Brunda Chouthoy
CSC 577 – Final Report
Evaluation of Recommender Systems using Book Crossings data

Description of the data

This project uses a subset of reviews for a book ratings dataset that contains 1.1 million ratings of 270,000 books by 90,000 users. The ratings are on a scale from 1 to 10, and implicit ratings are also included.

Data Source - http://www2.informatik.uni-freiburg.de/~cziegler/BX/

The Book-Crossing dataset comprises 3 tables:

- BX-Users - User IDs (`User-ID`) have been anonymized and mapped to integers. Demographic data is provided (`Location`, `Age`) if available.
- BX-Books - Books are identified by their respective ISBN. Some content-based information is given (`*Book-Title*`, `*Book-Author*`, `*Year-Of-Publication*`, `*Publisher*`)
- BX-Book-Ratings - Contains the book rating information. Ratings (`*Book-Rating*`) are either explicit, expressed on a scale from 1-10 (higher values denoting higher appreciation), or implicit, expressed by 0.

Data Cleaning and Transformation:

I have used R Studio to clean the data and transform to the format accepted by Librec.

'BX-Book-Ratings.csv' contained 1149780 ratings in the format – User, Item, Rating. A subset of data was generated - The ratings data was filtered and a subset was created to only retain the explicit ratings (1-10) and converted to a text format that is accepted by Librec. The size of the table is reduced to a total of **433,671** instances.

In the data preprocessing phase, there are two input components to be generated:

- the (user, Item, rating) file for the TrainMatrix in Librec
- the (item, feature) file for the content based Naïve Bayes and Hybrid recommenders

To generate the content features for the books, 3 of the book features available were considered: Book-Author, Year-of-Publication and Publisher. Feature look-up files containing FeatureID, Feature_name is generated by using the three item features considered for evaluation.

- Feature_author.txt containing 102040 features for the 'Book-Author' variable
- Feature_year.txt containing 117 features from the 'Year-of-Publication' variable
- Feature_publisher.txt containing 16822 features from the 'Publication' variable.

Each of the feature look-up files were used as input to create the ItemFeatures file containing ISBN (item id) and a feature list (feature ids) for the corresponding item.

In the end of data transformation phase, two input files are generated that are in the Librec acceptable format- **BookRatings_Explicit.txt, ItemFeatures.csv**

Approach

Algorithms to be used for evaluation:

The important part was to figure out and choose carefully which recommendation algorithm to use. Each algorithm has its own characteristics and properties that can interact in harder ways with the given data set. So, it was essential to use as many as possible variations of collaborative, content and hybrid algorithms to evaluate which one is faster or more accurate to the data set to work with. Here is the list of algorithms:

Collaborative Filtering Recommenders:

- User based k-NN
- BiasedMFRecommender
- SVDPlusPlusRecommender

Content based Recommender:

- Naïve Bayes Recommender

Hybrid Recommenders:

- Weighted rating from combining 2 algorithms – SVD++ and user k-NN
- Weighted rating from combining content based and collaborative recommender- SVD++ and Naïve Bayes Recommender

Evaluation metrics:

I have used standard, quantifiable ways to evaluate the utility of recommendations produced by a recommender system – Mean Absolute Error (**MAE) and Precision**. MAE is an evaluator of predictions (continuous) while precision evaluates classifications (categorical), just as a point of distinction. Mean absolute error in the context of recommender systems means predicting how a user will review a product, and evaluating how far off our prediction was. Doing these many times and getting an average is the 'mean' in mean absolute error. 'Precision' is the proportion of top results that are relevant, considering some definition of relevant for your problem domain. For instance, 'Precision@10' would be this proportion judged from the top 10 results.

Evaluation methodology:

As a part of this evaluation project, I decided to test the built-in LibRec baseline algorithms using the evaluation metrics, including GlobalAverageRecommender, MostPopularRecommender, ItemAverageRecommender and UserAverageRecommender.

Each model is trained using **5-fold cross validation** and the split ratio is set to be at 0.8 for the training set. Here is a summary of the performance of each algorithm:

Baseline Recommenders	MAE	Precision@10
Global Average Recommender	1.496304034	0.0000969
Most Popular Recommender	4.865610844	0.0026678
User Average Recommender	1.283089842	0.000102
Item Average Recommender	1.532952705	0.0000337

It indicates that UserAverageRecommender's performance should be used as the benchmark MAE score to evaluate the performance of other recommender algorithms in LibRec.

Results

Collaborative filtering algorithms:

User KNN Recommender:

To tune the model, I adjusted the K parameter (number of neighbors) but there is no significant change in the evaluation results.

User k-NN	MAE	Precision@10
K=3	1.500857302	0.000893
K=50	1.495245138	0.001554059

Table 1: User KNN results

As per the results in Table 1, it indicates that MAE is not significantly different with the 2 different K values and the Precision value for K=50 neighbors is much better.

