Multi-objective Recommender Systems Exploiting User Context

Aishwarya P (Roll No CS11B004)

***Abstract*—The main goal of this project is to study how context information can be used by a recommender system in order to maximize multiple objectives. This was studied for a specific case - maximizing accuracy and diversity of recommended movies, making use of user demographic data and movie genre information. The information obtained from a classifer that predicts aggregate statictics over a genre, based on user demographic data is used to increase the value of one objective - diversity, without a significant loss in the primary goal accuracy. The techniques used can also be generalized to other objectives.**

I. Introduction

TWO of the dominant research problems in recommender systems today are to optimize them for multiple objectives, most commonly accuracy, novely and diversity and to incorporate context information available to improve the quality of the recommendations. This project is aimed to incorporate context-awareness into multi-ojective recommender systems by combining approaches that solve the problems independently.

Traditional recommender systems deal with applications having two fuundamental types of entities - users and items. In the simplest case, the rating by a user on an item is predicted solely based on the ratings of the user on similar items and the ratings of similar users on the item in question. Context provides some other information about a user or the circumstances under which the recommendation is requested to improve the quality of the recommendation. More formally, in the data mining community, context is sometimes defined as those events which characterize the life stages of a customer and that can determine a change in his/her preferences, status, and value for a company [1]. For example, a travel recommendation system would provide differet suggestions based on whether it is summer or winter. The genres of music recommended to a college student and those recommended to a retired senior citizen are likely to be very different.

It can be argued that a considerable amount of context information is captured by making use of the ratings of users who have given similar ratings to the user in question in the past. While this may be true to some extent, it cannot be applied to say, new users for whom it is not yet established which users are similar to them. The common approach to deal with this case is to recommend the most popular items available to such users. But a recommendation that is most popular to a subset of users that appear to share similar contextual information is likely to be more useful. Thus, we would now find ”similar users” based on a combination of contextual information and existing ratings.

The information obtained from context would be more powerful in assisting a multi-objective recommender system. This is because, such a system would need to strike a balance between different objectives and the notion of balance can be formalized only by assigning some implicit or explicit weights to the different objectives. Contextual information could be used to design such weighting schemes to design a weighting that is more personalized to a given user at any given time.

Many methods have been proposed to design a multi-objective recommender system. In [2], the multi objective system has been modelled as a linear programming problem, where the objective function is a weighted sum of individual objectives and constraints are used to meet system requirements. In [3], the variation of likelihood of a customer buying a product based on the product price and relevance to the customer is captured using a probability distribution and an attempt is made to maximize the expected profit. In [4], the main idea is to define a dominance relation among objective functions and obtain the pareto-frontier - the set of possible solutions that have the property that the value of one objective function cannot be improved, except at the cost of another. This reduced serach space is then scanned for an optimal solution.

The technique used in this project borrows ideas from some of the above methods. It borrows the idea of maximizing the expected value of an objective function, whose value or any change in the value can be modelled using a probability distribution. Also, this method iteratively improves the current solution, improving an objective without affecting the other. But unlike in [4], a more coarse-grained search is used as the information inferred from the context is approximate. The focus of the project is to develop a scheme for increasing the diversity of recommendations without affecting their accuracy. The technique has been tailored to suit movie recommendations and has been experimented with on the MovieLens-100k dataset. However, possible suggestions have been included about how to extend the idea towards other domains and for other objectives.

II. Objective Definitions

For this project, two objective functions were considered. The primary objective is the accuracy, which is evaluated as the total rating of all items i recommended to all users u in their topN lists.

$$\text{Total Rating} = \sum_{u} \sum_{i \in topNList(u)} PredictedRating(u, i)$$

The secondary objective which is to be maximized without too much degradation of the primary objective is diversity. The notion of diversity addressed here is an explicit diversity defined using the nature of items, as in [4], rather than perceived diversity such as that measured in [5]. Here, the system is said to be more diverse if the spread of movies recommended from different genres is higher, that is, it is not sufficient to recommend a variety of movies but also to recommend a variety of movies from different genres. Intuitively, this appears to be useful to domains like movie, book and music recommendations to enable the system to cater to a variety of tastes and also because it is a reasonable assumption that this form of diversity is more perceptible to a user and also more likely to result in serendipitous recommendations.

