# Context Recommendation Using Multi-Label Classification

Yong Zheng
Center for Web Intelligence
DePaul University
Chicago, Illinois, 60604, USA
yzheng8@cs.depaul.edu

Bamshad Mobasher
Center for Web Intelligence
DePaul University
Chicago, Illinois, 60604, USA
mobasher@cs.depaul.edu

Robin Burke
Center for Web Intelligence
DePaul University
Chicago, Illinois, 60604, USA
rburke@cs.depaul.edu

*Abstract*—**Context-aware recommender systems (CARS) are extensions of traditional recommenders that also take into account contextual condition of a user to whom a recommendation is made. The recommendation problem is, however, still focused on recommending a set of items to a target user. In this paper, we consider the problem of recommending to a user the appropriate contexts in which an item should be selected. We believe that context recommenders can be used as another set of tools to assist users' decision making. We formulate the context recommendation problem and discuss the motivation behind and possible applications of the concept. We identify two general classes of algorithms to solve this problem: direct context prediction and indirect context recommendation. Furthermore, we present and evaluate several direct context prediction algorithms based on multi-label classification (MLC). Our experiments demonstrate that the proposed approaches outperform the baseline methods, and also that personalization is required to enhance the effectiveness of context recommenders.**

## I. Introduction

Recommender systems (RS) have been used to successfully address the information overload problem by providing personalized and targeted recommendations to the end users. There are two principal forms of recommendations – user recommendation (e.g. social recommender in Facebook and Twitter) and item recommendation (e.g. product recommendations on Amazon and eBay, or movie recommendation by Netflix).

In recent years, context-aware recommender systems (CARS) have emerged that are able to adapt recommendations to users' contextual situations. Traditional recommendation problem can be modeled as a two-dimensional (2D) prediction – $R$: *Users* $\times$ *Items* $\rightarrow$ *Ratings*, while CARS try to additionally incorporate contexts to estimate user preferences using a "*multidimensional*" rating function – $R$: *Users* $\times$ *Items* $\times$ *Contexts* $\rightarrow$ *Ratings* [1].

Several CARS algorithms have been developed to assist item recommendations in contexts, such as differential context modeling [18] (i.e. DCR [16], [17] and DCW [19]), context-aware splitting approaches [7], [22], context-aware matrix factorization [6], and so on. The research in CARS has demonstrated that context can play an important role in making recommender systems more effective – users' preferences or decisions may vary from context to context, even for the same item. Thus, incorporating context into the recommendation process may provide more pertinent results for users.

However, as in traditional recommender systems, the goal of a context-aware recommender is also to recommend an item (or a user in the case of social recommendation) to a target user. There are many scenarios where a user's final decision on whether to select an item may also depend on contexts in which the item may be used or consumed. Therefore, the identification of appropriate contextual situations for the user with respect to a given item may also provide critical information to assist that user in the selection decision. For example, a popular vacation destination may not always be appropriate for a particular traveler – factors such as weather conditions in a given season, proximity to other relevant locations, and even presence of companions or family members, are important contextual factors that can affect the user's selection decision. Also, a highly-rated movie may not always provide the best entrainment experience for a movie goer, unless it is seen in the proper facility, or with the right set of companions.

This observation leads to a new paradigm in recommender systems. We propose the notion of *context recommendation* which we define as the task of suggesting the most appropriate contexts in which a target user should select a specific item. The recommended contexts may be represented as a combination of contextual conditions that affect user's utility or preferences on the target item.

Our primary contributions in this paper can be summarized as follows.

- We introduce the notion of context recommendation and discuss novel applications of this type of recommender systems.
- We present an algorithmic framework for context recommendation and discuss different methodologies for learning and context prediction.
- We explore and empirically evaluate several specific algorithms for context recommendation based on multi-label classification (MLC) and demonstrate they outperform the baseline algorithms in our experiments.

## II. Motivations and Background

The idea of *recommending context* was first mentioned in the tutorial [3] on context-aware recommender systems by Adomavicius and Tuzhilin in 2008 where it was pointed out that the development of context-aware recommender systems could bring additional recommendation opportunities, such as recommending contexts to users. For example, one could envision recommending the best season or times for one user to go on a vacation to Hawaii, USA. However, this type of recommendation has not been explored formally or empirically in recommender systems research.

There are several underlying motivations for exploiting the context recommendation. First, users need context recommendations which suggest the most appropriate setting in which to consume the item. For example, a user may have decided to see a specific movie, but he may need suggestions about the location (e.g. at home or theatre) and companions (e.g. with family or with a partner).

In addition, context recommendation may be important to users in making item selection decisions. There are many scenarios in which users may not know how to select from a set of possible items (possibly generated by another recommender system based on recorded preferences), but information about which item is appropriate in which context can make the difference in the final selection. For example, a user may have several movie choices for a weekend activity all of which may be of interest to the user based on general preferences. However, the suggested contexts for each movie could result in different decisions. For example, the context recommender may suggest that a given movie is more appropriate to be seen with the whole family than with a partner on a romantic date. Another movie might provide the best experience if seen on a large screen at the theatre rather than at home on television.

