Brandeis University
Department of Computer Science
COSI 129a - Introduction to Big Data Analysis
Fall 2016

Assignment 4: Recommender System using Apache Spark

1 Introduction

If you have used Amazon, Netflix or Pandora, you have been targeted by 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 Content-based Filtering. For this assignment we will be doing only Collaborative Filtering. Based on the products liked by groups of users, you will recommend other products which the user may not have bought but are likely to buy. This is based on the assumption that other people who may have purchased similar product and given similar rating like you, may have similar interests or similar lifestyle.

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 calculated with a probabilistic model like a Bayesian Network or a non-probabilistic model like Nearest Neighbor. In this assignment we will be using only the non-probabilistic models. When Collaborative Filtering is done based on the similarity of Users, then the system is called a User Based Recommender. When it is done based on the similarity of the products (the items), then the system is called an Item Based Recommender. Although it is easy to see that both are similar problems seen from different angles, there are critical differences. First, items do not change, thus their similarity with other items remains constant. Second, unlike with users, in most cases the amount of information associated with items does not increase. This is helpful because now these values could be pre-computed and stored for faster online-recommendation. For this assignment we will be only using Item-Based Collaborative Filtering.

There are numerous algorithms for Item-Based Collaborative Filtering, but in this assignment we will work with an algorithm called Matrix Factorization. It is basically to find two matrices U (user matrix with $size = \#Users \times r$) and P (product matrix with $size = \#Products \times r$) so that $U \times P^T \approx A$ where A is the matrix of which each cell is the rating given by an user to a product (cell without rating is not approximated). It is tantamount to an embedding feature space for both users and items. Rating of a user u given a new product p, therefore, is calculated by $U_u * P_p$, and similarity between two products p_1 and p_2 is given by $P_{p1} * P_{p2}$. You can find reference to this algorithm at the end of the assignment.

The purpose of this assignment is to get you familiar with Spark and run some simple algorithms on it. It would also provide you a sense of the advantage or disadvantage of Spark in comparison to other frameworks such as Hadoop. Even though you can use either Java, Scala or Python, in this assignment we encourage you to try Scala in the Spark shell for your processing.

2 Data

You are given the same big Amazon Review file you used for Assignment 0 as well as a file that contains a list of products (items). These files are put on HDFS on both clusters:

```
/shared3/data.txt
/shared3/data-medium.txt
/shared3/data-small.txt
/shared3/items.txt
```

The file *data.txt* has the Amazon reviews, see assignment 0 for more details on this file. Do not download *data.txt* to your home directory, please do the processing by reading the file from its source on the cluster. *data-medium.txt* and *data-small.txt* are two smaller versions that you can download and test your code before moving to the bigger file. The file *items.txt* has a list of product identifiers from the reviews file.

3 The Problem

The goal is to provide the products most similar to the given items. Think of it 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 recommendations and they are given since Amazon does not have enough information yet to generate user based recommendations.

From the reviews, extract the fields you think are relevant. 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. While *data.txt* is a 34 GB file, the extracted and preprocessed file is expected to be around 1 GB and if you are the only user on the cluster it should take 15-30 minutes. If you have too large a file, you should check your preprocessing.

The input products to your recommender are in *items.txt*. Your output should be in the format `given_product,recommended_product1,recommended_product2,...`, where `recommended_productN` are the products you recommend given `given_product`. All products are represented by their identifiers. Do not recommend more than 10 products. Please ensure that the sequence of given_product is as provided in the items file and do not rearrange or randomize the list. Below is some example input and output where "068413263X" and "B0000A0QE6" are the given product identifiers and the other values are the identifiers of products you recommend.

```
Input (from items.txt):
   068413263X
   B0000A0QE6
Output:
   068413263X,B0002E568K,B0002DVDQK,B001STPJJO,B001JTH9E6
   B0000A0QE6,B0013OUGOC,B0002FOAT0,0553257994
```

This is a memory intensive operation and there is a good chance that you will encounter memory issues in one form or another. The best way to fix memory issues is to reduce the size of 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 that is part of this assignment. However, use caution when removing lines from reviews, if you randomly remove lines, your recommendation will be subpar and thus may get low accuracy.

We suggest you run your code on `data-small.txt` first. This file is 100 times smaller than `data.txt` and you should have no memory issues. Then ramp up to `data-medium.txt`, which is 10 times smaller than `data.txt`. There too, you should have no memory issues if you have the same solution as we have. Finally, run your code on `data.txt` and try to solve any memory issues that you will most likely get.

4 Submission

You need to submit, as a group, the following:

1. Your source code.
2. Your recommendation for the products in the items list, up to 10 recommendations per product, in the format specified above. Submit the recommendations from the largest data set that you were able to run your code on.
3. A report, about 3-4 pages. Give specific and complete pointers on how to run your code. Describe the techniques used and the thought process behind your design choices. Mention any issues you faced, especially on memory issues that you may have encountered. Include any resources or articles that you benefited from.

5 Grading

As usual, your grade will depend on:

- The quality of your code. Does it run? Is it well-structured? Is it easy to understand? (50 points)
- The quality of your report. (50 points)

We expect that you will be able to deal with the `data-medium.txt` file. If you manage to successfully run your code on `data.txt` you will get extra credit (20 points).

6 Important Notes

- For this assignment, the Hadoop installation has been updated to **Hadoop 2.7.3** and the Spark version is **spark-2.0.2-bin-hadoop2.7**. Therefore you need to reset your HADOOP_HOME and SPARK_HOME accordingly.
- You should start your Spark shell by running the following command. An alternative is to write your code in a .scala file and use spark-submit.

```
spark-shell --master yarn --deploy-mode client --queue YOUR_QUEUE
--driver-memory 4g --executor-memory 4g --executor-cores 2
```

- By running your Spark in Yarn, it would create a Yarn Application to host your Spark driver. You can then kill your Spark application using your ApplicationId (which you would see at the beginning of your shell)

```
yarn application -kill ApplicationId
```

- If you get problems related to GCOverhead Memory, check the relevant parameter in Spark Properties in http://spark.apache.org/docs/latest/running-on-yarn.html.

7 Resources

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

1. Paper about Recommendation using the Matrix factorization method.
2. Source code of Matrix factorization model class. You could base your vector multiplication on this.
3. Resource to run spark on Yarn.
4. A tutorial on using Spark.
5. The Spark Mllib example.