Hence, the diversity measure used is as follows. For every genre g, let Count(g) be the number of movies recommended so far (in a practical setup, this count would be maintained over a time window) that are in genre g. Let

$$TotalCount = \sum_{g} Count(g)$$

Then the diversity of the system is calculated as the entropy of the count array, which is,

$$Diversity = -\sum_{g} \frac{Count(g)}{TotalCount} \log_2 \frac{Count(g)}{TotalCount}$$

III. The Recommendation System

Intuitively, the method used is as follows. Top-N recommendations for users are obtained using a base recommender and these lists are tweaked to incorporate movies from different genres in a manner that does not affect the overall rating too much. The constraint is a weak constraint, that is enforced as far as possible using a classifier that determines whether a user typically rates movies from that genre high or low. If a top-N list is being modified, only movies from genres that a the user typically rates highly are introduced.

Thus, the system proposed makes use of 2 primitives -

- A base recommender that is used to predict ratings for each user for each item, and produce a base top-N list to be improved upon.
- A classifier that uses user-demographic data to predict whether the user typically rates movies from that genre highly

IV. Justification of some design choices

The basic design choice in the experiment is the interpretation of the objectives. Since this system does not change the predicted rating of a user for an item, it is not necessary to measure the quality of an individual prediction. However, as the top-N list of the base recommender is not directly being used, some accuracy measure of a list is needed. The total rating is an intuitive choice for this.

Diversity however, can be interpreted in many ways. Traditional recommender systems simply aim to include only those movies that are expected to fetch a very high rating from a user, in their list of top-N recommendatiosn for that user. Due to this, it is possible that the same movies, or at least, only some ”general favourite” genres, get recommended to many people. Introducing diversity into the recommendation can refer to one of the following -

- Increase the number of movies that get placed in the top-N of at least one user
- Increase the number of genres that get placed in the top-N of at least one user (or make each genre appear at least a certain number of times)
- Increase the number of genres that get placed in the top-N of every user

The first 2 interpretations look at global diversity, that is, the diversity from the system's point of view whereas the last one looks at diversity from the user's perspective - how many different genres does the system expose a user to.

This influenced the decision of using aggregate ratings of users for genres. The average rating and fraction of ratings greater than or equal to 4 by the user for the genre are used to indicate whether the user typically rates movies from that genre high or low. This information is used when it is to be decided whether to replace a movie in the top-N list of a user. The intuition is that predicting aggregate information about genres will allow tweaking of recommendations for individual users to improve some global objective. For example, if the recommender system was rarely recommending romance movies and we wanted to recommend them, it makes more sense to recommend such a movie to a user who has a high average rating for romance movies or who has mostly rated romance movies with 4 or 5 out of 5 (again, not exactly equivalent but related criteria) rather than to a user who has a low average rating for romance movies.

The use of contextual information rather than ordinary collaborative filtering to obtain a movie from a genre to recommend was motivated by the following reasons -

- If a user has seen only a few movies from a genre, a movie from that genre will not usually be recommended to the user, even if the user has rated those few movies highly. This system reduces the chances of such skew because many demographically similar users may have rated movies from that genre highly.
- This could be a potential mechanism to recommend items

to new users. If the classification system is observed to work well, recommending popular movies from genres the user is predicted to like may give better results than recommending the most popular movies.

The other design choice made is the use of a classifier to predict whether a user's rating is likely to be high or low for an average movie in the genre. This is done, even though the aggregates are computed and stored for all users (in order to train the classifier) because -

- Obtaining a prediction from a classifier could potentially be faster than scanning a database for a user's average rating for a genre if the user-base is large. This operation will have to be done many times in this technique.
- Classifiers allow for noise in the class labels. Thus, instead of storing enough information to get the exact value of an aggregate in a time window, approximation techniques can be used.
- Using a classifier will allow the scheme to be used for new users, as mentioned earlier.

V. Introducing Diversity Without Sacrificing Accuracy

Suppose the recommender system has N users and for every genre g_i, we know the fraction of users f_i who have a high -average rating for the genre. The aim is that across the top-N lists of all users, all genres get represented. In order to obtain a more concrete formulation of this objective, it is assumed that the recommender system takes a snapshot of the archived ratings at some point of time and generates recommendation lists for all users based on this. The lists are supplied to the users as needed. The same procedure is repeated periodically to take new ratings into account. Thus, the recommender is offline.

Then, the diversity objective can be formally expressed in the following manner. The expected number of times a genre is represented across the top-N lists of all users is at least k, for every run of the recommendation algorithm.