The idea of *context prediction* has been studied in the domain of pervasive computing. However, there are distinct differences between the problem of context prediction in recommender systems domain and in pervasive computing. Context prediction in pervasive computing is usually defined as the task of forecasting the *upcoming context information* to further assist proactive adaptations. For example, smart home applications are required to predict the user's next location in advance in order to adjust lights and temperature before the user enters a room, where there are different sensors working together to acquire real-time contexts and try to predict the next one based on a sequential context history. Thus, the predicted contexts are relatively correlated with sequential patterns and generally involve a very small number of specific conditions. In particular, location is typically the only contextual factor to be predicted in pervasive computing. In contrast, there are often several context variables involved in a typical recommender system application, and sequential patterns may only limited applicability recommender systems. Classification and prediction algorithms, such as Bayesian networks, neural networks and Markov models, are the most common approaches for context prediction in pervasive computing.

In the domain of recommender systems, context prediction has been studied by Baltrunas *et al.* [4], where the authors tried to predict the best contexts for users to listen to various music tracks in an album or a playlist. They proposed several classifiers based on the K-nearest-neighbor strategy to solve the context prediction problem. This work provides a specific application of context recommendation, but research in this area has not yet provided a clear and general delineation of the problem of context recommendation.

## III. Framework of Context Recommendation

In this section, we present a framework of context recommendation which includes different algorithmic paradigms for this type of recommendation problem. We also discuss evaluation challenges in the context recommendation task.

### A. The Context Recommendation Problem

We use the term *contextual dimension* to denote the contextual variable or factor (for example, "Time" and "Location" may be contextual dimensions). The term *contextual condition* refers to a specific value in a contextual dimension (for example "Weekend" and "Weekday", or "Summer" and "Winter" may be contextual conditions for "Time" dimension).

The general form of context recommendation can be represented as mapping of user-item pairs to a set of contexts. We denote this mapping by $\{\text{User, Item}\} \rightarrow \{\text{Contexts}\}$ meaning that user(s) and item(s) are inputs and a list of contexts as the outputs. For example, a context recommendation problem might be to identify the best contexts for Tom to see the movie Titanic? Example context suggestions may be *on a weekday night with his wife* or *at home with kids on the weekend*? Tom's user profile (e.g. gender), preferred (e.g. genre), and previous movie ratings, can further be used to provide a personalized ranking of the relevant contexts. Thus, the recommended contexts can be listed along with each item and they can be personalized for each user. For example, for a given movie, the recommended companions or locations may be different for two different users. Whether personalization is required or not is a question worth exploring in this line of research.

It is possible to extend this basic framework to incorporate additional types of inputs. For example, the inputs could include a group of users or items instead of a single user-item pair. This might provide new recommendation opportunities in group recommendation, where the recommended contexts must satisfy the needs of a group of users. For example, a user or a group of users may prefer a particular set of songs all of which are appropriate for a suggested occasion (e.g. at a party, driving, exercising, etc.). Similarly, a group of travelers may be seeking the best times and venues for a group tour in Australia, where the suggested contextual conditions should meet the requirements of every individual traveler in the group.

Other variants of the general context recommendation problem can also be envisioned. For example, the inputs could only include users ignoring the item component. For example, one might be interested in finding the best time to shop for a *gift for Mom*? without any specification on the item to purchase. Similarly, item(s) could be the only input. One can imagine that an e-commerce web site posting their products along with a list of the most relevant or appropriate contexts in which the item may be used (even without knowing anything about the current user).

In this paper, we mainly focus on the form of the context recommendation problem where the input is a user-item pair. We also consider a situation where content features are available as part of user profiles which are demonstrated to be able to improve the performance in the context recommendation task.

### B. Algorithmic Paradigms

In this section, we introduce two algorithmic paradigms, each of which can be used as the basis for a class of context recommendation algorithms.

*1)* ***Direct Context Prediction****:* Direct context prediction is modeled as: *User* $\times$ *Item* $\times$ *Rating* $\rightarrow$ *Contexts*. In this case, the suggested context labels (corresponding to specific combinations of contextual conditions) are directly predicted based on the rating profile tuples $\{$user, item, rating$\}$ using classification techniques. We assume that the predicted context labels are the contexts that will be recommended to the user. In addition, contexts must be filtered in two ways:

a) Pre-filtering Predictions. We can pre-select contextual dimensions in advance, and then provide predictions related to those contexts. In other words, the selected contexts are what we want to recommend and they are considered as "*a bag of words*" – it is not necessary to further evaluate the ranking of those contexts, because the pre-selected context are already considered to be relevant.

b) Post-filtering Predictions. No matter how many contexts there are, we can first compute predictions for all available contexts, and then provide a list of suggested contexts by post-filtering the initial list. Post-filtering can be achieved simply by removing irrelevant contexts or by selecting the top $N$ most relevant contexts.

*2)* ***Indirect Context Recommendation****:* Alternatively, we can infer the appropriate contexts by using the traditional context-aware recommendation algorithms. The rating function in CARS is modeled as *R*: *User* $\times$ *Item* $\times$ *Contexts* $\rightarrow$ *Ratings*. We can simply keep the *User* and *Item* unchanged, and vary different values of the *Contexts* to further generate a ranking of *Contexts* based on the predicted ratings. There are several existing CARS algorithms that can be used to perform indirect context recommendation. For example, differential context weighting [19] based on collaborative filtering can be applied, and the importance of the contexts can be inferred based on the weights on the contexts. Also, latent factor models such as Latent Dirichlet Allocation (LDA) can be used to model the contexts based on observations of user interactions with items and to generate the probability of users' interests on different contexts given a specific item.

