The Role of Emotions in Context-aware Recommendation

Yong Zheng, Robin Burke, Bamshad Mobasher
Center for Web Intelligence
School of Computing, DePaul University
Chicago, Illinois, USA
{yzheng8, rburke, mobasher}@cs.depaul.edu

ABSTRACT

Context-aware recommender systems try to adapt to users' preferences across different contexts and have been proven to provide better predictive performance in a number of domains. Emotion is one of the most popular contextual variables, but few researchers have explored how emotions take effect in recommendations – especially the usage of the emotional variables other than the effectiveness alone. In this paper, we explore the role of emotions in context-aware recommendation algorithms. More specifically, we evaluate two types of popular context-aware recommendation algorithms – context-aware splitting approaches and differential context modeling. We examine predictive performance, and also explore the usage of emotions to discover how emotional features interact with those context-aware recommendation algorithms in the recommendation process.

Categories and Subject Descriptors

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

General Terms

Algorithm, Experiment, Performance

Keywords

Recommendation, Context, Context-aware recommendation, Emotion, Affective recommender system

1. INTRODUCTION

Advances in affective computing have enabled recommender systems to take advantage of emotions and personality, leading to the development of affective recommender systems (ARS) [18]. At the same time, the emerging technique of context-aware recommender systems (CARS) takes contexts into consideration, converting a two-dimensional matrix of ratings organized by user and item: *Users* $\times$ *Items* $\rightarrow$ *Ratings*, into a multidimensional rating space [1]. CARS have been demonstrated to be effective in a variety of applications and domains [4, 15, 10, 24, 13]. Emotional variables are often included as contexts in CARS, which enables the further development of both ARS and CARS. Typical emotional contexts in recommender system domain are the ones relevant to users' subject moods or feelings, for example, user mood when listening to music tracks (e.g. happy, sad, aggressive, relaxed, etc) [19, 9], or users' emotions after seeing a movie (e.g. user may feel sad after seeing a tragic movie) [13].

Emotional context was first exploited by Gonzalez *et al* [8] for the recommender system domain. This work was followed by others [9, 18, 13, 17] considering emotions as contexts in CARS research. While the effectiveness of emotions as contextual variables is therefore well-established, there is little research that has examined specifically the role that these emotional variables play. We define "the role of emotions" as the concerns from two aspects – whether emotions are useful or effective to improve recommendation performance? And, the usage of emotions in the recommendation process – how emotions are used in the recommendation algorithms, e.g. which emotional variables are selected? which algorithm components are they applied to? and so forth. Currently, most research are focused on demonstrating the effectiveness of emotional variables, and few research further explore the usage of the emotions in the recommendation process.

In this paper, we explore the role of emotions by two classes of popular context-aware recommendation algorithms – context-aware splitting and differential context modeling. Since both of these approaches require that the algorithm learn the importance and utility of different contextual features, they help reveal how emotions work in each algorithm and what roles they can play in recommendation.

The purpose of this study is therefore to address the following research questions:

1. *Emotional Effect*: Are emotions useful contextual variables for context-aware recommendation?
2. *Algorithm Comparison*: What algorithms are best suited to make use of emotional variables? Do they outperform the baseline algorithms?
3. *Usage of Contexts*: How do emotional variables compete with other contextual variables, such as location and time?
4. *Roles*: How can we understand the specific roles emotional variables can play in those context-aware recommendation algorithms?

2. RELATED WORK

Gonzalez *et al* [8] explored emotional context in recommender systems in 2007. They pointed out that,"*emotions are crucial for user's decision making in recommendation process. The users always transmit their decisions together with emotions.*" With the rapid development of context-aware recommender systems, emotion turns

RecSys'13, October 12–16, 2013, Hong Kong, China.
Paper presented at the 2013 Decisions@RecSys workshop in conjunction with the 7th ACM conference on Recommender Systems.

out to be one of important and popular contexts in different kinds of domains, especially in the music and movie domain. For music recommendation, the emotional context is appealing because it can be used to establish a bridge between music items and items from other different domains, and perform cross-media recommendations [6, 1, 9]. Movie recommendation is another domain where emotion turns out to be popular in recent years. In 2010, Yue *et al* [16] produced the overall winner in a recent challenge on context-aware movie recommendation by mining mood-specific movie similarity with matrix factorization. More recent research [18, 13] motivates the tendency of taking emotions as contexts to assist contextual recommendations. Research has demonstrated that emotions can be influential contextual variables in making recommendations, but few of them explore how emotions interact with recommendation algorithm – the usage of emotional variables in the recommendation process.

