Differential Context Relaxation for Context-Aware Travel Recommendation

Yong Zheng, Robin Burke, and Bamshad Mobasher

Center for Web Intelligence
School of Computing, DePaul University
Chicago, Illinois, USA
{yzheng8,rburke,mobasher}@cs.depaul.edu

Abstract. Context-aware recommendation (CARS) has been shown to be an effective approach to recommendation in a number of domains. However, the problem of identifying appropriate contextual variables remains: using too many contextual variables risks a drastic increase in dimensionality and a loss of accuracy in recommendation. In this paper, we propose a novel treatment of context – identifying influential contexts for different algorithm components instead of for the whole algorithm. Based on this idea, we take traditional user-based collaborative filtering (CF) as an example, decompose it into three context-sensitive components, and propose a hybrid contextual approach. We then identify appropriate relaxations of contextual constraints for each algorithm component. The effectiveness of context relaxation is demonstrated by comparison of three algorithms using a travel data set: a contenxt-ignorant approach, contextual pre-filtering, and our hybrid contextual algorithm. The experiments show that choosing an appropriate relaxation of the contextual constraints for each component of an algorithm outperforms strict application of the context.

Keywords: Context-aware Recommender System, Collaborative Filtering, Contextual Pre-Filtering, Context Relaxation.

1 Introduction

The application of context to recommender systems has a strong intuitive appeal. It makes sense that a user's preferences will change from situation to situation: the set of restaurants desirable for a dinner date is not the same as for a quick business lunch. This insight is the motivation behind the expansion of recent research in context-aware recommender systems (CARS) [3,4,2]. A number of researchers have shown that appropriate application of context can create effective recommendation solutions in a variety of domains [5,7,10,15].

One of the key problems in CARS research is the identification of contextual variables. It is clear that these factors are very domain-specific. In the restaurant example above, obviously the occasion of the meal is an important contextual variable, but also the time of day, the day of the week, and the user's location are all plausible candidates, and if the dataset included all of this information, it would be tempting to build all available variables into a contextual model. However, it is unlikely that all of these contextual factors will repeat exactly in peer users' profiles, so choosing contextual variables becomes crucial. [16]

Context-aware algorithms typically require that contextual features are selected in advance and applied throughout the recommendation computation. By contrast, we propose a novel way to apply contexts in recommendation. First, we decompose a recommendation algorithm into different functional components. Second, we create a formalism for contextual constraints, so that contextual features may be fully or partially matched in computing recommendations. Last, we identify appropriate relaxations of the contextual constraints for each algorithm component. Applying context to recommendation therefore becomes a task not only of finding the best subset of contextual variables to select, but of discovering the best way to use context in each part of the algorithm. We take the traditional user-based collaborative recommendation as an example, decompose it into three components, apply our hybrid contextual approach and demonstrate the effectiveness of contextual relaxation for each component by comparing against our other baseline algorithms.

2 Related Work

Context-aware recommendation has been the subject of intensive research over the past several years, with a number of workshops and challenges on different aspects of the topic [CARS, RecSys; CAMRa, RecSys; CaRR, IUI; CoMoRea, PerCom; ICAS, CISIS; IIiX; etc].

Surprisingly, the field has yet even to agree on the definition of a context. The most commonly used definition is the one given in 1999 by Abowd *et al.* "*Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.*" [1] This definition hardly limits the set of variables that can be considered when performing recommendation.

In [2], G. Adomavicius, et al. introduce a two-part classification of contextual information. Their taxonomy is based on two considerations: what a recommender system knows about a contextual factor and how the contextual factor changes over time. System knowledge can be subdivided into *fully observable*, *partially observable* and *unobservable*. The temporal aspect of a factor can be categorized as *static* or *dynamic*. This analysis yields six possible classes for contextual factors. In this paper, we are concerned with *static* and *fully observable* factors – the question is which such factors to incorporate into a recommendation model and which to ignore.

Typically, researchers working in a particular domain will identify contextual variables that they believe are important, and that are available in their data. The question then becomes one of discovering the importance of each of these variables. Baltrunas *et al* [6] conducted a survey asking participants to evaluate if a particular contextual factor influenced their ratings or not, in order to acquire contextual relevance from subjective judgements and further build predictive models for mobile recommender systems. Another attempt to explore contextual relevance is proposed by Huang *et al* [12], where they combine attributes of contexts with items directly and extract a set of significant contextual attributes to index a particular item based on rough set theory reduction. The process is similar to contextual feature selection [16] or attribute reduction, and it requires large datasets with contextual ratings under multi-dimensional contexts for training purposes.

However, few available datasets have rich contextual ratings – cases where users have rated the same item multiple times in different contexts – or explicit meta-information from users about the importance of context. This limits the applicability of the methods discussed above where the context is treated as a strict constraint. In this paper, we assume that relaxations of the contexts can also play influential roles in recommendation, especially when the contextual information is sparse in the dataset. For instance, if a user is going to take a business trip to Chicago, he may take advice from his friend who once went to Chicago although the two may travel for different purposes.