When creating the recommendation list for a user, we make use of some ordinary recommender system to generate a top-N list. Then, one or more of the movies in this list is potentially replaced by a movie from a specific genre, if the user is sufficiently favourable towards that genre. In order to do this, we define a probability distribution over the genres such that p_i is the probability of choosing genre g_i. Let the number of movies that could potentially be changed in the list be a. Then, the procedure employed for each user is as follows -

Use the base recommender to obtain a top-N list for the user
for $i \leftarrow 1$ ***to*** a **do**
  Sample a genre g from the distribution over the genres
  if *there is no item in the top-N list in genre* g **then**
    if *average rating of the user for genre* g *is high* **then**
      IndexToReplace $\leftarrow$ Index of a movie in the top-N list which is present in the genre most represented so far
      NewMovie $\leftarrow$ with the highest ranked movie for the user in genre g. Replace the movie at index *IndexToReplace* with *NewMovie*
    end
  end
end

The sampling of the genre is independent of whether the user rates that genre highly. Then, for a user, the probability that in the place of one replaceable item, genre g_i gets introduced is,

P(Genre g_i gets introduced in one trial of a user)
= P(g_i is sampled).P(User rates g_i highly)
= $p_i f_i$

Then, in a trials of the user, the expected number of times genre g_i gets inroduced is $ap_i f_i$. Then, for N users, the expected number of times genre g_i gets inroduced is $Nap_i f_i$.

Let there be G genres. Then, given N, a, k, we need to define a probability distribution $\{p_i\}_{1 \leq i \leq G}$ such that $\sum_{i=1}^{G} Nap_i f_i \geq k$ and $\sum_{i=1}^{G} p_i = 1$. If such a probability distribution is used in the algorithm, as the expected number of times a genre is artificially introduced into the lists is k, the expected number of times it occurs in the lists is at least k, thus satisfying the required diversity criterion.

Also, it is a reasonable assumption to make that the rating of the user for the replaced movie is at least as high as the average rating for the genre considered during the replacement, as the movie being introduced is the one with the highest rating in the genre for that user. Thus, if t is the threshold above which an average rating is considered high, since at most Na items are replaced, each of which can have a rating of at most 5, the total rating falls by at most $Na(5-t)$. The values of a and t can be manipulated to keep this above a desired threshold. Thus, accuracy of the recommender can be maintained by this method.

VI. Experiment

A. Obtaining genre level aggregates

There are 19 possible genres in the MovieLens dataset, which are not completely independent. By combining this with the rating information, the average rating of each user for each

genre and the fraction of ratings greater than or equal to 4 of each user for each genre have been computed using MapReduce. Since a movie can belong to multiple genres, it is possible that it contributes to the aggregates for more than one genre. This was done as follows -

1) Join the <user, movie, rating> tuples that can be extracted from the file u.data and the <movie, genre> tuples that can be extracted from the file u.item, to obtain <user, movie, genre, rating> tuples.
2) Aggregate the rating of tuples obtained in step 1 using user and genre as grouping attributes to get th erequired aggregate rating value. This results in a set of <user, genre, aggregateRating> tuples.
3) Convert the tuples obtained in step 2 into a matrix for convenience. Each row of the matrix correspons to a user and each column corresponds to a genre.

B. Classification of users based on demographic data

The user demographic data has one numeric attribute - age and 2 nominal attributes - gender and occupation. These were used as feature vectors and separate classifiers were built for average rating and the farction of ratings that are at least 4, for each genre. The average rating was divided into 3 classes - High (>3.5), Medium (2.5-3.5) and Low (<2.5). The classifiers were built using Weka.

It was observed that treating age as a nominal attribute by group ages 0-10, 11-20 etc was not found to be helpful. This is probably because Weka does not support ordinal attributes. When age is treated as an unordered nominal attribute, the distance between 0-10 and 11-20 is the same as that between 0-10 and 21-30, which results in loss of information.

Many classifiers were experimented with on these such as Naive Bayes, Bayes Network, SVM (which uses a 0-1 distance measure for nominal attributes), logistic, classification trees and some ensemble classifers. The best result was obtained using the RandomCommittee classifier. It was also observed that resampling the data to balance class skew in those genres where this was present did not improve the accuracy or f-measures for these genres (it resulted in decrease of precision for the minority class without an increase in recall).

It has been observed that the classifiers are not very good in the sense that though they result in high f-scores on the training set, the performance obtained using 10-fold cross validation is poor. However, it is not clear whether the poor results obtained during cross validation are due to the skewed nature of the dataset (which makes one fold too small to classify effectively) or some inherent failing of the classifier. It seemed unlikely that the high values obtained in the trianing phase are due to overfitting due to the random nature of the classifier used. Also, it was found in later steps that this weak classifier was good enough to obtain perceptible changes using the technique introduced.