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 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. In this paper, we explore context-aware splitting approaches (a class of pre-filtering algorithms), and differential context modeling (a contextual modeling approach.)

The remainder of this paper is organized as follows. In Section 3 and Section 4, we formally introduce the two types of recommendation algorithms we are studying, including their capability to capture the role of emotional contexts in recommendation process. Sections 5 and 6 discuss the experimental evaluations and mining the role of emotions through those context-aware recommendation algorithms, followed by the conclusions and future work in Section 7.

3. CONTEXT-AWARE SPLITTING APPROACHES

Contextual pre-filtering is popular as it is straightforward to implement, and can be applied with most recommendation techniques. *Item splitting* [4, 5] is considered 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* [2, 15] 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. In this section, we introduce those two approaches and also propose a new splitting approach – *UI splitting*, a simple combination of item and user splitting.

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 data 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, for example a t-test or other statistical metric 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. We use only simple splitting in this work.

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	?

3.2 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.

3.3 UI Splitting

UI splitting is a new approach proposed in this paper – it 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.

Thus, given n items, m users, k contextual dimensions and d distinct conditions for each dimension, the time complexity for those three splitting approaches is the same as $O(nmkd)$ [5]. The process in UI splitting is simply an application of item splitting followed by user splitting on the resulting output. We keep the contextual information in the matrix after the transformation by item splitting so that user splitting can be performed afterwards.

3.4 Role of Emotions in Splitting

Context-aware splitting approaches have been demonstrated to improve predictive performance in previous research, but few of those research results report on the details of the splitting process. More specifically, it is useful to know which contextual dimensions and conditions are selected, and the statistics related to those selections. For our purposes, it is possible to explore how emotions interact with the splitting process by the usage of contexts in the splitting process, which may help discover the role of emotions from this perspective. In this paper, we try all three context-aware splitting approaches, empirically compare their predictive performance, and also explore the distributions of the usage of contexts (emotional variables included) for splitting operations.

4. DIFFERENTIAL CONTEXT MODELING

Differential context modeling (DCM) is a general contextual recommendation framework proposed in [23]. It can be applied to any recommendation algorithm. The "*differential*" part of the technique tries to break down a recommendation algorithm into different functional components to which contextual constraints can be applied. The contextual effect for each component is maximized, and the joint effects of all components contribute the best performance for the whole algorithm. The "*modeling*" part is focused on how to model the contextual constraints. There are two approaches: *context relaxation* and *context weighting*, where context relaxation uses an optimal subset of contextual dimensions, and context weighting assigns different weights to each contextual factor. Accordingly, we have two approaches in DCM: *differential context relaxation* (DCR) [21, 22, 20] and *differential context weighting* (DCW) [24].

4.1 DCM in UBCF

We have successfully applied DCM to user-based collaborative filtering (UBCF), item-based collaborative filtering (IBCF) and Slope-One recommendation algorithms in our previous research [23], showing that DCM is an efficient approach to improve context-aware prediction accuracy. However, we did not explore much about the modeling part in our previous work; that is, how contexts are selected, relaxed or weighted.

Recall that we separate different functional components from the recommendation algorithm, therefore, how the algorithms relax or weigh contextual variables can tell the role of the contexts in different components. In this paper, we continue to apply DCM to UBCF, where the four components we separate in UBCF can be described as follows:

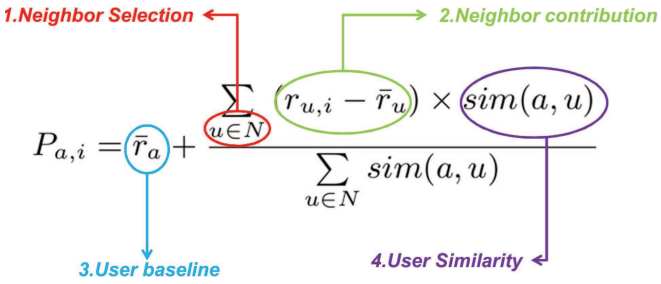

Figure 1: Algorithm Components in UBCF

Figure 1 shows the original rating prediction by the well-known Resnick's UBCF algorithm [14], where a is a user, i is an item, and N is a neighborhood of K users similar to a. The algorithm calculates $P_{a,i}$, which is the predicted rating that user a is expected to assign to item i. Then we decompose it to four components:

Neighborhood selection UBCF applies the well-known k nearest-neighbor (kNN) approach, where we can select the top-k neighbors from users who have rated on the same item i. If contexts are taken into consideration, the neighborhood can be further restricted so that users have also rated the item i in the *same contexts*. This gives a context-specific recommendation computation. However, the strict application of such a filter greatly increases the sparsity associated with user comparisons. There may only be a small number of cases in which recommendations can be made. DCM offers two potential solutions to this problem. DCR searches for the optimal relaxation of the context, generalizing the set of contextual features and contextual conditions to reduce sparsity. DCW uses weighting to increase the influence of neighbors in similar contexts.

Neighbor contribution The neighbor contribution is the difference between a neighbor's rating on an item i and his or her average rating over all items. Context relaxation and context weighting can be applied to the computation of $\bar{r}_u$. This computation is replaced by one in which this average is computed over ratings from a relaxed set of contexts in DCR, or the average rating can be aggregated by a weighted average across similar contexts under DCW.

User baseline The computation of $\bar{r}_a$ is similar to the neighbor's average rating and can be made context-dependent in the same way.

User similarity The computation of neighbor similarity $sim(a,u)$ involves identifying ratings $r_{u,i}$ and $r_{a,i}$ where the users have rated items in common. For context-aware recommendation, we can additionally add contextual constraints to this part: use ratings given in matching contexts with context relaxation or ratings weighted by contextual similarity using context weighting.

With these considerations in mind, we can derive a new rating prediction formula by applying DCR or DCW to UBCF. More details about the prediction equations and technical specifications can be found in [24].

4.2 The Role of Emotions in DCM

One advantage of DCM is that it allows us to explore the role of contexts in each algorithm component. DCM seeks to optimize the contribution of context in each component, and so, the output of the optimization phase, in which contextual dimensions are selected and/or weighted, can reveal the relative importance of different contextual features in different algorithm components. In this paper, we provide comparisons and also explore the role of emotions in these two classes of context-aware recommendation algorithms.

5. EXPERIMENTAL SETUP

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

5.1 Data Sets

We examine context-aware splitting approaches and DCM with the real-world data set *LDOS-CoMoDa*, which is a movie dataset collected from surveys and used by Odic *et al* [12, 13, 17]. After filtering out subjects with less than 5 ratings and rating records with incomplete feature information, we got the final data sets containing 113 users, 1186 items, 2094 ratings, 12 contextual dimensions and the rating scale is 1 to 5. We created five folds for cross validation purposes and all algorithms are evaluated on the same five folds. Specific descriptions of all the contextual dimensions and conditions can be found in previous work [12] and the description of the data set online [1].

In our experiments, we use all 12 contextual dimensions, including three emotional contexts: *Mood*, *DominantEmo* and *EndEmo*, and non-emotional contexts, including *Location*, *time*, etc. The goal is to compare emotional contexts with others. For example, in DCR, only influential contextual variables for a specific component are selected – whether emotional contexts are selected or not can indicate the significance of their impacts for this component. For a comprehensive comparison, we also performed experiments without emotional contexts to better evaluate the importance of the emotional dimensions.

In this data, there are the three contextual dimensions that contain emotional information. "EndEmo" is the emotional state experienced at the end of the movie. "DominantEmo" is the emotional state experienced the most during watching, i.e. what emotion was induced most times during watching. "Mood" is what mood the user was in during that part of the day when the user watched the movie. Mood has lower maximum frequency than emotions, it changes slowly, so we assumed that it does not change during watching. "EndEmo" and "DominantEmo" contain the same seven conditions: *Sad, Happy, Scared, Surprised, Angry, Disgusted, Neutral*, where "Mood" only has simple three conditions: *Positive, Neutral, Negative*.