Biased MF Recommender:

This matrix factorization algorithm is trained using gradient descent. To tune the model, I adjusted three parameters – number of iterations, Regularization bias parameter and the learning rate for the Gradient Descent. Table-2 shows the tuning experiment results, with manual grid search, from this model.

Iteration	Reg Bias	GD Learn Rate	MAE	Precision@10
100	0.000001	0.001	1.262352106	0.0003816
200	0.000001	0.001	1.272578421	
500	0.000001	0.001	1.326807389	
1000	0.000001	0.001	1.354108982	
100	0.000001	0.01	1.354315027	
10	0.000001	0.01	1.262293153	0.0004281
10	0.0001	0.01	1.262299656	0.0005273
10	0.000001	0.001	1.344506732	0.0013228

Table 2: Biased MF Factorization results

As shown above, by increasing learning iteration does not help the model to learn because the learning rate was initiated to be too high. Lower learning rate and bias regularization term with lower learning iterations were then attempted, and a much better MAE result was returned.

SVDPlusPlus Recommender:

Table 3 shows the results for SVDPlusPlusRecommender.

Regularization bias	MAE	Precision@10
0.015	1.344398467	0.000381653
0.001	1.281601031	0.000662027

Table 3: SVDPlusPlusRecommender results

Content based Recommender – Naïve Bayes

The content based algorithm evaluation was done with a subset of data consisting of the first 50k ratings from the Book Crossings data: Train set-39999 and Test set: 9999 (using 0.8 as the ratio).

Threshold	MAE	Precision@10
5	5.220249245	0.003192747
7	3.801156238	0.002021586

Table 4: Naive Bayes Results

Table 4 shows the results for the Naïve Bayes recommender – I have used 2 different threshold values for the evaluation.

Hybrid Recommenders – For this project, I have used weighted hybridization that combines the results of two recommenders to generate a recommendation list or prediction by integrating the scores from each of the techniques in use by a linear formula.

(1) - Weighted rating from combining 2 algorithms – SVD++ and user k-NN:

Content weight	MAE	Precision@10
0.5	1.399475675	0.00085836

(2) - Weighted rating from combining content based and collaborative recommender- SVD++ and Naïve Bayes Recommender

Content weight	MAE	Precision@10
0.5	1.399475675	0.002839221
0.25	1.995576187	0.002641135

Results analysis:

Based on the results, the BiasedMFRecommender performs best compared to the baseline algorithm – User Average Recommender. However, I cannot consider the content based and Hybrid recommenders for comparison as the evaluation was done on a subset of data.

Given the sparsity of Book Ratings data within the dataset, mean absolute error was within a tolerable range for the most part (about 1.5-star off the user's actual review), but when evaluated by Precision the recommender systems were unimpressive. Whether the data even provides the opportunity for Precision to be high (due to data sparsity), is an open question, but at the same time an "eyeball test" of how well the recommender systems are performing will not work at scale.

Challenges faced:

Some issues that were encountered in evaluating the recommender algorithms in Librec were runtime, memory space and uncertainty on the best ways to leverage the metadata content provided with the data. Runtime was especially an issue with using the Precision ranking metric to evaluate algorithms. Most of the algorithms took 60+ minutes to complete when 5-fold cross validation was used. It became increasingly to try different tuning parameters to evaluate the model. Runtime and memory space was also an issue with the content-based Naïve Bayes Recommender and Hybrid methods. Since the total number of ratings for was huge – it resulted in the error 'java.lang.OutOfMemoryError: Java heap space'. I updated the VM arguments with different configuration settings for the –Xms<size> and –Xmx<size> (64, 512, 1024,2048 and 4000M), but it still resulted in the same error. Hence, with the given time frame

and resources , I decided to use a subset of the book ratings data for evaluation. Still, the runtime for Hybrid recommenders were very slow.

Conclusion

Through this project, I have explored various traditional learning algorithms like Singular Vector Decomposition etc. used in generating recommendation models and evaluation metrics used in measuring the quality and performance of recommendation algorithms. This knowledge gained will empower me and serve as a road map to improve the state of the art recommendation techniques.

If not for the limited timeframe, I would have liked to consider the entire dataset (including implicit ratings) to test and tune many of the collaborative filtering, content based and knowledge based recommenders using different evaluation metrics available to discover and evaluate results. In the future, it would be interesting not only to leverage more of the metadata to make better content-based recommendations, but also to try an entirely different method like knowledge based recommenders and other Hybrid algorithms. For example, I could not explore ensemble approaches in this project and it would be worthwhile to see if that method brings any improvements to speed or accuracy/relevance of recommendations.

I conclude that evaluations are important in the recommendation engine building process, which can be used to empirically discover improvements to a recommendation algorithm.