CS 6675 Project Proposal (Group 17) Developing Fast, Scalable and Useful Recommendation System on Large Datasets

Parminder Bhatia (Leader) **Revant Kumar** **Gopi Krishnan Nambiar**

1 Motivation and Objectives

Recommendations systems have become an integral cog in any business. Suggesting any product to the user that he/she might be interested in among the entire product catalog, helps the user in saving time as well as makes the decision process easy. It also attracts the user in purchasing more, thus increasing the overall revenue for the company.

There has been a lot of research in building good recommender systems for the users. However, one of the biggest challenges is that of scalability, i.e. when the number of users and products are in the scale of millions. The objective of our project is to develop a scalable recommendation system, particularly Collaborative Filtering using the Map Reduce Paradigm. We also feel that social intuitions are really important for a good Collaborative Filtering algorithm, and thus we plan to also add social elements like location, number of friends etc. Adding Social elements can also help us in reducing the cold start problem, as we can get basic profile information about the person from demographics and other data from relevant social networks.

We plan to do optimized Matrix Factorization as a Map Reduce Task. We would also have to investigate as to how we can incorporate social elements so as provide decent results to new users.

2 Related Work

In [1], the authors have introduced a new idea to formulate topic based social influence analysis using Graph - (V,E) where V represented users or entities and E, for undirected edges. The authors attempt to design a model which takes into consideration both topic distribution and network information to get a sense of the social influence, which is very different from the existing works on social network analysis conducted till that point of time. These two items are used as inputs for the program. In order to make the system conform to Map-Reduce framework, the authors implement a modified message passing algorithm which is compatible for the same. First, a Topic Factor Graph(TFG) is constructed which is a probabilistic model that includes all the information in a combined model. This is followed by the construction of the TAP model and parallelization procedure for the Map Reduce implementation. The TFG model consists of observed variables and latent variables as well which are the nodes in the network. They have run the algorithm in a distributed manner which contributes to a speedup of about 15X for large datasets. As expected with parallel processing, this method is not efficient for smaller datasets.

[7] The cold-start recommendation task in an online retail setting for users who have not yet purchased (or interacted in a meaningful way with) any available items but who have granted access to limited side information, such as basic demographic data (gender, age, location) or social network information (Facebook friends or page likes). In this paper[7] , they formalize neighborhood-based methods for cold-start collaborative filtering in a generalized matrix algebra framework that does

not require purchase data for target users when their side information is available. In real-data experiments with 30,000 users who purchased 80,000+ books and had 9,000,000+ Facebook friends and 6,000,000+ page likes, they show that using Facebook page likes for cold-start recommendation yields up to a 3-fold improvement in mean average precision (MAP) and up to 6-fold improvements in Precision and Recall compared to most-popular-item, demographic, and Facebook friend cold-start recommenders. These results demonstrate the substantial predictive power of social network content, and its significant utility in a challenging problem like recommendation for cold-start users.

3 Proposed Method

In order to build our recommendation system we will be using a class of algorithms called called collaborative filtering, specifically Item-Based Collaborative Filtering. The basic idea of collaborative filtering is very simple and intuitive. If a user has shown interest in item A and a second user has shown interest in A and B, then it's very likely that the first user may also be interested in item B.

Now, we will explain the main parts of the project one by one:

3.1 Data Collection

In addition to item user data, we may have the user fill up a questionnaire, where he can select his interests, similar to what Quora, Pinterest do with the new user. We can also look into already build datasets such as MovieLens.