These approaches, however, are dependent on the effectiveness of the underlying CARS algorithm. A CARS algorithm may work well for context-aware item recommendations, but it is not necessarily the case that this would result in good context recommendations. Also, the computational costs increase with increase in the number of contextual conditions. Here also, pre-filtering performed on the contextual dimensions can be used to reduce the computational costs. Furthermore, post-filtering on contexts can be conducted to adjust the final recommendations.

### C. Evaluation Challenges

The evaluation of context recommendation algorithms may be a big challenge, especially when it comes to the ranking of the contexts. It is difficult to evaluate the recommended contexts if we do not have explicit or implicit feedback about users' preferences on different context dimensions – usually a user study or survey is required for evaluations. For example, two context recommendations for seeing a movie could be "*on Sunday with the family*" and "*in theatre with partner*"; however, it would be difficult to judge which of these is better without any knowledge of past users preferences in similar situations. It is also possible that only the "*partner*" is a sufficient condition and the specific location is not important to the user. Another example is that, *companion* may be filtered out in the approach of Post-filtering Predictions mentioned above, but it may be exactly the key context the user cares about in reality. In short, the ranking or the post-filtering of the contexts cannot be correctly evaluated unless we have users' priority preference on contextual dimensions.

Among the methodologies mentioned above, only the *Pre-filtering Predictions* approach does not require evaluating the ranking of context recommendations, because it assumes the pre-selected contexts are already the relevant ones to be recommended. In that case, the quality of the recommended contexts can be evaluated by the performance of context prediction, using metrics proposed in typical prediction problems. In this paper, we mainly focus on the approach of *Pre-filtering Predictions*. We leave the examination and evaluation of the indirect context recommendation approaches as our work in the future.

## IV. An Approach for Direct Context Prediction

In this paper, we focus on the direct context prediction approach. More specially, we focus on the *Pre-filtering Prediction* approach, where we pre-select the contextual dimensions from which recommendations are generated. We consider approaches based on multi-label classification to predict contextual conditions.

### A. Context Selection and Predictions

As described above, *Pre-filtering Prediction* requires context selection in advance. Contexts can be selected by domain knowledge, user studies [5] or statistical analyses [11]. However, whether the contexts are appropriate to be recommended should also be taken into consideration. In some application, specific combination of contextual conditions may be inappropriate or even impossible due to domain constraints.

The next step is to predict the contexts for a given user. As introduced before, in both pervasive computing and CARS area, the technique of context prediction has been explored. Classification algorithms are the most popular and common ways to predict contexts [8]. In pervasive computing, location is usually the only variable and sequential classification is more preferred because the predicted location should follow the sequential patterns in the history of context usages. However, in the topic of context recommendation, there may be several contextual dimensions required to be predicted and there are usually dependencies among those contexts, thus traditional multi-class classification is not applicable. Correspondingly, multi-label classification (MLC) is a good fit as the solution to the context recommendation problem. In the CARS area, Baltrunas, *et al.* [4] proposed three KNN classifiers: Rating-Based (RB), BestContextVectorBased (BCVB) and BestContextBased (BCB), where they used the same KNN-classifier based prediction function, the only difference is how they find

TABLE I: Transformed Rating Matrix for Multi-label Classification

(a) Original Matrix

| User | Item | Rating | Time | Companion |
|---|---|---|---|---|
| U1 | T1 | 3 | Weekday | Kids |
| U1 | T2 | 5 | Weekend | Kids |
| U2 | T1 | 2 | Weekend | Spouse |

(b) Transformed Matrix

| User | Item | Rating | Time=Weekday | Time=Weekend | Companion=Kids | Companion=Spouse |
|---|---|---|---|---|---|---|
| U1 | T1 | 3 | 1 | 0 | 1 | 0 |
| U1 | T2 | 5 | 0 | 1 | 1 | 0 |
| U2 | T1 | 2 | 0 | 1 | 0 | 1 |

the K-nearest neighbors. Actually those approaches fall into the category of multi-label classification too. In this paper, we explore more MLC algorithms and conduct an empirical study on the performance of using MLC for context predictions.

## B. Multi-label Classification (MLC)

In contrast to traditional classification (e.g. binary-class or multi-class classification), MLC allows multiple predictions on several classes at the same time – the examples are associated with a set of labels Y $\subseteq$ L, where L is a set of all possible labels and there are at least two labels in the set L. It is widely used in annotation applications, e.g. image annotation, text categorization, and it has already been applied to tag recommendations [9] which was demonstrated to obtain accurate predictions with decent degree of personalization. But it is seldom discussed and explored for context predictions and recommendations in existing research, where we try to provide such an overall introduction and evaluation in this paper.

The context predication can be easily transformed to a multi-label classification task, because the contextual conditions can be decomposed into a set of labels. For example, in the movie domain, *Time* is a context dimension, *Weekend* and *Weekday* are two contextual conditions in this dimension, where we can consider each condition as a label in the data. As a result, each contextual condition in all context dimensions can be considered as one label and the value in each label is a binary value (i.e., 1 or 0). An example of transforming the original ratings matrix can be viewed in Table I.