5.2 Configurations

To evaluate the performance of context-aware splitting approaches, we used 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 a 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 as 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 are 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. A finer-grained operation is to set another threshold for each impurity value and each data set. 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 we choose the top first (highest impurity) as the best split. Items or users without qualified splitting criteria are left unchanged.

In DCM, we used the same configuration in our previous work [24]: we select the top-10 neighbors in UBCF, choose 100 as the maximal iteration in the optimization process, and Pearson Correlation is used as the user similarity measure. See our previous work for more details.

5.3 Evaluation Protocols

For the evaluation of predictive performance, we choose the root mean square error (RMSE) evaluated using the 5-fold cross validation. The data set is relatively small and some subjects in the survey were required to rate specific movies, thus precision and recall may be not a good metric, but we plan to evaluate them and other metrics in our future work.

We applied DCM only to user-based collaborative filtering. For splitting approaches, there are more options – we evaluate the performance of three recommendation algorithms: user-based collaborative filtering (UBCF), item-based collaborative filtering (IBCF) and matrix factorization (MF) for each splitting approach. Koren [11] introduced three MF techniques: MF with rating bias (BiasMF), Asymmetric SVD (AsySVD) and SVD++; we found that BiasMF was the best choice for this data.

We used the open source recommendation engine MyMediaLite v3.07 [7] to evaluate UBCF, IBCF and BiasMF in our experiments. (We choose K=30 for KNN-based UBCF and IBCF.) In order to better evaluate MF techniques, we tried a range of different factors ($5 \leq N \leq 60$, increment 5) and training iteration T ($10 \leq T \leq 100$, increment 10). 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 choose the well-known context-aware matrix factorization algorithm (CAMF) [3], i.e. CAMF_C, CAMF_CI and CAMF_CU [2] as the baseline. We tried our best to fine-tune the configurations (e.g. learning parameters, training iterations, etc) for CAMF in order to make a reasonable and comprehensive comparison.

6. EXPERIMENTAL RESULTS

In this section, the comparisons of predictive performance are introduced first, followed by the discussion of emotional roles discovered in our experiments.

6.1 Prediction

We use three treatments of the context information – one is the data with *All Contexts* where both emotional contexts and non-emotional variables are included, and the second one is the data omitting the emotional context dimensions marked by *No Emotions* in the table below. The third one is the data with *Emotions Only* emitting all non-emotional variables. The overall experimental results are shown in Table 3, where the numbers in underlined in italic are the best RMSEs by each approach (i.e. the best one in

[1] http://212.235.187.145/spletnastran/raziskave/um/comoda/comoda.php

[2] CAMF_CU is a CAMF approach which utilizes the interaction between contexts and users. [12]

Table 3: Overall Comparison of RMSE
(The results for splitting approaches are based on BiasMF.)

	Algorithms	**All Contexts**	**No Emotions**	**Emotions Only**
CAMF [3]	CAMF_C	1.012	1.066	*0.968*
	CAMF_CI	1.032	1.083	*1.019*
	CAMF_CU	0.932	1.021	*0.902*
DCM [24]	DCR	*1.043*	1.057	1.046
	DCW	*1.017*	1.037	1.036
Splitting Approaches	Item Splitting	*1.011*	1.014	1.014
	User Splitting	*0.913*	0.971	0.932
	UI Splitting	**0.892**	**0.942**	**0.903**

each row), and the bold numbers are the best RMSEs by each data forms (i.e. the best one in each column).

The comparison of performances of those approaches over the three forms of data can be visualized by Figure 2. We see that including emotions as contexts can improve RMSE compared with the situation we only use non-emotional contexts (i.e. *No Emotions*). The results differ when we switch our attention to the "*Emotions Only*" one. In CAMF, it helps achieve the lowest RMSE with *Emotions Only* data. But, including all of the contextual information yielded the best RMSE for the other approaches: DCM and context-aware splitting algorithms.

