Splitting Approaches for Context-Aware Recommendation: An Empirical Study

Yong Zheng
Center for Web Intelligence
DePaul University
Chicago, Illinois, USA
yzheng8@cs.depaul.edu

Robin Burke
Center for Web Intelligence
DePaul University
Chicago, Illinois, USA
rburke@cs.depaul.edu

Bamshad Mobasher
Center for Web Intelligence
DePaul University
Chicago, Illinois, USA
mobasher@cs.depaul.edu

ABSTRACT

User and item splitting are well-known approaches to context-aware recommendation. To perform item splitting, multiple copies of an item are created based on the contexts in which it has been rated. User splitting performs a similar treatment with respect to users. The combination of user and item splitting: UI splitting, splits both users and items in the data set to boost context-aware recommendations. In this paper, we perform an empirical comparison of these three context-aware splitting approaches (CASA) on multiple data sets, and we also compare them with other popular context-aware collaborative filtering (CACF) algorithms. To evaluate those algorithms, we propose new evaluation metrics specific to contextual recommendation. The experiments reveal that CASA typically outperform other popular CACF algorithms, but there is no clear winner among the three splitting approaches. However, we do find some underlying patterns or clues for the application of CASA.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information filtering

General Terms

Algorithm, Experiment, Performance

Keywords

Recommendation, Context, Context-aware recommendation, Context-aware splitting approaches

1. INTRODUCTION

The standard formulation of the collaborative recommendation problem begins with a two-dimensional (2D) matrix of ratings, organized by user and item: *Users* $\times$ *Items* $\rightarrow$ *Ratings*. Recommendation then becomes a prediction problem, interpolating new ratings not present in the original matrix. Context-aware recommender systems (CARS) add contextual variables to this question, making use of the context in which a rating was made and the context in which a recommendation is sought, in order to add nuance to the resulting recommendations [1]. The context-aware approach has been demonstrated to be effective in several domains [4, 12, 7, 16].

One popular and efficient technique for context-aware recommendation is *item splitting* [4, 5], where multiple copies of an item are created to hold ratings generated in different context. For example, if restaurant R receives different ratings at lunch time and at dinner, two new items could be created R_d and R_l. The dinnertime ratings would be associated only with R_d and the lunch ratings with R_l as if these were entirely significant different entities. When a recommendation is sought, only those items matching the current context are considered. The benefit of this approach is that it does not require a new recommendation algorithm: an algorithm that works on the original 2D matrix can also be applied to the new transformed matrix. *User splitting* [2, 12], which splits users instead of items, is a similar pre-filtering solution. And, *UI splitting*, a combination of item splitting and user splitting was demonstrated to work better [17].

As an empirical study, our contributions in this paper can be summarized as follows: 1). We formally evaluate those three context-aware splitting approaches (CASA) over multiple data sets. It is important and necessary because splitting approaches were reported as the most efficient context-aware recommendation algorithms by several researchers, but there are no existing work contributing an empirical comparison of them. 2). In addition, we also compare the recommendation performance with other popular context-aware collaborative filtering (CACF) approaches. 3). To evaluate those context-aware approaches, where traditional precision and recall may not work, we introduce new evaluation metrics, named as contextual precision (CPrecision) and contextual ROC (CROC), which we propose as standard metrics for CARS evaluations.

The remainder of this paper is organized as follows. Section 2 positions related work. Section 3 presents the three CASA: item splitting, user splitting and UI splitting. Section 4 and 5 discusses the experimental evaluations and empirical comparisons among three splitting approaches as well as other CACF algorithms, followed by the conclusions and future work in Section 6.

2. RELATED WORK

As described in [1], there are three basic approaches to develop context-aware recommendation algorithms: pre-filtering, post-filtering, and contextual modeling. A pre-filtering approach applies a context-dependent criterion to the list of items, selecting only those appropriate to a given context. Only the filtered items are considered for recommendation. A post-filtering approach is similar but applies the filter after recommendations have been computed. Contextual modeling takes contextual considerations into the recommendation algorithm itself.

Among all those contextual recommendation algorithms, pre-

SAC'14 March 24-28, 2014, Gyeongju, Korea.

http://dx.doi.org/10.1145/2554850.2554989

filtering is popular as it is reasonably straightforward, and can be applied with most recommendation techniques. Item splitting [4, 5] is considered as one of the most efficient pre-filtering algorithms and it has been well developed in recent research. The underlying idea of item splitting is that the nature of an item, from the user's point of view, may change in different contextual conditions, hence it may be useful to consider it as two different items [5].