The classifcation on the fraction of ratings that are at least 4 resulted in similar results. This is expected as there is fairly high correlation between the two outputs.

Results for classification of average rating on training data

Genre	Measure	High	Medium	Low
Unknown	True Count	5	3	935
	Precision	0.8	0.5	0.9968
	Recall	0.8	0.3333	0.9979
	F-Score	0.8	0.4	0.9973
Action	True Count	518	369	56
	Precision	0.7630	0.8848	1
	Recall	0.9633	0.6450	0.3571
	F-Score	0.8515	0.7461	0.5263
Adventure	True Count	474	373	96
	Precision	0.7547	0.8295	0.9091
	Recall	0.9283	0.6783	0.5208
	F-Score	0.8325	0.7463	0.6623
Animation	True Count	369	227	347
	Precision	0.7287	0.7313	0.8526
	Recall	0.9024	0.6476	0.7003
	F-Score	0.8063	0.6869	0.7690
Children	True Count	361	343	239
	Precision	0.7161	0.7764	0.8589
	Recall	0.9086	0.7289	0.5858
	F-Score	0.8010	0.7519	0.6965
Comedy	True Count	439	462	42
	Precision	0.7896	0.8383	0.875
	Recall	0.8633	0.7965	0.5
	F-Score	0.8248	0.8169	0.6364
Crime	True Count	563	297	83
	Precision	0.7726	0.88	0.9428
	Recall	0.9716	0.5926	0.3976
	F-Score	0.8607	0.7082	0.5593
Documentary	True Count	220	95	628
	Precision	0.6814	0.6875	0.8797
	Recall	0.7682	0.5789	0.8615
	F-Score	0.7222	0.6286	0.8705
Drama	True Count	660	270	13
	Precision	0.8430	0.8508	1
	Recall	0.9682	0.5704	0.3077
	F-Score	0.9013	0.6829	0.4706
Fantasy	True Count	205	217	521
	Precision	0.7083	0.6991	0.8513
	Recall	0.7463	0.7604	0.8023
	F-Score	0.7268	0.7285	0.8261
Film-Noir	True Count	433	148	362
	Precision	0.7358	0.7363	0.8152
	Recall	0.9007	0.5473	0.6823
	F-Score	0.8100	0.6279	0.7428
Horror	True Count	315	376	252
	Precision	0.6961	0.7613	0.8619
	Recall	0.8508	0.7633	0.6190
	F-Score	0.7657	0.7623	0.7205

Musical	True Count	356	304	283
	Precision	0.6829	0.7601	0.8571
	Recall	0.8652	0.7401	0.5936
	F-Score	0.7633	0.7500	0.7015
Mystery	True Count	513	329	101
	Precision	0.7428	0.8306	0.975
	Recall	0.9571	0.6109	0.3861
	F-Score	0.8364	0.7040	0.5532
Romance	True Count	605	311	27
	Precision	0.8063	0.8960	1
	Recall	0.9768	0.5820	0.2963
	F-Score	0.8834	0.7056	0.4571
Sci-fi	True Count	496	353	94
	Precision	0.7682	0.8587	0.9298
	Recall	0.9556	0.6544	0.5638
	F-Score	0.8518	0.7428	0.7020
Thriller	True Count	505	397	41
	Precision	0.7476	0.8772	0.9375
	Recall	0.9505	0.6297	0.3658
	F-Score	0.8370	0.7331	0.5263
War	True Count	649	248	46
	Precision	0.8324	0.9161	0.9444
	Recall	0.9877	0.5726	0.3696
	F-Score	0.9034	0.7047	0.5312
Western	True Count	285	173	485
	Precision	0.7304	0.7230	0.8298
	Recall	0.8175	0.6185	0.8144
	F-Score	0.7715	0.6667	0.8221