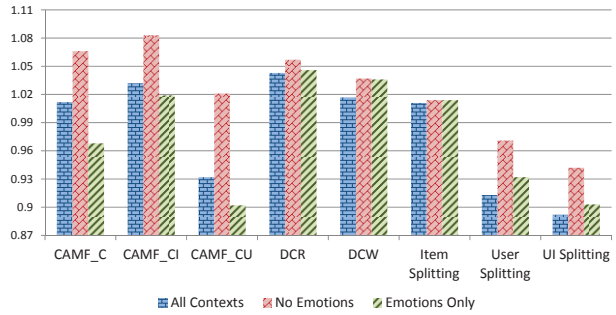

Figure 2: Comparison of RMSE Over Three Contextual Situations

This result is not surprising because both DCM and splitting approaches choose among contextual features, deciding which to apply in recommendation. CAMF, on the other hand, uses all available context information in performing its factorization. Without the benefit of feature selection, adding additional features to CAMF may increase noise.

From an overall view, UI splitting has the best RMSE across all data treatments among those context-aware recommendation algorithms. Table 4 shows more details of the predictive performance among the three splitting approaches if we include emotions as the contexts (i.e. using all contexts). The numbers are shown as RMSE values, where 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 the data set using each splitting approach. The numbers in underlined in bold are the global best RMSE for the data set.

These tables show that adding emotions to contexts is able to provide improvement in terms of RMSE, thus answering research question 1 in the affirmative. It also suggests an answer to question 2: that the UI splitting approach outperforms other two splitting approaches if it is configured optimally. In particular, the best RMSE values are achieved by UI splitting using MF as the recommendation algorithm with t_{chi} as the splitting criteria.

Table 4: Comparison of Predictive Performance (RMSE) Among Splitting Approaches

		LDOS-CoMoDa			
	Algorithms	t_{mean}	t_{chi}	t_{prop}	t_{IG}
Item Splitting	UBCF	1.040	1.021	1.028	1.043
	IBCF	1.030	1.024	1.026	1.034
	BiasMF	1.020	*1.011*	1.016	1.020
User Splitting	UBCF	1.026	0.987	0.999	1.052
	IBCF	0.985	0.967	0.985	1.039
	BiasMF	0.934	*0.913*	0.928	1.011
UI Splitting	UBCF	**1.012**	**0.956**	**0.989**	**1.042**
	IBCF	**0.972**	**0.946**	**0.972**	**1.020**
	BiasMF	**0.927**	**0.892**	**0.915**	**0.998**

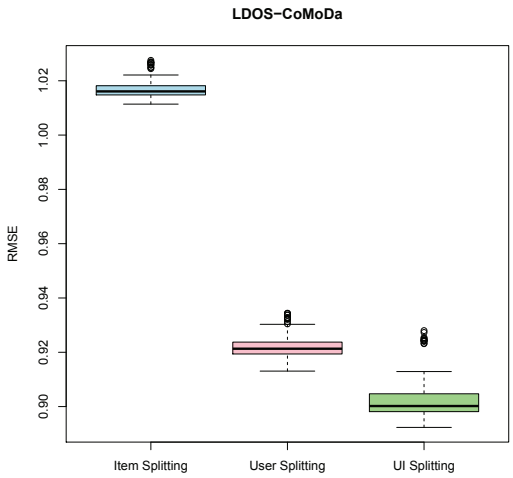

Figure 3: Boxplot of RMSE Among Splitting Approaches

Generally, MF is the best performing recommendation algorithm. This is not surprising because splitting increases sparsity and MF approaches are designed to handle sparsity data. Because the difference in RMSE is small, we show the boxplot of RMSEs among the best performing item splitting, user splitting and UI splitting approaches (i.e. the configuration as the underlined values in Table 4) in Figure 3. The data in the figure are the 120 RMSE values which comes from the training iterations – different factors