User splitting is based on a similar intuition – it may be useful to consider one user as two different users, if he or she demonstrates significantly different preferences across contexts. The user splitting approach is not as well-developed as item splitting. In 2009, Baltrunas *et al.* proposed *micro-profiling* [2], which was the first attempt to examine user splitting. Said et al. made a similar proposal called *contextual user profiles* [12]. Neither of these studies performed comprehensive evaluation of the approach. The micro-profiling work examined only temporal splits. The work on contextual user profiles was limited to consideration of a single binary contextual condition (i.e. there are only two values for a context feature). And there are no existing work for the comparison between user and item splitting. *UI splitting*, a combination of item splitting and user splitting was first proposed and explored in [17], but the results were just based on a single data set and evaluated only on root-mean-square error (RMSE) values. In this paper, we continue to provide a comprehensive empirical study of those approaches over multiple datasets and evaluation metrics.

3. METHODOLOGY

To better understand and represent the splitting approaches, consider the following movie recommendation example:

Table 1: Movie Ratings in Contexts

User	Item	Rating	Time	Location	Companion
U1	T1	3	Weekend	Home	Friend
U1	T1	5	Weekend	Cinema	Girlfriend
U1	T1	?	Weekday	Home	Family

In Table 1, there are one user $U1$, one item $T1$ and two ratings (the first two rows) in the training and one unknown rating that we are trying to predict (the third row). There are three contextual dimensions – time (weekend or weekday), location (at home or cinema) and companion (friend, girlfriend or family). In the following discussion, we use *contextual dimension* to denote the contextual variable, e.g. "Location" in this example. The term *contextual condition* refers to a specific value in a contextual dimension, e.g. "home" and "cinema" are two contextual conditions for "Location".

3.1 Item Splitting

Item splitting tries to find a contextual condition on which to split each item. The split should be performed once the algorithm identifies a contextual condition in which items are rated significantly differently. In the movie example above, there are three contextual conditions in the dimension $companion$: friend, girlfriend and family. Correspondingly, there are three possible alternative conditions: "*friend and not friend*", "*girlfriend and not girlfriend*", "*family and not family*". Impurity criteria [4] are used to determine whether and how much items were rated differently in these alternative conditions, e.g. a t-test or other statistical metrics can be used to evaluate if the means differ significantly across conditions.

Item splitting iterates over all contextual conditions in each context dimension and evaluates the splits based on the impurity criteria. It finds the best split for each item in the rating matrix and then items are split into two new ones, where contexts are eliminated from the original matrix – it transforms the original multi-dimensional rating matrix to a 2D matrix as a result. Assume that the best contextual condition to split item T1 in Table 1 is "Location = *home and not home*", T1 can be split into T11 (movie T1 being seen at home) and T12 (movie T1 being seen not at home). Once the best split has been identified, the rating matrix can be transformed as shown by Table 2(a).

This example shows a *simple split*, in which a single contextual condition is used to split the item. It is also possible to perform a *complex split* using multiple conditions across multiple context dimensions. However, as discussed in [5], there are significant costs of sparsity and potential overfitting when using multiple conditions. Usually, simple splitting is applied to avoid the serious problem, where we applied the simple splitting in this work for all splitting approaches. Complex splitting will be explored in future work.

3.2 User Splitting

Similarly, user splitting tries to split users instead of items. It can be easily derived from item splitting as introduced above. Similar impurity criteria can also be used for user splits. Assume that the best split for user $U1$ in Table 1 is "Companion = *family and not family*", $U1$ can be split into $U11$ ($U1$ saw the movie with family) and $U12$ ($U1$ saw the movie with others). The rating matrix can be transformed as shown by Table 2(b). The first two rows contain the same user $U12$ because $U1$ saw this movie with others (i.e. not family) rather than family as shown in the original rating matrix.

Table 2: Transformed Rating Matrix

(a) by Item Splitting

User	Item	Rating
U1	T11	3
U1	T12	5
U1	T11	?

(b) by User Splitting

User	Item	Rating
U12	T1	3
U12	T1	5
U11	T1	?

(c) by UI Splitting

User	Item	Rating
U12	T11	3
U12	T12	5
U11	T11	?