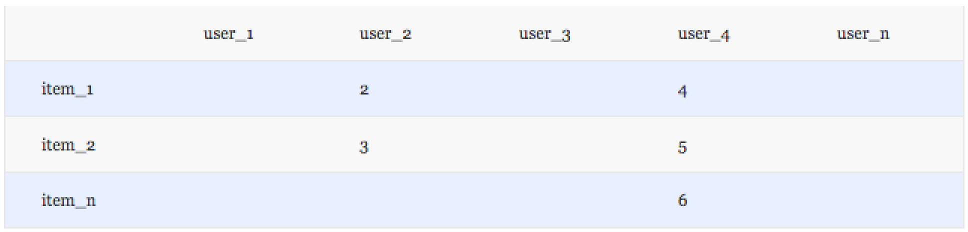
3.2 User Engagement

Level of user engagement can be estimated based on the following user actions. Typically, explicit rating by user is not available and other engagement events as below are used to estimate user's interest or engagement level.

- Time spent on a product page
- Item added to wish list
- Item added to shopping cart
- Item purchased

3.3 Utility Matrix

All the social algorithms take an utility matrix as input. It contains data on user engagement for different items. In our case, If there are m items and n users, an utility matrix is a $m \times n$ matrix. The value of a matrix element (i, j) represents the rating of an item i by an user j. A row represents an item and it's associated ratings. A column represents an user and all the ratings for that user.

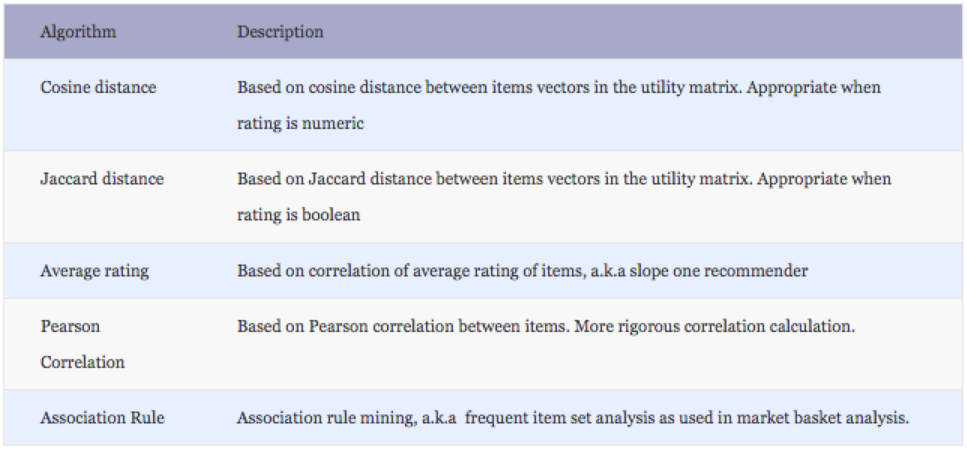


	user_1	user_2	user_3	user_4	user_n
item_1		2		4	
item_2		3		5	
item_n				6	

The utility matrix is very sparse, because typically a user will engage with a small fraction of the items available in the inventory. There is a duality about the utility matrix. The utility matrix can be used either to find similar items or similar users. We will be using item based similarity which has been found to produce better results.

3.4 Algorithms for distance metric

There are various social algorithms for distance metrics that can be used. We have listed some of them in the below table.



Algorithm	Description
Cosine distance	Based on cosine distance between items vectors in the utility matrix. Appropriate when rating is numeric
Jaccard distance	Based on Jaccard distance between items vectors in the utility matrix. Appropriate when rating is boolean
Average rating	Based on correlation of average rating of items, a.k.a slope one recommender
Pearson Correlation	Based on Pearson correlation between items. More rigorous correlation calculation.
Association Rule	Association rule mining, a.k.a frequent item set analysis as used in market basket analysis.

3.5 Vector Distance

Each row of the utility matrix represents an item. The values of the row vector are the ratings by different user. Essentially we find the cosine distance or jaccard between each pair of item vectors. Since these vector are sparse, they are effective means of finding similarities between the vectors.