Results for classification of average rating using 10-fold cross validation

Genre	Measure	High	Medium	Low
Unknown	True Count	5	3	935
	Precision	0	0	0.9915
	Recall	0	0	0.9957
	F-Score	0	0	0.9936
Action	True Count	518	369	56
	Precision	0.5709	0.4346	0.0968
	Recall	0.6679	0.3604	0.0536
	F-Score	0.6156	0.3941	0.0690
Adventure	True Count	474	373	96
	Precision	0.5224	0.4029	0.1452
	Recall	0.5907	0.3726	0.0938
	F-Score	0.5544	0.3872	0.1139
Animation	True Count	369	227	347
	Precision	0.4378	0.2442	0.4384
	Recall	0.5339	0.2335	0.3487
	F-Score	0.4811	0.2387	0.3884
Children	True Count	361	343	239
	Precision	0.4345	0.3787	0.3176
	Recall	0.5235	0.3732	0.2259
	F-Score	0.4749	0.3759	0.2641
Comedy	True Count	439	462	42
	Precision	0.4886	0.5057	0.0870
	Recall	0.5398	0.4762	0.0476
	F-Score	0.5130	0.4905	0.0615
Crime	True Count	563	297	83
	Precision	0.5926	0.2982	0.08
	Recall	0.7105	0.2188	0.0482
	F-Score	0.6462	0.2524	0.0602
Documentary	True Count	220	95	628
	Precision	0.2331	0.0941	0.6624
	Recall	0.25	0.0842	0.6561
	F-Score	0.2412	0.0889	0.6592
Drama	True Count	660	270	13
	Precision	0.7091	0.2879	0
	Recall	0.8015	0.2037	0
	F-Score	0.7525	0.2386	0
Fantasy	True Count	205	217	521
	Precision	0.2547	0.2759	0.5551
	Recall	0.2634	0.2949	0.5317
	F-Score	0.2590	0.2851	0.5431
Film-Noir	True Count	433	148	362
	Precision	0.4774	0.1376	0.4178
	Recall	0.5843	0.1013	0.3508
	F-Score	0.5254	0.1167	0.3814
Horror	True Count	315	376	252
	Precision	0.3757	0.4010	0.3048
	Recall	0.4317	0.4202	0.2261
	F-Score	0.4018	0.4104	0.2597
Musical	True Count	356	304	283
	Precision	0.3309	0.3278	0.3219
	Recall	0.3792	0.3256	0.2650
	F-Score	0.3534	0.3267	0.2907
Mystery	True Count	513	329	101
	Precision	0.5260	0.3275	0.0339
	Recall	0.6120	0.2857	0.0198
	F-Score	0.5658	0.3052	0.0250
Romance	True Count	605	311	27
	Precision	0.6541	0.35	0
	Recall	0.7471	0.2701	0
	F-Score	0.6975	0.3049	0
Sci-fi	True Count	496	353	94
	Precision	0.5658	0.4034	0.1471
	Recall	0.6673	0.3314	0.1064
	F-Score	0.6124	0.3639	0.1234
Thriller	True Count	505	397	41
	Precision	0.5339	0.4012	0
	Recall	0.6237	0.3325	0
	F-Score	0.5753	0.3636	0
War	True Count	649	248	46
	Precision	0.7106	0.3296	0.1904
	Recall	0.8135	0.2379	0.0869
	F-Score	0.7586	0.2763	0.1194
Western	True Count	285	173	485
	Precision	0.3671	0.2130	0.5546
	Recall	0.4070	0.2081	0.5237
	F-Score	0.3860	0.2105	0.5387

C. Recommender101

In order to conduct validate the theories about the usefulness of context information, experiments had to be carried out on the dataset. Recommender101 is a lightweight and easy-to-use

framework written in Java to carry out offline experiments for Recommender Systems. It provides implementations of many standard recommenders and by default works on the movielens dataset. The source and further information about Recommender101 is available at http://ls13-www.cs.uni-dortmund.de/homepage/recommender101/index.shtml.

The proposed algorithm was implemented as an additional recommender in the Recommender101. The framework was convenient because the system already contained implementations of many well-known recommender systems that could be used as base recommender systems. Also, it provided a clean method implement mechanisms to evaluate recommendation lists.

D. Implementation of the Diversity Algorithm

The most important class created was the recommender itself, called DiverseRecommender. This has a field which is a base classifier. For predicting specifc ratings, this made use of the base recommender and obtained a top-N list from the base recommender and used the above-mentioned algorithm to modify the list appropriately. The probability distribution used to sample a genre had to be chosen such that a genre that not many users rate highly is sampled more often (thus increasing the probability of it's count increasing, as the aim is to make the counts more equalized). This was done as follows. The fraction of users that rated a genre highly was obtained from the classifier (the classifier makes this computation from the training data alone). Let the fraction of users who rated genre i highly be fracHighs(i). Then, the probability of sampling genre i, which was the g_i, used in the algorithm, used was

$$g_i = \frac{\sum_i fracHighs(i) - fracHighs(i)}{\sum_i fracHighs(i) * (G-1)}$$

where G is the number of genres. The numerator of this expression was chosen to ensure that as the fraction of users likely to rate a genre highly increases, the genre is sampled less frequently. The denominator of the expression is a normalization constant. This strategy was empirically found to be more effective than sampling genres not recommended much so far more often.