3.3 UI Splitting

UI splitting [17] applies item splitting and user splitting together. Assuming that the best split for item and user splitting are the same as described above, the rating matrix based on UI splitting can be shown as Table 2(c). Here we see that both users and items were transformed, creating new users and new items. The process is simply an application of item splitting followed by user splitting on the resulting output.

4. EXPERIMENTS SETUP

In this section, we introduce the data sets, new evaluation metrics and the specific configurations in our experiments.

4.1 Data Sets

We examine item splitting, user splitting and UI splitting with three data sets: Food, Movie and LDOS-CoMoDa. Food and Movie are two well-known data sets for contextual recommendation. LDOS-CoMoDa is another movie dataset [10] collected from surveys. After filtering out subjects with less than 5 ratings and rating records with incomplete features, we got the final data sets described in Table 3. Descriptions of the contexts can be found in [16, 10].

4.2 Experimental Goals

There are two comparisons to be explored in this study:

Table 3: Recommendation Data Sets

	Food Data	Movie Data	LDOS-CoMoDa
# of users	212	69	113
# of items	20	176	1186
# of ratings	6360	1010	2094
# of contexts	2	3	12
Rating Scale	1-5	1-13	1-5
Density	Dense	Sparse	Sparse

1. *Comparison among context-aware splitting approaches*: which splitting approaches work the best? which splitting criteria are the best appropriate ones? any underlying clues to indicate which splitting approach should be used?

2. *Comparison between splitting approaches and other context-aware algorithms*: how the splitting approaches perform competing with other context-aware recommender algorithms?

4.3 Splitting Configurations

To evaluate the performances of our algorithms, we use four splitting criteria described in [5]: t_{mean}, t_{chi}, t_{prop} and t_{IG}. t_{mean} estimates the statistical significance of the difference in the means of ratings associated to each alternative contextual condition using t-test. t_{chi} and t_{prop} estimates the statistical significance of the difference between two proportions – high ratings (>R) and low ratings (≤R) by chi square test and z-test respectively, where we choose R = 3 for Food and LDOS-CoMoDa data, and R = 7 for the Movie data, which are the same values used in [5]. t_{IG} measures the information gain given by a split to the knowledge of the item i rating classes which are the same two proportions as above.

Usually, a threshold for the splitting criteria should be set so that users or items will only be split when the criteria meets the significance requirement. We use an arbitrary value of 0.2 in the t_{IG} case. For t_{mean}, t_{chi} and t_{prop}, we use 0.05 as the p-value threshold [1]. We deem it as a significant split once the p-value is no larger than 0.05. We rank all significant splits by the impurity value, and choose the top first (highest impurity) as the best split. Items or users without qualified splitting criteria are left unchanged.

4.4 Evaluation Metrics

There are three popular classes of evaluation metrics in the domain of recommender systems: prediction errors (e.g. MAE, RMSE), IR metrics (e.g. precision, recall, ROC) and ranking metrics (e.g. MAP, NDCG). In this paper, we chose the first two classes of metrics as these are the most commonly employed. We will explore ranking-oriented metrics in future work. We measured RMSE in our experiments using 5-fold cross validation. But, traditional IR metrics do not work on the evaluation of contextual recommendation, especially the comparison among CASA, because the number of users and items may differ after the splitting process, and it is not comparable to other CACF algorithms if evaluations are directly based on the transformed 2D rating matrix. IR metrics depend on user-specific relevance judgement, where relevance is assessed by looking at user ratings in traditional recommendation, but in CARS, relevance is also a function of the context.

Precision [8] is defined as the ratio of relevant items selected to number of items selected, and recall presents the probability that a relevant item will be selected. For contextual recommendation, we make the concepts related to the context and define the measures CPrecision and CRecall. When we perform a retrieval operation on our test set, we query using a user and a context. An item counts towards our CPrecision score if it was preferred by the user in the same context. Similarly, for CRecall, the set of possible recommendations whose cardinality is the denominator of the recall calculation is the set of items preferred in the retrieval context.

These same principles can be applied to measure context-relative ROC scores. The ROC curve measures the ability of the information filtering system to tell signal (relevant user-item pairs) from noise (items that are irrelevant for users). Our CROC measure is a similar curve drawing by false positive rate (i.e. fallout) v.s. true positive rate (i.e. recall), where those two rates are measured within specific contexts as similar as the CPrecision and CRecall above. In splitting approaches, we store mapping records (e.g. a mapping from the original UserID to new UserID, and also rolling back) in the memory, which guarantees and enables the comparable evaluations among multiple contextual algorithms – they are using the same 5-fold cross validation with original UserIDs, ItemIDs and contexts, and original user or item IDs will only be transformed to new ones (based on the mapping records) for the rating prediction purpose in CASA. In other words, the transformed rating matrix is only used in rating prediction process, they are mapped back to original ones when measuring IR metrics so that it is comparable to compete among CASA and with other CACF algorithms.

