is randomized by Apache Mahout. For the list of given Users you will be asked to submit 10 product recommendations. You may optimize your code, Nearest Neighbour parameters etc to suit this application.

Memory Issues

This is a memory intensive operation and there is a chance that you will encounter memory issues in one form or another.

See the appendix below for some ideas to overcome these issues.

Best way to fix memory issues, is to reduce your data. Since the number of products for which you have to recommend is only 100 therefore try to reduce the size of your input data by removing the lines which you find are completely unrelated. How you do an interesting exercise and a thoughful process, and is part of learning of this assignment. However remove the review lines by caution, if you randomly remove lines, your recommendation will be subpar and thus may get low accuracy.

5 Submission

You have to provide the "Products most similar or Products most likely to be bought" for the given items.

Think of this way, when you see the Amazon website, without being logged in, and search for some products then what other products does Amazon show you as "You may also like" ? These are item based recommendation, since they do not have enough information yet to generate "User based recommendation".

Please provide **max upto 10** products for every given product id in the list. You have to submit your recommendations for two different similarity measures. Thus you will have two output file, one for each similarity measure.

In some cases you may have only few but that is okay, but do not submit more than 10 product recommendations.

Each group gets their own set of "items".

Please find the set of ProductIds to recommend on for your group at the following path

/home/o/class/cs129a/assignment4/items_GroupNum_<Group_NUMBER>.txt

For e.g if you are Group 14 then the file is present at

/home/o/class/cs129a/assignment4/items_GroupNum_14.txt

Kindly submit your output in the following format:

given_product_id,productId1,productId2,productId3—***NEWLINE***—

Please ensure that the sequence of given_product_id is as provided in the '**items_GroupNum_<Group_NU**
file. Please do not rearrange or randomize the list.

If you are confused see this.

Say your items file contains following two lines

B0000006YI

B000KXNYMS

Then your output file should look like this:-

B0000006YI,B0002E568K,B0002DVDQK,B001STPJJO,B001JTH9E6

B000KXNYMS,B0013OUGOC,B0002FOAT0,0553257994

where "B0000006YI" and "B000KXNYMS" are the 'given product id' whereas other values are the products you recommend.

Please submit a report and describe the techniques and thought process behind design choices. Also mention what similarity measure you used and give your explanation of any differences observed. Mention any issues you faced. Also please try to include any resources or articles which you benefitted from.

Your output folder should contain atleast the following items

Output_Similarity_measure_1.txt

Output_Similarity_measure_2.txt

Report.pdf (or .txt)

6 Extra Credit

6.1 Recommendation for items of all the groups - 10 Extra Points.

Provide the Product recommendation for every group files

Find all the group files at the following path

/home/o/class/cs129a/assignment4

They are called items_GroupNum_1.txt to items_GroupNum_19.txt.

Provide your output as separate file for each group. Number the ouput files as the group numbers.

The extra points you will earn will depend on the accuracy you get. If you get 0 accuracy but you did recommendation for all the groups, you get 0 extra points. If you get 50% accuracy for all the groups combined, you get 50% of the extra points.

6.2 User Based Recommendation - 20 Extra Points

Do the userbased recommendation recommendation. List of Users for which recommendations we want is provided here

/home/o/class/cs129a/assignment4/amazon_user.txt

*Warning: Do this only when you have completed rest of the assignment.

**Major warning: This is very slow and time consuming process. Try not to overload the cluster

6.3 Contextual Recommendation based on text - 20 Extra Points

Use the "text review" for each product, to increase the similarity of the product. You may consider using techniques of the last assignment (clustering of reviews) to augment the distance for the products. If you are CL student you are encouraged to do this part.

Please see this wonderful article on the techniques.

Easier way to do this would be to do recommendation sepearately, using the ratings provided and the textual information and then take the union of the set of the product recommended.

7 Useful Resources

Apart from class notes and wikipedia articles, you may find the following resources useful in understanding Item Based Collaborative Filtering.

Mahout tutorial - Contains basics of getting started, FAQ and exhaustive list of resources
https://mahout.apache.org/users/recommender/recommender-documentation.html
https://mahout.apache.org/users/recommender/intro-itembased-hadoop.html
http://www.slideshare.net/Cataldo/tutoria-mahout-recommendation
http://www.cs.carleton.edu/cs_comps/0607/recommend/recommender/itembased.html

8 Appendix

Issues with memory management

The more data you give, the better. Though Mahout is designed for performance, you will undoubtedly run into performance issues at some point. For best results, consider using the following command-line flags to your JVM:

- -server: Enables the server VM, which is generally appropriate for long-running, computation-intensive applications.

- -Xms1024m -Xmx1024m: Make the heap as big as possible – a gigabyte doesn't hurt when dealing with tens millions of preferences. Mahout recommenders will generally use as much memory as you give it for caching, which helps performance. Set the initial and max size to the same value to avoid wasting time growing the heap, and to avoid having the JVM run minor collections to avoid growing the heap, which will clear cached values.
- -da -dsa: Disable all assertions.
- -XX:NewRatio=9: Increase heap allocated to 'old' objects, which is most of them in this framework
- -XX:+UseParallelGC -XX:+UseParallelOldGC (multi-processor machines only): Use a GC algorithm designed to take advantage of multiple processors, and designed for throughput. This is a default in J2SE 5.0.
- -XX:-DisableExplicitGC: Disable calls to System.gc(). These calls can only hurt in the presence of modern GC algorithms; they may force Mahout to remove cached data needlessly. This flag isn't needed if you're sure your code and third-party code you use doesn't call this method.

Also consider the following tips:

- Use CachingRecommender on top of your custom Recommender implementation.
- When using JDBCDataModel, make sure you wrap it with the ReloadFromJDBCDataModel to load data into memory!.

These tips were obtained from https://mahout.apache.org/users/recommender/recommender-documentation.html

All the best