The classifier used was the RandomCommittee classifier from Weka, that was found earlier to produce the best results. The aggregate values were pre-computed as meantioned earlier and loaded from files as needed.

One of the basic primitives needed by the algorithm is the genre of a movie. A special class was created for this purpose that stored a hash-map of the genres a movie was present in.

The two evaluation metrics needed - calculation of the total rating and entropy over all top-N lists also had to be implemented.

Suitable control experiments were also set up to validate the need for many of the assumptions used in the algorithm. First, identical recommenders were set up using the classifier on average rating and fraction of ratings of atleast 4 respectively, each divided into 3 classes - high, medium, low. It was observed that the resultant entropy values and total rating were almost identical in almost all cases for both. Thus, the classifier that used the fraction of ratings of atleast 4 was modified into a binary classifier. The results shown below for DiverseClassifierFractionOfAtleast4s correspond to the recommender that uses this classifier.

Another modifcation was while choosing the index of the movie to be replaced. Since a movie could be present in multiple genres, there was a possibility that a movie placed in one genre that is very well-represented and another that is very poorly represented gets removed because the well-represented genre is currently the one with the maximum representation. But this may not be desirable because one representative of the poorly represented genre gets lost in this manner. Thus, for each movie, the genre which it was placed under that had least representation was considered and the movie with the maximum value for this was removed. This was done in the recommender called DiverseRecommenderMoreCarefulRemoval.

To test that the classification of users was useful, a control experiment was set up where instead of checking if a user typically rates movies from a genre under consideration highly, a random number is chosen between 0 and 1. If this is less than the fraction of users who rate that genre highly (an event with same probability as the classification returning a "high"), a movie from that genre was introduced. This accounts for the fact that soem genres are more popular than otehrs but does not check which users actually prefer which genres. This experiment was done by the recommender called RandomDiverseRecommender.

To test that any sort of check made after choosing the genre to replace is necessary, another recommender was created that does no such check. This was called UnconstrainedRandomDiverseRecommender.

E. Recommendation Results

The performance of the different recommenders using various base recommenders is tabulated below. They are listed in alphabetical order of the names of the base recommenders .

Base recommender: BP-RMF Recommender [6]

Recommender	Entropy	% Increase in Entropy	Total Rating	% Decrease in Total Rating
BPRMFRecommender	3.476	0	7940	0
DiverseRecommender	3.477	0.03	7864	0.96

DiverseRecommender MoreCarefulRemoval	3.477	0.03	7866	0.93
DiverseRecommender FractionOfAtleast4s	3.48	0.12	7507	5.45
RandomDiverseRecommender	3.478	0.06	7677	3.31
UnconstrainedRandomDiverseRecommender	3.481	0.14	7404	6.75

Base recommender: Content Based Recommender

Recommender	Entropy	% Increase in Entropy	Total Rating	% Decrease in Total Rating
ContentBased Recommender	3.366	0	1293	0
DiverseRecommender	3.375	0.27	1288	0.39
DiverseRecommender MoreCarefulRemoval	3.376	0.3	1288	0.39
DiverseRecommender FractionOfAtleast4s	3.403	1.1	1256	2.86
RandomDiverse Recommender	3.391	0.74	1276	1.31
UnconstrainedRandomDiverseRecommender	3.413	1.4	1256	2.86

Base recommender: Factorized Neighbourhood Recommender [7]

Recommender	Entropy	% Increase in Entropy	Total Rating	% Decrease in Total Rating
Factorized Neighbourhood Recommender	3.546	0	6672	0
DiverseRecommender	3.548	0.06	6672	0
DiverseRecommender MoreCarefulRemoval	3.548	0.06	6672	0
DiverseRecommender FractionOfAtleast4s	3.552	0.17	6665	0.1
RandomDiverse Recommender	3.551	0.14	6672	0
UnconstrainedRandomDiverseRecommender	3.554	0.23	6655	0.25

Base recommender: Baseline SVD Recommender

Recommender	Entropy	% Increase in Entropy	Total Rating	% Decrease in Total Rating
FunkSVD Recommender	3.479	0	6874	0
DiverseRecommender	3.48	0.03	6865	0.13
DiverseRecommender MoreCarefulRemoval	3.48	0.03	6859	0.22
DiverseRecommender FractionOfAtleast4s	3.483	0.11	6778	1.4
RandomDiverse Recommender	3.482	0.09	6783	1.32
UnconstrainedRandomDiverseRecommender	3.485	0.17	6725	2.17