There are mainly two types of MLC algorithms: transformation algorithms and adaptation algorithms [13]. Transformation algorithms are able to convert the MLC task to several single-label classification tasks; as a result, all traditional classifiers (e.g. decision trees, naive bayes, SVM, etc) can be applied to, and it is not necessary to develop new MLC algorithms. For example, the easiest and most straightforward one is binary relevance (BR) which simply perform a binary-class classification for each label individually, but it does not take label dependency into consideration. Label dependency is usually important, e.g. user's choice on the context dimension *Time* may also influence his choice on another dimension *Companion*. However, the significance of label dependency may vary from domains to domains. The research in MLC has further developed several transformation algorithms to additionally take label dependency into consideration, such as label powerset (LP), classifier chains (CC) and random k-labelsets (RAkEL) [13]. Another category is adaptation algorithms where traditional classification algorithms are modified to adapt to the MLC task. But, there are not many of those algorithms; common ones are extensions of KNN classifiers, such as BRKNN and MLKNN. The three KNN classifiers proposed by Baltrunas, *et al.* [4] in the CARS area can be considered as adaptation algorithms too. In this paper, we evaluate different MLC algorithms, including both transformation algorithms (BR, LP, CC, RAkEL) and adaptation algorithms (BRKNN, MLKNN, three KNN classifiers by Baltrunas, *et al.*) to compare their prediction performances.

## V. Experiment Setup

We choose three real-world data sets from the domain of context-aware recommender systems; notice that context-aware data are usually difficult to collect and turn out to be either small or sparse. The TripAdvisor data is an enriched hotel rating data from our previous work [16], rating profiles were scripted from hotel reviews. This data is pretty sparse in context, because very few users lived in a same hotel for several times with different contexts, not to mention that they will leave ratings every time. AdomMovie [2] and LDOS-CoMoDa [10][1] are two context-aware movie data sets collected from surveys. The specific description of data sets can be viewed by Table II, where we just give the number of context conditions in each context, more details about those conditions can be found from previous works using those data. We empirically set a rating threshold (it is 7 for AdomMovie data and 3 for the other two data sets, which are adopted by previous work [7], [22]) to separate the ratings as positive and negative ones, where only ratings larger than this threshold are considered as positives and the inputs of a standard context recommender are {*user, item, positive*}, *positive* is a binary variable to indicate a rating is positive or negative, because the application should suggest the appropriate contexts assuming the user will like the item (i.e. the binary value in *positive* should be 1).

For the context selection, we kept all contexts without selections for the TripAdvisor and AdomMovie data sets, since the number of contexts is limited and they were all demonstrated as influential ones in previous work. There are originally 12 contextual dimensions in the LDOS-CoMoDa data, and we select 4 of them. The selection is based on previous work of statistical relevance analyses [11] on this data and common sense, e.g. the emotional variables are not applicable, even if they were demonstrated as the most influential contexts in our previous work [20], [21]. It is because the emotions in this data set are not the ones before the user choosing and seeing the movie. Weather is assumed as inappropriate one because it is weird to recommend users to see a movie in a specific weather condition. Instead, day, time, location and companion are the most common and influential ones in the movie domain.

For evaluation purposes, we further split the positive ratings into five folds – each fold is considered as the test set, and the other four folds plus a fold of negative rating profiles are

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

TABLE II: Descriptions of Data Sets

| Data sets | # of users | # of items | # of ratings | # of contexts | # of labels | rating scale | # of positives | Contexts | Contents |
|---|---|---|---|---|---|---|---|---|---|
| AdomMovie | 69 | 176 | 1010 | 3 | 8 | 1-13 | 771 | Time (2), Location (2), Companion (4) | User gender, Movie year |
| LDOS-CoMoDa | 113 | 1186 | 2094 | 4 | 17 | 1-5 | 1374 | Time (4), Location (3), Day (3), Companions (7) | User gender, country, Movie year, genre, language |
| TripAdvisor | 2731 | 2269 | 14175 | 1 | 5 | 1-5 | 11216 | TripType (5) | User country, timezone, Hotel city, state, country, timezone |

considered as the training set in each iteration. We performed 5-folds cross-validation for all three data sets. We also took content features into account as additional inputs, our experimental results showed that adding content features helped improve the prediction performance.

For MLC, we use the toolkit *Mulan*[2] [15] which is a popular open-source Java-based library for MLC and it can be easily integrated with Weka[3]. Recall that the transformation MLC algorithms can use any traditional classifiers after the transformation, where in our experiments we tried KNN-classifier (KNN), decision trees (J48), naive bayes (NB), Bayesian nets (BN) and support vector machine (SMO in Weka) as the classifiers which are utilized from the Weka Java library version 3.6.10. BN results are close to the ones by NB and NB performs slightly better, so we just present the BN results in this paper. As introduced before, there are two categories of MLC algorithms, we evaluated BR, CC, LP and RAkEl as the transformation MLC algorithms, and also the MLKNN and BRKNN as the adaptation algorithms. For all KNN-based approaches, we tune up the performances by varying different number of the neighbors. In J48, we tried pruning and without pruning, and the latter one works better. We use non-linear SVM in SMO and set other settings as default ones in the transformation MLC algorithms.