To apply the IR metrics, we need to identify items that are relevant and non-relevant. We set a threshold for each data and consider ratings above this value to count as relevant and those below as non-relevant to the associated user and context. We choose a threshold of 3 for Food and LDOS-CoMoDa data and 7 for the Movie data, which is the same value used in the splitting process as required.

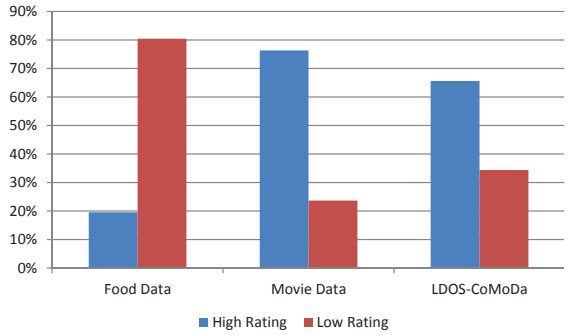

Figure 1: Rating Distribution

Figure 1 shows the split between high/relevent and low/irrelevent ratings in each data set. The Food dataset is unusual in that low ratings dominate. Most ratings datasets have many more postive ratings due to selection bias. With only a small number of positive ratings, CRecall becomes an unstable measure: highly-dependent on the number of positively-rated items in a given context. For this reason, we use CPrecision and CROC as our evaluation metrics, where CROC is a complement if CPrecision is not that reliable.

In our experiments on CASA, we evaluate the performance of three recommendation algorithms: user-based collaborative filtering (UBCF), item-based collaborative filtering (IBCF) and matrix factorization (MF). Our experiments with different matrix factorization algorithms did not reveal a clear winner. Of the three forms introduced by Koren [9]: MF with rating bias (BiasMF), Asymmetric SVD (AsySVD) and SVD++, we found that BiasMF was the best choice for the Food and LDOS-CoMoDa data, and AsySVD was the best for the Movie data. Thus we use "MF" to refer to the best MF technique for each data set.

We use the open source recommendation engine MyMediaLite

[1] A finer-grained operation is to set another threshold for each impurity value and each data set.

Table 4: Comparison of Predictive Performances (RMSE)

		Food Data				**Movie Data**				**LDOS-CoMoDa**			
	Alg	t_{mean}	t_{chi}	t_{prop}	t_{IG}	t_{mean}	t_{chi}	t_{prop}	t_{IG}	t_{mean}	t_{chi}	t_{prop}	t_{IG}
Item Splitting	UBCF	1.147	1.153	1.151	**1.200**	2.785	2.686	2.725	2.922	1.040	1.021	1.028	1.043
	IBCF	1.229	1.230	1.230	**1.232**	**2.855**	**2.822**	**2.793**	**2.896**	1.030	1.024	1.026	1.034
	MF	*1.036*	1.037	1.037	**1.083**	2.544	2.530	*2.514*	2.554	1.020	*1.011*	1.016	1.020
	SP	100%	100%	100%	100%	20.5%	34.7%	19.9%	15.9%	3.0%	6.0%	3.8%	3.5%
	Sparsity	25%	25%	25%	25%	92.8%	93.1%	93.8%	93.3%	98.5%	98.5%	98.5%	98.5%
User Splitting	UBCF	1.160	1.158	1.158	1.214	2.794	2.780	2.780	2.944	1.026	0.987	0.999	1.052
	IBCF	1.174	1.185	1.170	1.327	2.882	2.874	2.859	2.981	0.985	0.967	0.985	1.039
	MF	1.004	1.007	*0.999*	1.119	2.561	*2.531*	2.547	2.594	0.934	*0.913*	0.928	1.011
	SP	88.7%	95.8%	85.8%	74.5%	36.2%	73.9%	37.7%	56.5%	35.4%	39.8%	35.4%	35.4%
	Sparsity	20.5%	19.3%	23.4%	14.1%	93.9%	94.0%	95.2%	94.7%	98.8%	98.8%	98.8%	98.8%
UI Splitting	UBCF	**1.092**	**1.088**	**1.088**	1.211	**2.702**	**2.686**	**2.689**	**2.810**	**1.012**	**0.956**	**0.989**	**1.042**
	IBCF	**1.112**	**1.141**	**1.124**	1.275	2.896	2.837	2.910	3.061	**0.972**	**0.946**	**0.972**	**1.020**
	MF	**0.967**	**0.975**	**0.971**	1.098	**2.534**	**2.489**	**2.508**	**2.520**	**0.927**	**0.892**	**0.915**	**0.998**
	Sparsity	62.0%	60.0%	61.7%	57.0%	94.9%	95.0%	96.4%	95.7%	99.2%	98.9%	98.9%	98.9%