Base recommender: LibFm Recommender [8]

Recommender	Entropy	% Increase in Entropy	Total Rating	% Decrease in Total Rating
LibFmRecommender	3.476	0	7678	0
DiverseRecommender	3.478	0.06	7678	0
DiverseRecommender MoreCarefulRemoval	3.477	0.03	7671	0.09
DiverseRecommender FractionOfAtleast4s	3.481	0.14	7671	0.09
RandomDiverse Recommender	3.479	0.09	7666	0.16
UnconstrainedRandomDiverseRecommender	3.483	0.2	7660	0.23

Base recommender: Nearest Neighbours Recommender

Recommender	Entropy	% Increase in Entropy	Total Rating	% Decrease in Total Rating
Nearest Neighbours	3.539	0	8546	0
DiverseRecommender	3.54	0.03	8546	0
DiverseRecommender MoreCarefulRemoval	3.54	0.03	8541	0.06

DiverseRecommender FractionOfAtleast4s	3.545	0.17	8538	0.09
RandomDiverse Recommender	3.543	0.11	8543	0.04
UnconstrainedRandom DiverseRecommender	3.548	0.25	8535	0.13

Base recommender: RfRec Recommender [9]

Recommender	Entropy	% Increase in Entropy	Total Rating	% Decrease in Total Rating
RfRecRecommender	3.478	0	7255	0
DiverseRecommender	3.48	0.06	7255	0
DiverseRecommender MoreCarefulRemoval	3.48	0.06	7255	0
DiverseRecommender FractionOfAtleast4s	3.483	0.14	7245	0.14
RandomDiverse Recommender	3.482	0.12	7245	0.14
UnconstrainedRandom DiverseRecommender	3.486	0.23	7245	0.14

Base recommender: SlopeOne Recommender [10]

Recommender	Entropy	% Increase in Entropy	Total Rating	% Decrease in Total Rating
SlopeOne Recommender	3.542	0	8120	0
DiverseRecommender	3.544	0.06	8120	0
DiverseRecommender MoreCarefulRemoval	3.544	0.06	8115	0.06
DiverseRecommender FractionOfAtleast4s	3.549	0.2	8111	0.11
RandomDiverse Recommender	3.546	0.12	8111	0.11
UnconstrainedRandom DiverseRecommender	3.551	0.25	8108	0.15

Base recommender: Weighted Average Recommender

Recommender	Entropy	% Increase in Entropy	Total Rating	% Decrease in Total Rating
Weighted Average Recommender	3.476	0	7512	0
DiverseRecommender	3.476	0.06	7512	0
DiverseRecommender MoreCarefulRemoval	3.478	0.06	7512	0
DiverseRecommender FractionOfAtleast4s	3.481	0.14	7500	0.16
RandomDiverse Recommender	3.48	0.11	7505	0.09
UnconstrainedRandom DiverseRecommender	3.483	0.2	7500	0.16

VII. Observations

It is observed that the increase in the entropy is very small, of the order of 0.1 % in most cases. On the positive side, the accuracy also drops only by about the same percentage, in many cases not at all, for the DiverseRecommender. Since the entropy is a small value characterizing an array of relatively large values, a 0.1 % increase may also be useful in many cases.

It is observed that the total rating of DiverseRecommender is always greater than or equal to the two variants that introduce randomness. Thus, it may be concluded that the classification is useful in determining whether it is appropriate to intoduce a genre for a specific user. Also, the unconstrained recommender typically incurs a much higher penalty in the total rating, which indicates that the popularity of a genre must be taken into account when introducing it.

The results can also be seen in a different light. Though the classifier results in a marginally better total rating, it also results in a marginally lower entropy. Thus, the random recommenders, which require far less computation, can safely be used as they do not perform significantly worse in terms of accuracy and to compensate for that, provide a gain in diversity.

It is also observed that the recommender DiverseFractionOfAtleast4s consistently results in a higher diversity that DiverseRecommender but often, the total rating is worse than even RandomDiverseRecommender. This indicates that the 2-class classification is not a sufficiently stringent constraint to prevent loss of accuracy. The exact threshold for both classifiers, however, is specific to the dataset. The threshold here have been chosen as the values that are relatively easy

for the classifier to manage, while still being useful.

Also, there is no such relation between the results of DiverseRecommender and DiverseRecommenderMoreCarefulRemoval. Thus, neither method of choosing the movie to be replaced appears to be better than the other.