For baselines, we chose two classes of algorithms. The first was popular prediction which came in three forms – global popular (GP, i.e. the most frequent contexts from a global view), user popular (UP, i.e. a user-personalized popular, where predictions are the user-specific most frequent contexts for each user), and item popular (IP, i.e. an item-specific popular prediction). These popular prediction approaches were used to determine whether personalized predictions are required or not, and how MLC performs compared with the simple personalized baselines (i.e. UP and IP). Our second set of baselines were the three KNN classifiers proposed by Baltrunas et al. [4]: RatingBased (RB), BestContextVectorBased (BCVB) and BestContextBased (BCB) which are considered adaptations of MLC algorithm.

For evaluation metrics we used the example-based metrics commonly used for MLC [13]: hamming loss, precision (prec), recall (rec) and accuracy (acc). Hamming loss is defined as the average percentage of incorrectly predicted labels. This metric was also used in the previous work by Baltrunas et al. [4]. The other metrics are defined by Equations 1 below, where $m$ is the number of examples in the testing set, $Y$ is the set of TRUE labels in the ground truth, and $Z$ is the set of predicted TRUE labels.

[2]http://mulan.sourceforge.net/

[3]http://www.cs.waikato.ac.nz/ml/weka/

$$prec = \frac{1}{m}\sum_{i=1}^{m}\frac{|Y_i \cap Z_i|}{|Z_i|} \quad rec = \frac{1}{m}\sum_{i=1}^{m}\frac{|Y_i \cap Z_i|}{|Y_i|} \quad acc = \frac{1}{m}\sum_{i=1}^{m}\frac{|Y_i \cap Z_i|}{|Y_i \cup Z_i|} \tag{1}$$

One challenge worth mentioning is that context prediction is able to predict more or less contexts than the exact number of contexts we expect. For example, contexts in a positive rating profile could be *on weekend at home with kids*. But, the user may be satisfied with just a portion of the full context. For example, *with kids* may be the most important aspect of the context regardless of the time and location. However, it is difficult to evaluate this without any information about users' ranked preferences among contexts. Thus, we allow the predictor to have partially matched predictions. For instance, the predictor may suggest two contextual conditions in a same dimension (e.g. home and theatre at the same time), or miss predictions for some dimensions (e.g. no predictions in the dimension *companion*). In the final set of recommendations there can be further adjustments to the suggested contexts using a post-filtering approach, for example by recommending the most frequent or the most recent location. In this work we focus on the prediction performance and leave this challenge for future work.

There are four experimental questions to be explored:

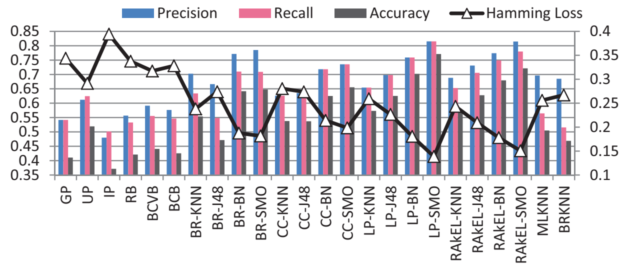
- What is the impact of personalization?
- Which algorithms perform the best? Do MLC algorithms perform better than the baselines?
- Which classifiers (such as, SMO, Bayesian net or KNN classifiers) perform the best among MLC transformation algorithms?
- Which algorithms have the best running performance?

## VI. Experimental Results

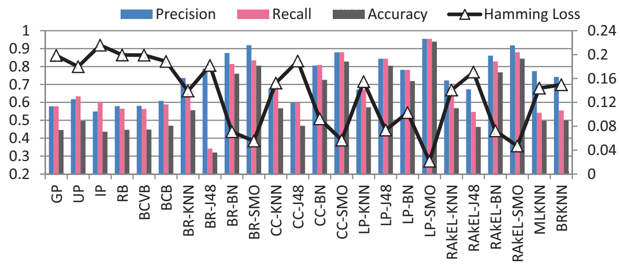
The experimental results are presented in Figure 1. The bars and the left y axis represent precision, recall and accuracy, where the curves and the right y axis represent running time in d) and the hamming loss for other figures. The results are better if it is higher value on the bars (i.e. precision, recall and accuracy) and lower result on the curves (i.e. hamming loss). Figure d) shows the running time in seconds for the LDOS-CoMoDa data. For MLC transformation algorithms, we use "Alg-Classifier" to denote the algorithm, e.g. BR-BN denotes we use binary relevance (BR) as transformation algorithm and choose bayesian nets as the classifier. The patterns revealed from the results can be summarized as follows:

***Is there personalization issue involved?*** The only non-personalized prediction is the global popular (GP) approach, where we can see that personalized approaches generally outperform GP except when the personalized algorithm works