v3.07 [6] to evaluate UBCF, IBCF (the neighborhood size in KNN is 30) and MF in our experiments. In order to better evaluate MF techniques, we tried different N factors ($5 \leq N \leq 60$, with 5 increment in each step) and training iteration T ($10 \leq T \leq 100$, with 10 increment in each step). Other parameters like learning and regularization factors are handled by MyMediaLite, where stochastic gradient descent is used as the optimization method.

For comparison purposes, we also implemented other popular contextual modeling approaches, common alternatives to contextual pre-filtering. We choose context-aware matrix factorization (CAMF), i.e. CAMF_C, CAMF_CI [3] and CAMF_CU [10], and another approach – differential context modeling (DCM) [15], i.e. DCR [13, 14] and DCW [16] [2] as comparative algorithms.

5. EXPERIMENTAL RESULTS

The results about two comparisons are discussed as follows.

5.1 Comparison Among Splitting Approaches

The experimental results evaluated by RMSE values are shown in Table 4. The numbers in bold are the best performing RMSE values for each recommendation algorithm across all three splitting approaches. The numbers in underlined in italic are the best RMSE values achieved for each data set using each splitting approach. And the numbers in underlined in bold are the global best RMSE for each data set. *SP* in the table denotes splitting percentage – equal to the percentage of items or users being split. Sparsity measures how sparse the rating matrix is after the splitting process; that is by the percentage of unknown ratings in a full *Users* $\times$ *Items* 2D rating matrix. Larger sparsity value indicates sparser rating data.

Overall, UI splitting outperforms other two splitting approaches in RMSE if it is configured optimally. More specifically, the best RMSE values are achieved by UI splitting using MF as the recommendation algorithm, where the best splitting criteria is the t_{chi} for the two movie data sets and t_{mean} for the food data set. From an overall view of the impurity criteria, t_{IG} performs the worst. Generally, MF is the best performing recommendation algorithm, because splitting increases sparsity and MF approaches usually handle sparsity better than KNN-based CF.

Compared with the best performed item splitting, the best performer, UI splitting, helps gain 6.7%, 1.0% and 11.8% improvements on RMSE. Note that in Table 4, we use the same impurity criteria to split items and users for UI splitting. In a follow-up experiment, we tried all possible combinations of the four impurity criteria and found that additional improvement is available if we use different criteria for items and for users, where we found one pattern – ***the best impurity criteria for UI splitting are the best ones for item splitting and user splitting individually***, e.g. in the Food data, the best splitting criteria is t_{mean} for item splits and t_{prop} for user splits in UI splitting, where t_{mean} is the best one for item splitting, and t_{prop} is the best one for user splitting when we use MF as recommendation algorithm in this data. Similar patterns were found for the other two data sets. Using this multi-criteria approach, the best RMSEs by UI splitting are 0.959, 2.477 and 0.892 for the Food, Movie and LDOS-CoMoDa data respectively.

User splitting is able to outperform item splitting for the food data and LDOS-CoMoDa – there are 3.6% and 9.7% improvements on RMSE compared with item splitting if we focus on the best performing ones. User splitting also splits more users than items for the two movie data sets in view of the SP values in the table. In terms of sparsity, there are no clear associations between the sparsity and RMSE performance. Except the Food data, sparsity is not increased too much for the two Movie data sets comparing UI splitting with others. However, UI splitting results in significant increment of sparsity for Food data, but it still outperforms the other two splitting approaches, this pattern is also confirmed by IR metrics discussed in the following sections.