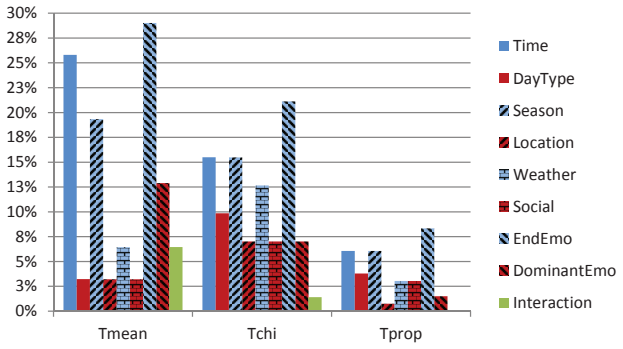

Figure 4: Item Splitting

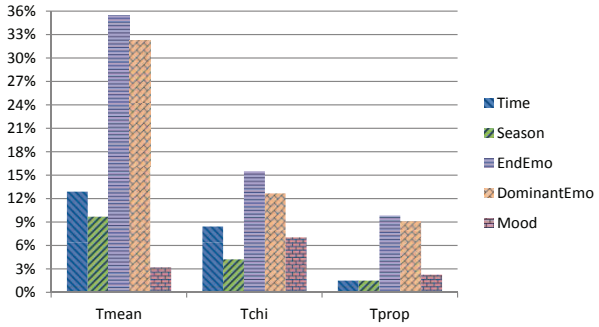

Figure 5: User Splitting

Table 5: Context Relaxation and Weighting by DCM

Algorithm Components	Context Relaxation By DCR	Context Weighting By DCW
Neighbor Selection	N/A	Day, *Mood*
Neighbor Contribution	Movie Year, Genre	Movie Genre
User Baseline	*DominantEmo, EndEmo*, Movie Language	*DominantEmo, EndEmo*, Interaction
User Similarity	*EndEmo*, Location	*DominantEmo*

($5 \leq N \leq 60$, with 5 increment in each step) and training iteration T ($10 \leq T \leq 100$, with 10 increment in each step). The figure confirms the comparative effectiveness of UI splitting – the box is significantly lower than the other two approaches with the same training iterations.

6.2 The Role of Emotions

As mentioned before, one reason we sought to explore the usage of emotional contexts to discover how emotions interact within context-aware recommendation algorithms. Both splitting and DCM offer insights into how contextual dimensions and features are influential for recommendation.

6.2.1 *Context-aware Splitting*

In splitting approaches, the splitting statistics help discover how contexts were applied in our experiments. In Figure 4 and 5, we show the top selected contextual dimensions for item splitting and user splitting[3]. The right legend indicates the contextual dimensions, and the y axis denotes the percentage of splits (item splits or user splits) using each dimension. For a clearer representation in the figures, we only show contextual dimensions used on more than 5% (item splitting) or 6% (user splitting) of the recommendations. We do not show results of t_{IG} because this splitting criterion had the worst performance.

In general, the top two dimensions are consistent across those three impurity criteria: *EndEmo* and *Time* for item splitting and *EndEmo* and *DominantEmo* for user splitting. However, the percentages as y axis in the figures are different, not to mention that the selected condition in each dimension differs too, which results in different performance by using various impurity criteria. In terms of the specific selected emotional conditions, the results are not consistent – the top context dimension for item and user splitting is the same – *EndEmo*, but the most frequent selected condition in this dimension is "Happy" for item splitting and "Neutral" for user splitting.

[3]The actual splitting is based on a specific contextual condition in a dimension, but the results of selected contextual conditions are fuzzy, thus the distribution of selected contextual dimensions is explored and reported here.

EndEmo denotes the emotion of the users after seeing the movie, and this result is consistent with previous work [12] on this data. Obviously, emotion is a personal quality and can be considered as more dependent with users other than items – we conjecture that this may be the underlying clue explaining why user splitting works better than item splitting for the LDOS-CoMoDa data. And in user splitting, the top two selected contextual dimensions are the two emotional variables: *EndEmo* and *DominantEmo* – emotions are generated and owned by users, therefore they are more dependent with users other than items. This pattern is confirmed by the comparison between two CAMF approaches: CAMF_CI and CAMF_CU – contexts are more dependent with users than with items, which results in better performances by CAMF_CU.

In short, the statistics based on splitting approaches reveal the importance of emotions – at least the top first selected contextual dimension is the "EndEmo" for both item splitting and user splitting.

6.2.2 *Differential Context Modeling*

In DCM, the context selection and context weighting can be examined and show, in a detailed way, the contribution of each contextual dimension in the final optimized algorithm. The results are shown in Table 5, where the four components in UBCF were described in the previous section. "N/A" in the table indicates no contextual constrains were placed. For clearer representation, we did not show specific weights of contexts in DCW; instead, we only list variables which were assigned weights above a threshold of 0.7. The weights are normalized to 1, and 0.7 therefore represents a very influential dimension.