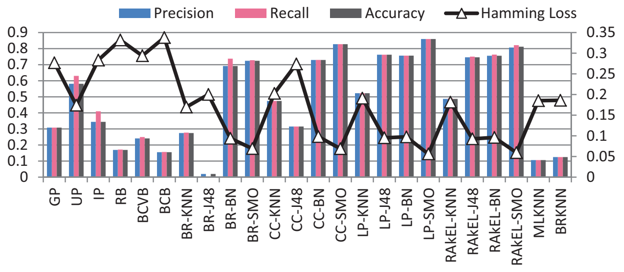
Fig. 1: Experimental Results



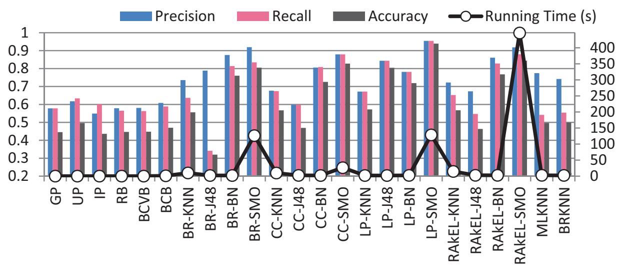
(a) Prediction Performance in AdomMovie



(b) Prediction Performance in LDOS-CoMoDa



(c) Prediction Performance in TripAdvisor



(d) Running Performance in LDOS-CoMoDa

pretty bad, e.g. in TripAdvisor data, BR-J48, MLKNN and BRKNN works worse than GP. This is not surprising, because one user may travel five places one time, and rated five hotels separately, thus all the trip types for those five ratings are the same – GP or UP will work better in the TripAdvisor data. Meantime, we can find many more cases that MLC algorithms work much better than GP, e.g. LP-SMO which is the best performing algorithm which helps obtain an improvement of 65% on precision and 90% on hamming loss over the GP approach in TripAdvisor data. This significant advantage is also shown in other data sets, which indicates that personalized prediction is required in this task.

***Which algorithms perform the best? Do MLC algorithms perform better than the baselines?*** From an overall view, the best performing one is LP approach using SMO as classifier, and the results by RAkEL using SMO are very close to the best one. It is not surprising because RAkEL [14] was developed as an improvement over the LP approach to alleviate the computational cost by LP. RAkEL is supposed to get the optimal results by avoiding iterating all possible subsets of labels, especially when it comes to the situation that there are hundreds and even thousands of labels. However, it is not usual to have more than 10 contexts in the RS domain, and not all contexts are influential, so it is hard to say RAkEL will have any advantages over LP in this task, where we further evaluate the running performance which is introduced later in this paper.

It is also important to pay attention to the values of precision and recall for each algorithm. Notice that the only difference between precision and recall is the denominator in the Equation 1; in other words, it is the number of TRUE labels in the predictions and real situations. As mentioned before, it is possible that predictors may suggest more or less than the number of contextual dimensions in the data. In order to further inspect the details, we design two metrics: $P_{full}$ and $P_{multi}$ shown in the equations as follows:

$$P_{full} = \frac{\text{\# of examples where all contextual dimensions have predictions}}{\text{\# of all examples in the testing set}} \quad (2)$$

$$P_{multi} = \frac{\text{\# of examples where more than one conditions are predicted for a single context}}{\text{\# of all examples in the testing set}} \quad (3)$$

$P_{full}$ equals to the percentage of examples where all contextual dimensions have predictions. And $P_{multi}$, represents the percentage of examples where there are more than one predictions for a single contextual dimension (e.g. *at home* and *at cinema* are both predicted in dimension *location*). The results are shown in Table III, where we use SMO as the classifier for the MLC transformation algorithms. Among all algorithms, LP and Classifier chain are able to provide predictions for all context dimensions (i.e. $P_{full}$ = 100%), and there are no multiple predictions for a single dimension (i.e. $P_{multi}$ = 0%). This pattern happens to all those three data sets – that's the reason why precision and recall values are the same when using those two MLC algorithms as shown in Figure 1. The explanation is that they both take label dependency into consideration and are able to mine the dependencies very well. RAkEL is an improvement over LP approach, but it is an ensemble method which cannot guarantee the same $P_{full}$ and $P_{multi}$ as LP does after it combines the results from predictors. From the table, we can see that RAkEL achieves the same effect as LP and CC only when it was applied to the TripAdvisor data, it is because there is only one contextual dimension in this data, which reduces the complexity in label predictions. The three KNN classifiers (RB, BCVB, BCB) proposed by Baltrunas *et al* cannot guarantee a 100% $P_{full}$ and 0% $P_{multi}$ too, as well as other MLC algorithms. Therefore, LP and CC approaches indicate their advantages over other

TABLE III: Comparison of Prediction Characteristics

| AdomMovie | BR | CC | LP | RAkEL | BRKNN | MLKNN | RB | BCVB | BCB |
|---|---|---|---|---|---|---|---|---|---|
| $P_{full}$ | 65.50% | ***100%*** | ***100%*** | 74.53% | 4.18% | 40.57% | 62.2% | 63.9% | 68.6% |
| $P_{multi}$ | 1.48% | ***0%*** | ***0%*** | 14.15% | 0% | 0% | 2.13% | 2.89% | 3.2% |

| TripAdvisor | BR | CC | LP | RAkEL | BRKNN | MLKNN | RB | BCVB | BCB |
|---|---|---|---|---|---|---|---|---|---|
| $P_{full}$ | 71.64% | ***100%*** | ***100%*** | 100% | 3.35% | 2.74% | 66.4% | 68.14% | 69.2% |
| $P_{multi}$ | 7.01% | ***0%*** | ***0%*** | 0% | 0% | 0% | 0% | 0% | 0% |