We also explore the usage of contextual dimensions in those three splitting approaches, as the same in [17]. In short, we define a percentage of usage of context for splitting. By those percentages, we can discover what are the top selected contextual dimensions used in splitting approaches. Based on those analyses, the top two selected dimensions for the movie data are *companion* and *time*, and they are two emotional variables for the LDOS-CoMoDa. In Food data, they are the two contexts representing the degree of hungriness. In the LDOS-CoMoDa data, emotion is a personal quality and can be considered as more dependent with users other than items, where the degree of hungriness in the food data is a kind of subjective feelings from users. Those variables are more dependent with users than with items from common sense, but the *companion* and *time* in the Movie data are more neutral compared with users' subjective feelings or emotions. ***We conjecture that those usage patterns reveal the dependence between contexts and users/items, where user splitting may be preferred than the item splitting if contexts are more dependent with users rather than the items*** – that may be the underlying clue explaining the reason why user splitting works better that item splitting for the Food and LDOS-CoMoDa data sets, and this pattern is confirmed by the comparison between CAMF_CI and CAMF_CU discussed in the following.

Based on the RMSE results, UI splitting outperforms all other algorithms for those three data sets, even when item splitting or

[2] We apply DCR and DCW to user-based CF in this paper. See [16].

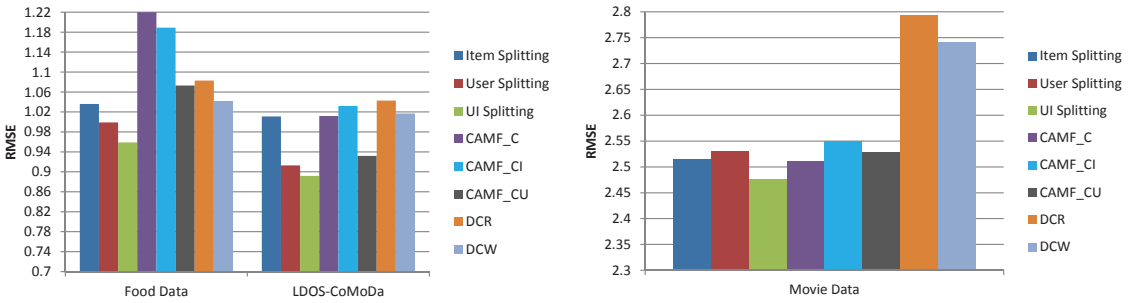

Figure 2: RMSE Comparison Among Different Context-aware Algorithms

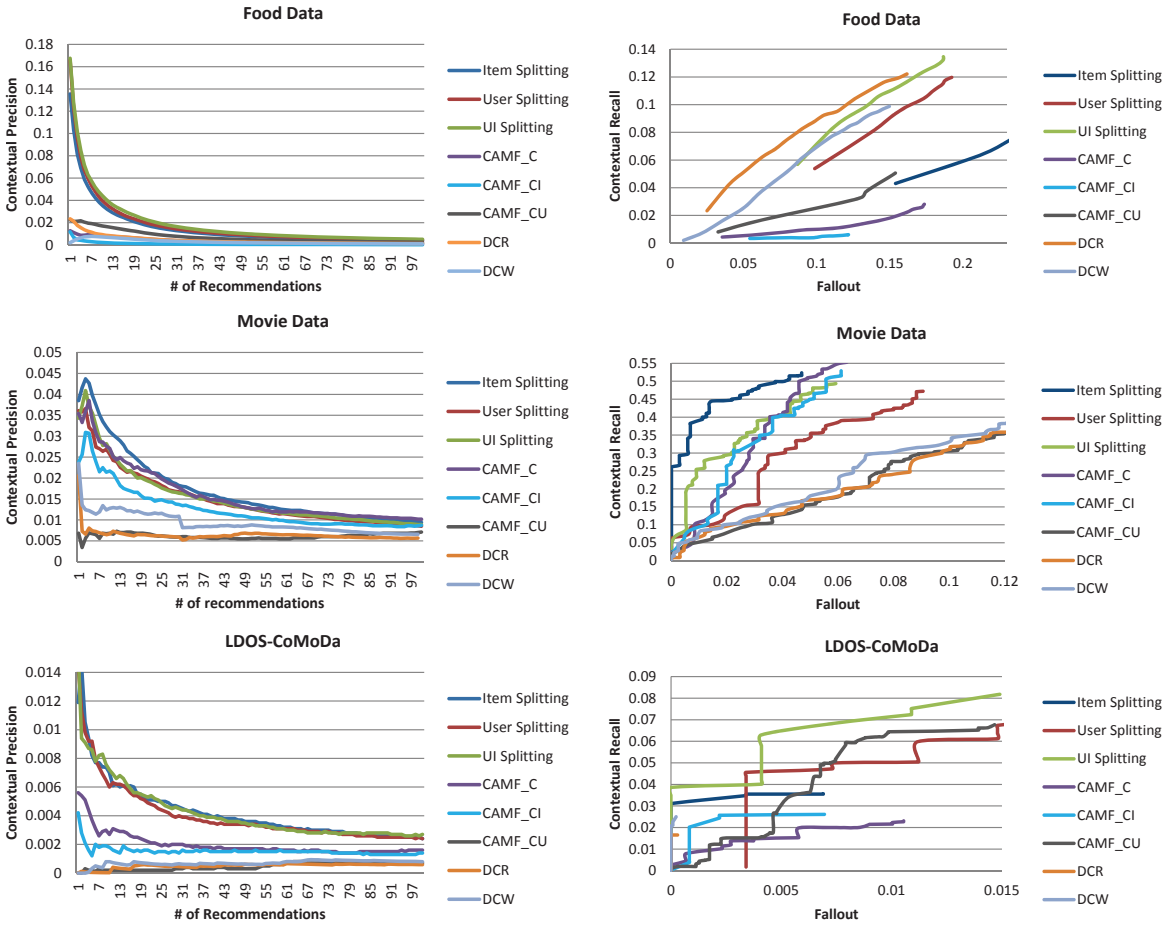

Figure 3: CPrecision and CROC curve for Each Data
(Left side one is CPrecision, and right side one is CROC curve)

user splitting alone cannot. These results confirm the effectiveness of UI Splitting approach for context-aware recommendations.

5.2 Comparison With Other Algorithms

In this section, we pick up the best performing CASA to compete with other CAMF algorithms by multiple evaluation metrics. Splitting approaches are the representative as pre-filtering approaches. CAMF and DCM are the contextual modeling approaches. We also tried the post-filtering approach described in [11], but its performance did not rival the competing algorithms and results are not included here [3]. Those results can be considered as contributions on comparisons among popular CACF algorithms.