It is observed that the increase in entropy for the content based classifier is significantly higher than for the others. However, it can be observed that the total rating for this classifier is quite low. This is probably because the recommender is unable to recommend items to many users due to insufficient information. Hence, no inference can be made about whether the technique is more amenable to such a recommender. The technique appears to work equally well on all other recommenders.

One possible reason for the poor results is that the algorithm implicitly assumes that a movie belongs to a single genre, which is generally not the case. It is possible that this technique would work better in such a setting. The problem with a movie being in multiple genres is that a popular genre could get introduced many times with many not-so-popular genres so that the gain in entropy obtained by introducing the not-so-popular genre is offset by the popular one.

VIII. Conclusions

The technique proposed for increasing the diversity of recommendations, on testing on the MovieLens dataset was found to yield an increase, but not a particularly impressive one. However, many assumptions made while devising the algorithm such as the need to consider the popularity of a genre and the innate preference of the user for the genre were validated could potentially be better exploited for this purpose.

One of the main plus points of this technique is that it is agnostic to the base recommender being used. Thus, it can be easily added on top of any existing recommender. Also, this introduces another potential solution to the problem of recommending items to a new user.

IX. Extending the Idea to Other Objectives

Though the technique proposed here was tailored for the objectives accuracy and diversity, it can be applied to other objectives where external contextual information appears relevant, with suitable modifications. The basic idea is that there is a primary objective(s) already being maximized by an existing recommender system. In order to maximize the secondary objective, items are divided into groups derived from the second objective. For example, if the objective was profit, items would be divided into groups where each group corresponds to a price range. This does not suffer from the problem with grouping movies into genres because an item would lie only in one price range, as opposed to a movie being in multiple genres.

Following this, a classification is done to predict whether a user typically likes items in a particular group. For instance with the accuracy-profit case, accuracy would correspond to buying the item. Thus, the classifer would have to predict whether the user would buy an item in a particular price range.

Then, while creating a top-N list for a user, groups to be introduced are picked using a suitable probability distribution and the classifier is used to determine whether a replacement may occur. If so, the ”best” item from the group replaces the ”worst” item in the list.

Acknowledgment

The author would like to thank Dr B Ravindran and Mr Deepak Pai from Adobe Research for their assistance in identifying the project area, topic and the invaluable advice during the course of the project.

References

[1] G. Adomavicius and A. Tuzhilin, “Context-aware recommender systems,” in *Recommender Systems Handbook*, F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, Eds. Springer, 2011, pp. 217–253.

[2] T. Jambor and J. Wang, “Optimizing multiple objectives in collaborative filtering,” in *RecSys*, X. Amatriain, M. Torrens, P. Resnick, and M. Zanker, Eds. ACM, 2010, pp. 55–62.

[3] A. Das, C. Mathieu, and D. Ricketts, “Maximizing profit using recommender systems,” *CoRR*, vol. abs/0908.3633, 2009.

[4] M. T. Ribeiro, A. Lacerda, A. Veloso, and N. Ziviani, “Pareto-efficient hybridization for multi-objective recommender systems,” in *RecSys*, P. Cunningham, N. J. Hurley, I. Guy, and S. S. Anand, Eds. ACM, 2012, pp. 19–26.

[5] K. M. Svore, M. Volkovs, and C. J. C. Burges, “Learning to rank with multiple objective functions,” in *WWW*, S. Srinivasan, K. Ramamritham, A. Kumar, M. P. Ravindra, E. Bertino, and R. Kumar, Eds. ACM, 2011, pp. 367–376.

[6] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, “Bpr: Bayesian personalized ranking from implicit feedback,” *CoRR*, vol. abs/1205.2618, 2012.

[7] Y. Koren, “Factor in the neighbors: Scalable and accurate collaborative filtering,” *TKDD*, vol. 4, no. 1, 2010.

[8] S. Rendle, “Factorization machines with libfm,” *ACM TIST*, vol. 3, no. 3, p. 57, 2012.

[9] F. Gedikli, F. Bagdat, M. Ge, and D. Jannach, “Rf-rec: Fast and accurate computation of recommendations based on rating frequencies,” in *CEC*, B. Hofreiter, E. Dubois, K.-J. Lin, T. Setzer, C. Godart, E. Proper, and L. Bodenstaff, Eds. IEEE, 2011, pp. 50–57.

[10] D. Lemire and A. Maclachlan, “Slope one predictors for online rating-based collaborative filtering,” *CoRR*, vol. abs/cs/0702144, 2007.