We can see in the table that emotional variables are selected in DCR and weighted significantly in DCW and for which components. Emotion is influential for a specific component but may not for other ones. In DCR for example, *EndEmo* turns out to be influential when measuring user similarities and user baselines, but it is not that significant in computing the neighbor contribution.

It is not surprising that the results from DCW are not fully consistent with ones from DCR. DCW is a finer-grained approach. Emotional variables are assigned to neighbor selection in DCW but not for the same component in DCR, and the specific selected emotions

Table 6: Comparison of Context-aware Splitting and Differential Context Weighting

Answers	Context-aware Splitting Approaches	Differential Context Weighting
Q1. Emotional Effect	Yes. RMSE is improved with emotional context dimensions.	Yes. RMSE is improved with emotional context dimensions.
Q2. Algorithm Comparison	UI splitting is the best and outperforms DCM and CAMF approaches.	DCW is better than DCR and also works better than some CAMF approaches in specific contextual situations.
Q3. Usage of Contexts	Emotional dimension is the top selected dimension for context splitting.	Emotional dimensions get more weight than other contextual variables in most algorithm components.
Q4. Roles	*EndEmo* is used as the top first selected context for both user and item splitting. *DominantEmo* is the top second one in user splitting.	Emotions are significantly influential for some algorithm components, e.g. *EndEmo* and *DominantEmo* for user similarities and baselines in DCW.

are different too, e.g. it is *DominantEmo* selected in DCW for user similarity calculation, but it is the *EndEmo* in DCR. In DCW, the weights for *EndEmo* and *DominantEmo* are close to 1 (the weights are all above 0.92) in the component of user baseline, which implies significant emotional influence on this component. Emotional contexts are important in DCM, where it can be further confirmed by Table 3 – the RMSE values are increased if we remove emotions from the contexts.

The result by DCM is useful for further applications, such as affective computing or marketing purposes. Take the results of DCW in Table 5 for example, *Mood* is influential for selecting neighbors, which implies that if two users rated the same item under the same mood situation, it is highly possible that they are the good neighbor candidates for each other (though neighbor selection also depends on the user similarities). Similarly, *DominantEmo* is useful to measure user similarities, which infers that users rated items similarly with the same dominant emotions are probably the good neighbors in user-based collaborative filtering.

7. CONCLUSION AND FUTURE WORK

In conclusion, both context-aware splitting approaches and DCM are able to reveal how emotions interact with algorithms to improve recommendation performance by exploring the usage of contexts in the recommendation or splitting process. More specifically, in context-aware splitting approaches, the percentage of emotional contexts used by item splits or user splits can tell the importance of emotions in distinguishing different user rating behavior. DCM provides a way to see very specifically which emotional contexts are influential for which components in the recommendation process.

Table 6 examines how our research questions are answered for each type of context-aware recommendation. As discussed above, we find that contextual dimensions keyed to emotions are very useful in recommendation (Question 1) and that UI splitting outperforms the DCM and CAMF approaches (Question 2).

Question 3 asks about how emotional dimensions compare with other contextual information. Our results show that emotion-linked context makes an important contribution to context-aware recommendation, although other dimensions are also certainly important. For example, we see that *EndEmo* is the top selected contextual dimension in item and user splitting, with *Time* running the second in item splitting, and *DominantEmo* is the top second selected contextual dimension in user splitting.

Our fourth research question asks about how those two classes of context-aware recommendation algorithms can tell the roles of emotions by usage of contexts. Context-aware approaches are able to infer the roles by the distribution of usage of emotions selected for the splitting process. DCM techniques help answer this question, by pointing out which components make good use of different contextual variables – showing here that *DominantEmo* is important for calculating the user's baseline. Note that the neighbor contribution component in DCW is alone in making significant use of non-emotional context dimensions. This is interesting because in this component we are determining which movies to use to compute the neighbor's baseline for prediction. It appears that the system prefers to use non-emotional considerations in making use of others' ratings, even though the user's baseline is computed based on emotional dimensions.

In addition, we can get extra information from our splitting experiments because splitting is performed based on specific contextual conditions. As described above, we found in particular that "Happy" vs not-"Happy" was the important split on the *EndEmo* dimension for item splitting. Essentially, the algorithm is indicating that there is an important difference between users who feel happy at the conclusion of a given movie and those that do not.