| LDOS-CoMoDa | BR | CC | LP | RAkEL | BRKNN | MLKNN | RB | BCVB | BCB |
|---|---|---|---|---|---|---|---|---|---|
| $P_{full}$ | 64.90% | ***100%*** | ***100%*** | 72.30% | 17.90% | 14.30% | 90.4% | 86% | 83.6% |
| $P_{multi}$ | 31.40% | ***0%*** | ***0%*** | 30.60% | 0.05% | 1.10% | 5.6% | 6.4% | 9.6% |

examined MLC algorithms if the application requires a 100% in $P_{full}$ and 0% in $P_{multi}$. And the prediction accuracy indicated by the three metrics also reveals that LP and CC can also provide better predictions than others, especially when SMO is selected as the classifier. Again, a 100% in $P_{full}$ and 0% in $P_{multi}$ could be not necessary, because the final recommendations can be adjusted by post-filtering ways, for example, if two locations (e.g. at home or at cinema) are suggested to see the movie, we can just recommend the location by the most frequent or the most recent location user seeing movies. Or, we can simply skip this dimension and recommend other contexts, which assumes that user may be more interested in other contexts and no matter where the location is to see the movie. However, the evaluations on those suggestions require further user studies to confirm which strategy is the appropriate one.

Compared with the two categories of baselines – the popular predictions and KNN classifiers by Baltrunas et al, MLC approaches using Bayesian nets and SMO outperform them from an overall view. Simple recommendations, such as UP or IP may not perform very well – in other words, it indicates that the most popular contexts for a <user, item> pair are not appropriate for everyone, a more personalized recommender is required. The results by the three KNN classifiers (RB, BCVB, BCB) proposed by Baltrunas et al are close to the ones by popular predictions, and even worse sometimes. Also we find that KNN classifier based approaches, including both MLC transformation algorithms and adaptation algorithms, do not work good compared with other classifiers, because it is not easy to find the accurate nearest neighbors from sparse context-aware data. KNN-based approaches may not work well – this can be considered as a special pattern in context prediction in the context recommendation task unless the data set is pretty dense with contextual information; in other words, there are very few items which were rated by users for multiple times within different contexts.

***Which classifiers perform the best among MLC transformation algorithms?*** In our results, transformed algorithms usually outperform the KNN-based adaptation algorithms. Among all classifiers we evaluated, SMO is generally the best, and bayesian net is the next one. Also keep in mind that SMO scales somewhere between linear and quadratic in the training set size for various test problems [12], thus it is cost-expensive and time-consuming in running time. It is necessary to further evaluate the running performance.

***How about the running performances?*** The running time based on LDOS-CoMoDa can be shown as d) in Figure 1, where running time reveals similar patterns in other data sets. Apparently, SMO cost more time to complete training, where other approaches can complete the training and calculations within 5 to 10 seconds. Similar patterns can be found for the other two data sets, e.g. putting SMO on the TripAdvisor data cost around 4-5 hours, but less than 3 minutes for the other approaches. Taking both prediction and running performances into consideration, classifier chains is the eclectic choice as the MLC transformation algorithm which helps achieve decent prediction performance within acceptable running time. Alternative solution is using Bayesian nets as the classifier instead of SMO, where it saves time but the prediction performance may vary from data to data.

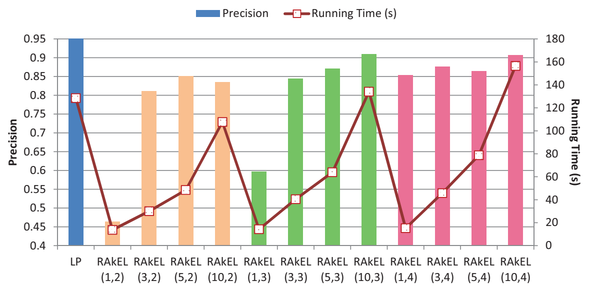


Fig. 2: Running Performance Using SMO as Classifier

Among all those transformation MLC algorithms, the LP and RAkEL approaches are worth further exploring. LP considers each unique subset of labels that exists in the multi-label dataset as a single label. There will be $2^N$ subsets if there are originally $N$ labels, which poses significant label costs in training iterations. RAkEL [14] was developed to optimize the training process by avoiding iterating all possible subsets. There are two parameters in RAkEL: number of subsets (i.e. $k$, $1 \leq k \leq |L|$) and number of models (i.e. $M$, $1 \leq M \leq |L^k|$). RAkEL iteratively constructs an ensemble of $M$ LP classifiers, and it randomly selects a $k$-labelset from $L^k$ without replacement. To compare running performance between LP and RAkEL ($M$, $k$), we choose SMO as the classifier, vary the number of $k$ (i.e. 2, 3, 4) and $M$ (i.e. 1, 3, 5, 10) and provide the results of precision and running time (in seconds) on the LDOS-CoMoDa data as shown in Figure 2. The bars in the figure represent precision, where orange bars using $k$ = 2, green bars $k$ = 3 and red bars $k$ = 4. The curve represents running time in seconds. RAKEL aims to achieve the optimal results by selecting a small number $k$ and appropriate value for $M$. From the figure, we can see that RAkEL may not have significant advantages for LDOS-CoMoDa in balancing precision and running time, compared with LP which just costs 130 seconds. RAkEL may be a good option in MLC task like image annotation or text categorization, because there are hundreds or even thousands of labels. However, it is possible to fully explore LP if the number of labels is limited, which is another reason why we suggest to select relevant contexts in advance to avoid label costs in MLC training iterations. This is one significant difference between the traditional MLC task and the MLC task for context predictions. RAkEL may work better than LP if there are large scale of contexts, but the appropriate number for $M$ and $k$ should be carefully selected to balance the prediction accuracy and running performance.