Figure 2 compares the different CACF algorithms based on RMSE. ***In terms of RMSE, UI splitting works the best among those approaches, and generally, CASA outperform other two classes of algorithms.*** Notice that the RMSE differences among CASA, as well as the CAMF approaches for the Movie data are small and not that significant based on paired t-test. Also, CAMF_CU outperforms CAMF_CI for the food and LDOS-CoMoDa data sets, which confirms our previous conjecture – contexts are more depen-

[3]It is not efficient if the data set is small and with sparse contexts.

dent with users than items for those two data sets.

The experimental results based on CPrecision and CROC curve are represent in Figure 3. Due to that CPrecision only consider items which have ratings above the rating threshold as relevant ones, CROC curve is one important complementary metric which additionally measures the ratio between CRecall and fallout. ***Based on the comparison among the three splitting approaches, the winner varies from data to data*** – UI splitting performs the best for Food and LDOS-CoMoDa data sets in terms of CPrecision and CROC curve, though there is no clear winner between UI splitting and item splitting for the LDOS-CoMoDa data in CPrecision. This results are consistent with our findings in RMSE evaluations. However, item splitting outperforms the other two splitting approaches for the Movie data in both CPrecision and CROC curve, where UI splitting only works better than user splitting for this data, recall that the RMSE differences are not significant for the Movie set.

In terms of the empirical comparison among all those CACF algorithms, CASA generally outperform the other algorithms in terms of RMSE, CPrecision and CROC curve. DCM performs better only for the Food data, but is the worst for the other data in the IR metrics. The reason is that DCM is built on top of user-based collaborative filtering, which cannot handle sparsity and cold start problems very well compared with matrix factorization approaches. The food data is quite dense and this enables DCM to outperform some CAMF approaches. But DCM performs bad on CROC for LDOS-CoMoDa, because the rating in this data set is very sparse.

Basically, the CAMF_CU outperforms CAMF_CI, and user splitting outperforms item splitting for the Food and LDOS-CoMoDa data, except the CPrecison in the LDOS-CoMoDa data. This further confirms our conjecture about the association between data dimensions. If a contextual feature is more closely associated with the users in a dataset, such as emotions in the LDOS-CoMoDa data, we see better performance for user splitting.