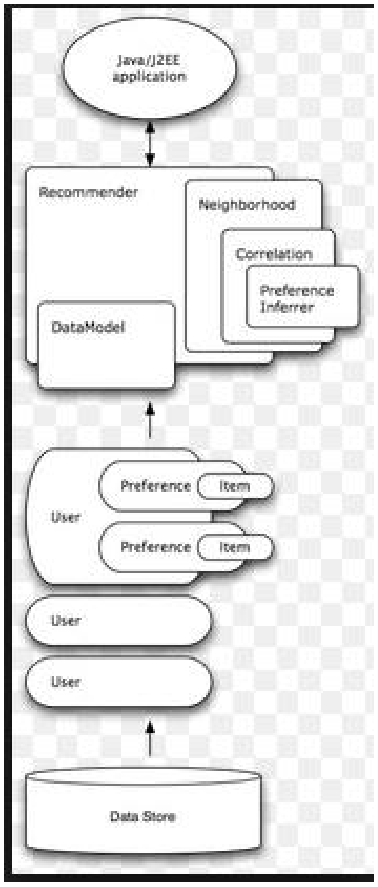
3.6 MapReduce/Hadoop/MRJOB

MapReduce is a framework originally developed at Google that allows easy large scale distributed computing across a number of domains. Apache Hadoop is an open source implementation of it. It scales well to many thousands of nodes and can handle petabytes of data. For recommendations where we have to find the similar products to a product you are interested at , we must calculate how similar pairs of items are. For instance, if someone watches the movie Matrix, the recommender would suggest the film Blade Runner. So we need to compute the similarity between two movies. One way is to find correlation between pairs of items. But if you own a shopping site, which has 500,00 products, potentially we would have to compute over 250 billion computations. Besides the computation, the correlation data will be sparse, because it's unlikely that every pair of items will have some user interested in them. So we have a large and sparse dataset. And we have to also deal with temporal aspect since the user interest in products changes with time, so we need the correlation calculation done periodically so that the results are up to date. For these reason the best way to handle with this problem is going after a divide and conquer pattern, and MapReduce is a powerful framework and can be used to implement data mining algorithms.

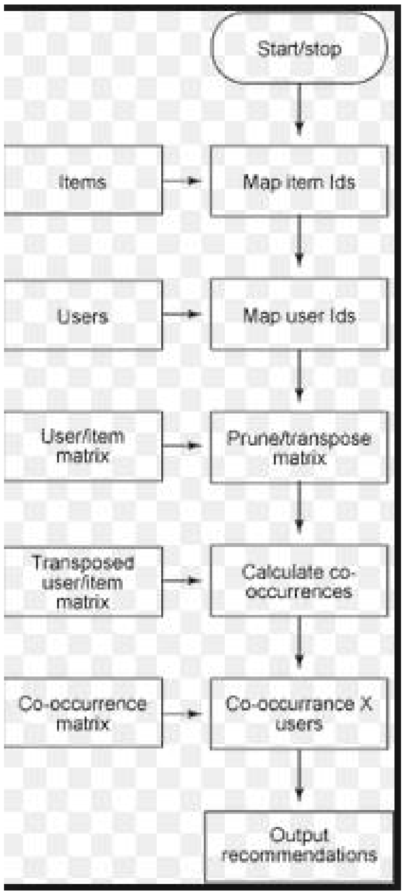
MRJob is a Python package that helps you write and run Hadoop Streaming jobs. It supports Amazon's Elastic MapReduce(EMR) and it also works with your own Hadoop cluster. It has been released as an open-source framework by Yelp and we will use it as interface for Hadoop since it is easy to handle with MapReduce tasks.

3.7 User Rating Support

It may be desirable to exclude users who are not sufficiently engaged i.e the number of items that an user has been engaged with is below some percentage of the total number of items. This threshold is know as support in the data mining world.