In a summary, our experiments confirm that personalization improved the performance of context recommendation algorithms, and appropriate MLC algorithms (such as LP-SMO) are able to outperform the baselines: both the popular predictions and the KNN classifiers proposed in previous research. Algorithms should be carefully selected if running performance is also taken into account. We suggest alternative algorithms to the most computationally intensive approach (i.e. LP-SMO), such as using classifier chains as the transformation MLC algorithm, or using a Bayesian net as the classifier. We further compare the LP and RAkEL in running time to reveal the difference of using RAkEL in traditional MLC task and the context prediction task.

## VII. Conclusions and Future Work

In this paper, we introduced a basic framework for the novel context recommendation problem, including two algorithm paradigms to generate context recommendation – direct context prediction and indirect context recommendation. It is difficult to evaluate the ranking of the recommendations if we have no information about users' preferences on contexts. Thus we mainly focus on the direct context prediction approach with pre-filtering of contextual dimensions. Specifically, we pre-select relevant contexts in advance and explore multi-label classification algorithms to predict the context labels representing different combinations of contextual conditions. In terms of the prediction performance, we observed that appropriate MLC algorithms, such as LP using SMO as classifier, can significantly outperform the baselines. We also discuss the running performance of the proposed algorithms. Context recommendation is a novel problem in recommender systems research in its early stages of development. In future work, we will explore other possible algorithms to perform context recommendation, including those based on indirect context recommendation whose evaluation may require user studies or utilizing available implicit feedbacks.

## References

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

[2] G. Adomavicius, R. Sankaranarayanan, S. Sen, and A. Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Transactions on Information Systems (TOIS)*, 23(1):103–145, 2005.

[3] G. Adomavicius and A. Tuzhilin. Tutorial on Context-aware Recommender Systems, the 2nd ACM Conference on Recommender Systems. http://ids.csom.umn.edu/faculty/gedas/talks/RecSys2008-tutorial.pdf.

[4] L. Baltrunas, M. Kaminskas, F. Ricci, R. Lior, B. Shapira, and K.-H. Luke. Best usage context predictions for music tracks. In *Proceedings of the 2nd Workshops on Context-aware Recommender Systems*, 2010.

[5] L. Baltrunas, B. Ludwig, S. Peer, and F. Ricci. Context relevance assessment and exploitation in mobile recommender systems. *Personal and Ubiquitous Computing*, 16(5):507–526, 2012.

[6] 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.

[7] 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.

[8] A. Boytsov and A. Zaslavsky. Context prediction in pervasive computing systems: Achievements and challenges. In *Supporting Real Time Decision-Making*, pages 35–63. Springer, 2011.

[9] I. Katakis, G. Tsoumakas, and I. Vlahavas. Multilabel text classification for automated tag suggestion. In *Proceedings of the ECML-PKDD Discovery Challenge Workshop*, 2008.

[10] A. Košir, A. Odic, M. Kunaver, M. Tkalcic, and J. F. Tasic. Database for contextual personalization. *ELEKTROTEHNISKI VESTNIK*, 78(5):270–274, 2011.

[11] A. Odic, M. Tkalcic, J. F. Tasic, and A. Košir. Relevant context in a movie recommender system: Users opinion vs. statistical detection. *ACM RecSys*, 12, 2012.

[12] J. Platt et al. Sequential minimal optimization: A fast algorithm for training support vector machines. 1998.

[13] G. Tsoumakas, I. Katakis, and I. Vlahavas. Mining multi-label data. In *Data mining and knowledge discovery handbook*, pages 667–685. Springer, 2010.

[14] G. Tsoumakas and I. Vlahavas. Random k-labelsets: An ensemble method for multilabel classification. In *Machine Learning: ECML 2007*, pages 406–417. Springer, 2007.

[15] G. Tsoumakas, E. S. Xioufis, J. Vilcek, and I. P. Vlahavas. Mulan: A java library for multi-label learning. *Journal of Machine Learning Research*, 12(7):2411–2414, 2011.

[16] 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.

[17] 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*. ACM, 2012.

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

[19] 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.

[20] Y. Zheng, R. Burke, and B. Mobasher. The role of emotions in context-aware recommendation. In *ACM RecSys' 13, Proceedings of the 3rd International Workshop on Human Decision Making in Recommender Systems*, pages 21–28. ACM, 2013.

[21] Y. Zheng, R. Burke, and B. Mobasher. Emotions in context-aware recommender systems (to appear). In M. Tkali, B. Decarolis, M. de Gemmis, A. Koir, and A. Odi, editors, *Book: Emotions and Personality in Personalized Services*. Springer Berlin Heidelberg, Germany, 2014.

[22] Y. Zheng, R. Burke, and B. Mobasher. Splitting approaches for context-aware recommendation: An empirical study. In *Proceedings of the 29th ACM Symposium on Applied Computing*, pages 274–279. ACM, 2014.