6. CONCLUSION AND FUTURE WORK

In this paper, we empirically evaluated the three CASA, and also compared the recommendation performances between CASA with other CACF algorithms. Also, we propose the new contextual evaluation metrics for the comparison purposes.

In terms of the comparison among splitting approaches, we found that UI splitting outperform other two splitting approaches for the Food and LDOS-CoMoDa data over RMSE, CPrecision and CROC curve. The Movie data shows slightly different outcomes for the IR metric. In addition, the best splitting criteria is not also predictable and the best choice differs from data to data. We also found that user splitting may outperform item splitting if contexts are more dependent with users than items, which has been also confirmed by the performance comparison between CAMF_CU and CAMF_CI approaches. UI splitting will increase sparsity, but applying simple split to the splitting process will not hurt predictive performances very much. UI splitting is demonstrated to outperform the other two splitting methods in RMSE and IR metrics for two of the data sets we use. A further comparison with other popular context-aware recommendation algorithms reveal that splitting approaches generally perform the best for those three data sets. DCM may work better for denser data. In future work, we will explore complex splits for CASA and derive an optimization approach to assign best number of dimensions used for splitting approaches.

7. REFERENCES

[1] G. Adomavicius, B. Mobasher, F. Ricci, and A. Tuzhilin. Context-aware recommender systems. *AI Magazine*, 32(3):67–80, 2011.

[2] L. Baltrunas and X. Amatriain. Towards time-dependant recommendation based on implicit feedback. In *ACM RecSys, the 4th Workshop on Context-Aware Recommender Systems*, 2009.

[3] L. Baltrunas, B. Ludwig, and F. Ricci. Matrix factorization techniques for context aware recommendation. In *ACM conference on Recommender systems*, pages 301–304, 2011.

[4] L. Baltrunas and F. Ricci. Context-based splitting of item ratings in collaborative filtering. In *Proceedings of ACM conference on Recommender systems*, pages 245–248, 2009.

[5] L. Baltrunas and F. Ricci. Experimental evaluation of context-dependent collaborative filtering using item splitting. *User Modeling and User-Adapted Interaction*, pages 1–28, 2013.

[6] Z. Gantner, S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. Mymedialite: A free recommender system library. In *Proceedings of ACM conference on Recommender systems*, pages 305–308. ACM, 2011.

[7] N. Hariri, B. Mobasher, R. Burke, and Y. Zheng. Context-aware recommendation based on review mining. In *IJCAI, the Workshop on Intelligent Techniques for Web Personalization and Recommender Systems*, 2011.

[8] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53, 2004.

[9] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of ACM conference on Knowledge discovery and data mining (KDD)*, pages 426–434, 2008.

[10] A. Odic, M. Tkalcic, J. F. Tasic, and A. Košir. Relevant context in a movie recommender system: Users' opinion vs. statistical detection. In *ACM RecSys, the 4th Workshop on Context-Aware Recommender Systems*, 2012.

[11] U. Panniello, A. Tuzhilin, M. Gorgoglione, C. Palmisano, and A. Pedone. Experimental comparison of pre-vs. post-filtering approaches in context-aware recommender systems. In *Proceedings of ACM conference on Recommender Systems*, pages 265–268. ACM, 2009.

[12] A. Said, E. W. De Luca, and S. Albayrak. Inferring contextual user profiles – improving recommender performance. In *ACM RecSys, the 4th Workshop on Context-Aware Recommender Systems*, 2011.

[13] Y. Zheng, R. Burke, and B. Mobasher. Differential context relaxation for context-aware travel recommendation. In *13th International Conference on Electronic Commerce and Web Technologies*, pages 88–99, 2012.

[14] Y. Zheng, R. Burke, and B. Mobasher. Optimal feature selection for context-aware recommendation using differential relaxation. In *ACM RecSys, the 4th Workshop on Context-Aware Recommender Systems*, 2012.

[15] Y. Zheng, R. Burke, and B. Mobasher. Differential context modeling in collaborative filtering. In *School of Computing Research Symposium*. DePaul University, USA, 2013.

[16] Y. Zheng, R. Burke, and B. Mobasher. Recommendation with differential context weighting. In *The 21st Conference on User Modeling, Adaptation and Personalization*, pages 152–164, 2013.

[17] Y. Zheng, R. Burke, and B. Mobasher. The role of emotions in context-aware recommendation. In *ACM RecSys, the 3rd Workshop on Human Decision Making in Recommender Systems*. ACM, 2013.