(a) Figure 1



(b) Figure 2

Figure 1: Flow Diagrams

3.8 Social Network Angle

We know that humans are multifaceted. Only some of these facets are manifested in a social network. However, those facets may be relevant or irrelevant in the context of recommendation of some products and services. Thus, we will try to study whether an user's social network can be overlaid on the utility matrix to create a personalized utility matrix, so that it could be used for recommendations. Essentially, user's rating in the utility matrix could be weighted down by some radial function based on the degree separation in the social graph.

3.9 Flow Diagrams

Flow diagrams can be seen in the Figures 1 and 2 above.

4 Plan of Action

February:

- Week 2
  - Preliminary preparation and setup
    - Cluster setup
    - Software stack familiarization

        - Collection and decisions regarding the dataset, size and formatting
    - Testing and experimentation with algorithms on a local machine setup
- Week 3
    - Data set finalization : Decide on test sets, training sets and on the final data source
- Week 4
    - Perform experiments with CF on different parts of the dataset
        - Experiment with the available algorithms
        - Run the tests with the cleaned dataset on a Hadoop cluster
    - Data cleaning

March:

- Week 1
    - Algorithm improvements and tweaks
        - Try to achieve improvements in the algorithms that already exist
- Week 2, 3
    - Application of algorithms
        - Test with a suite of available tools and benchmark datasets to obtain efficient results for improvement of the recommender system
        - These may be either applied from an open source library or a combination of libraries and manual techniques which are devised after significant experimentation with a subset of the entire data available
- Week 4
    - Performance improvements
        - Test and try to achieve performance improvements compared to the already existing algorithms.

April:

- Week 1,2
    - Testing algorithms on the large dataset
        - Polishing and fine tuning the algorithms for desired results
        - Evaluation of the experiment results with current algorithms and techniques
- Week 3,4
    - Report
        - A detailed report on the findings and improvements of this work over the current ones.

5 Evaluation and Testing Method

We will review and evaluate our recommender system using three types of experiments as follows:

1. Starting with an offline setting, where recommendation approaches are compared without user interaction
2. Then reviewing user studies, where a small group of subjects experiment with the system and report on the experience
3. Finally describe large scale online experiments, where real user populations interact with the system.

In each of these cases we plan to describe types of questions that can be answered, and suggest protocols for experimentation. We will also discuss how to draw trustworthy conclusions from the conducted experiments. We then review a large set of properties, and explain how to evaluate systems given relevant properties. We will also survey a large set of evaluation metrics in the context of the property that they evaluate.

Some Evaluation Metrics that we plan to report fall in two categories:

- Statistical Accuracy Metrics:
  - Root Mean Square Error (RMSE)
  - Mean Absolute Error (MAE)
  - Correlation
- Decision Support Accuracy Metrics:
  - Reversal Rates
  - Weighted Errors
  - ROC Sensitivity

6 Bibliography

1. J. Tang, J. Sun, C. Wang, and Z. Yang. Social influence analysis in large-scale networks. In KDD, 2009.
2. User-based Collaborative-Filtering Recommendation Algorithms on Hadoop
3. http://aimotion.blogspot.com/2012/08/introduction-to-recommendations-with.html
4. https://pkghosh.wordpress.com/2012/04/21/socially-accepted-recommendation/
5. Guy Shani and Asela Gunawardana. Evaluating Recommendation System
6. Jonathan Herlocker, Joseph Konstan, Loren Terveen, and John Riedl. Evaluating Collaborative Filtering Recommender Systems. ACM Transactions on Information Systems, Vol. 22, No. 1, January 2004, Pages 5-53
7. Social collaborative filtering for cold-start recommendations. S Sedhain, S Sanner, D Braziunas, L Xie - Proceedings of the 8th, 2014 - dl.acm.org
8. Feng, Qinyuan, et al. ”Enhancing personalized ranking quality through multidimensional modeling of inter-item competition.” Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2010 6th International Conference on. IEEE, 2010.
9. Herlocker, Jonathan L., et al. ”Evaluating collaborative filtering recommender systems.” ACM Transactions on Information Systems (TOIS) 22.1 (2004): 5-53.