In our future work, we plan to continue our exploration of these algorithms and more data using additional metrics, such as recall and/or normalized discounted cumulative gain. We are also looking at additional data sets to see if similar effects are found with respect to emotions, as well as the empirical comparison among those context-aware recommendation algorithms. We also plan to examine the effects by the correlations between different contextual variables, e.g. how emotional effects change if emotions are significantly dependent with other contexts or features. In addition, it is interesting to explore the association among emotions, user profiles, item features and users' ratings. For example, user may feel "*sad*" after seeing a tragedy movie, but the emotion could be "*happy*" because he or she saw such a good movie even if it is a tragedy. Therefore, which specific emotions will result in a higher rating? the "*sad*" or the "*happy*"?

8. 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' 09, Proceedings of the 4th International Workshop on Context-Aware Recommender Systems (CARS 2009)*, 2009.

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

[4] L. Baltrunas and F. Ricci. Context-based splitting of item ratings in collaborative filtering. In *Proceedings of the third*

ACM conference on Recommender systems, pages 245–248. ACM, 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] S. Berkovsky, T. Kuflik, and F. Ricci. Cross-domain mediation in collaborative filtering. In *User Modeling 2007*, pages 355–359. Springer, 2007.

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

[8] G. Gonzalez, J. L. de la Rosa, M. Montaner, and S. Delfin. Embedding emotional context in recommender systems. In *Data Engineering Workshop, 2007 IEEE 23rd International Conference on*, pages 845–852. IEEE, 2007.

[9] B.-j. Han, S. Rho, S. Jun, and E. Hwang. Music emotion classification and context-based music recommendation. *Multimedia Tools and Applications*, 47(3):433–460, 2010.

[10] N. Hariri, B. Mobasher, R. Burke, and Y. Zheng. Context-aware recommendation based on review mining. In *IJCAI' 11, Proceedings of the 9th Workshop on Intelligent Techniques for Web Personalization and Recommender Systems (ITWP 2011)*, pages 30–36, 2011.

[11] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434. ACM, 2008.

[12] 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' 12, Proceedings of the 4th International Workshop on Context-Aware Recommender Systems (CARS 2012)*, 2012.

[13] A. Odić, M. Tkalčič, J. F. Tasič, and A. Košir. Predicting and detecting the relevant contextual information in a movie-recommender system. *Interacting with Computers*, 25(1):74–90, 2013.

[14] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186. ACM, 1994.

[15] A. Said, E. W. De Luca, and S. Albayrak. Inferring contextual user profiles – improving recommender performance. In *ACM RecSys' 11, Proceedings of the 4th International Workshop on Context-Aware Recommender Systems (CARS 2011)*, 2011.

[16] Y. Shi, M. Larson, and A. Hanjalic. Mining mood-specific movie similarity with matrix factorization for context-aware recommendation. In *Proceedings of the Workshop on Context-Aware Movie Recommendation*, pages 34–40. ACM, 2010.

[17] M. Tkalčič, U. Burnik, A. Odić, A. Košir, and J. Tasič. Emotion-aware recommender systems–a framework and a case study. In *ICT Innovations 2012*, pages 141–150. Springer, 2013.

[18] M. Tkalcic, A. Kosir, and J. Tasic. Affective recommender systems: the role of emotions in recommender systems. In *ACM RecSys Workshop on Human Decision Making*, 2011.

[19] A. L. Uitdenbogerd and R. G. van Schyndel. A review of factors affecting music recommender success. In *ISMIR*, volume 2, pages 204–208, 2002.

[20] Y. Zheng, R. Burke, and B. Mobasher. Assist your decision-making in various situations: Differential context relaxation for context-aware recommendations. In *Research Colloquium, School of Computing, DePaul University, Chicago IL, USA*, 2012.

[21] 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 (EC-WEB 2012)*, pages 88–99, 2012.

[22] Y. Zheng, R. Burke, and B. Mobasher. Optimal feature selection for context-aware recommendation using differential relaxation. In *ACM RecSys' 12, Proceedings of the 4th International Workshop on Context-Aware Recommender Systems (CARS 2012)*. ACM, 2012.

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

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