2.1 Travel Recommendation

For our experiments, we make use of a dataset crawled from Tripadvisor.com, which is one of the world's largest travel sites. The site supports travel planning with its extensive database of reviews of travel destinations and accommodations. Tripadvisor.com allows users to post ratings and reviews for hotels where they have lodged, and it provides an option for the users to explicitly indicate the type of trip using one of five categories: family, friend, couples, solo and business trip. Previous research into context-aware recommendation on Tripadvisor.com has used this trip type information as the key contextual variable [11,14].

Research on tourism behavior [9,13] indicates that geographical locations including both the origins and destinations of travel are considered as influential factors beyond user profiles, hotel amenities and prices. Klenosky and Gitelson [13] also pointed out that trip type appeared to have a primary influence and trip origin a secondary influence on travel recommendations. We decided to follow this line of inquiry and incorporate location of origin and destination as contextual variables in addition to trip type.

We quickly discovered that exact matches in origin city and destination city were fairly rare. So, we generalized the notion of location, creating a geographical hierachy for locations in the United States with three levels: city, state, and time zone.

3 Methodology

As described in [3], there are three basic approaches to context-aware recommendation: pre-filtering, post-filtering, and contextual modeling. The filtering approaches apply a context-dependent criterion to list of items, selecting only those appropriate to a given context. For example, in a pre-filtering system using location as a context, all items that are not nearby the user would be filtered out before recommendations are processed. A post-filtering approach applies the filter after recommendations have been computed. These approaches work best if the contextual data is relatively dense – most objects can be evaluated relative to most contexts. Contextual modeling, by contrast, builds contextual considerations into the recommendation process itself. The application of this technique therefore is intimately tied to the particular recommendation algorithm being employed. In this section, we show how a user-based collaborative algorithm can be adapted to form a contextual hybrid.

3.1 Algorithm Decomposition

User-based collaborative recommendation using Resnick's popular algorithm is treated as a prediction problem where the task is to predict the expected rating $P_{a,i}$ that a user a would give to an item i. The first step in this process is to identify a set of neighbors N who are similar to user a (often using Pearson correlation as a metric). Once the neighbors are identified, the prediction is given by

$$P_{a,i} = \bar{r}_a + \frac{\sum\limits_{u \in N} (r_{u,i} - \bar{r}_u) \times sim(a,u)}{\sum\limits_{u \in N} sim(a,u)} \tag{1}$$

where a is the target user, i is an item, N is a neighborhood of users u similar to a, and sim is a similarity measure between users. Correspondingly, $r_{u,i}$ is neighbor u's rating on item i, $\bar{r}_a$ is user a's average rating over all items, and $\bar{r}_u$ is user u's average rating.

We can think of this algorithm as weighting the variance of neighbors' ratings by their similarity to the user and then applying this calculated variance to the user's own average rating. The assumption is that every user has an average "baseline" rating that may differ from user to user and what is significant about a particular rating is how it deviates from the user's personal baseline.

We turn this algorithm into a context-sensitive one by incorporating context into the computation of each prediction. Assume that there exists some contextual constraint C that applies to the current situation and we are attempting to recommend items that would be appropriate to a in this context. There are three places in this algorithm where context can come into play, and three components that could be used to form a contextual recommender:

1. Instead of considering all users who have rated item i, the neighborhood calculation can choose only users who have rated item i in a context that matches C.
2. The neighbor's baseline that is used to calculate the variance for each neighbor $\bar{r}_u$ can be limited to those ratings associated with contexts matching C, rather than an overall average for that user. For example, a user might be less stringent in his rating of hotels he stays in for business because he generally has little choice, but rate more critically on family vacations where he is paying himself.
3. The user's baseline $\bar{r}_a$ can be a function of C for the same reason.

3.2 Context Relaxation

For our purposes, we will assume that each single context c consists of a set of contextual features and that there is a binary matching process between the contextual constraint C and any individual context c – either a context matches the constraint or it does not.[1] For this research, we are considering three possible types of constraints for

[1] We plan to consider the partial matching of constraints in future research.

each feature: `any`, which means no constraint for a given feature, `exact`, meaning that the feature must match exactly, and `contained` which applies to location information where one geographical designation (like city) might be contained in another (such as time zone).

For example, we may have the following contextual features: { `trip_type`, `trip_duration`, `origin_city`, `destination_city`, `month` }. If we know that Adam, a resident of Los Angeles, stayed at Club Quarters during his stay in Chicago on business for three days last summer and gave it a rating of 4 stars, the overall context of this rating would be C_{adam} = { "business trip", "3", "Los Angeles", "Chicago", "July" }. If Betsy is seeking a prediction for Club Quarters, but she is coming from Seattle on business and staying for a week in January, Adam's rating would not contribute to her prediction if the algorithm requires a strict match between contexts. This highlights a typical problem in applying context strictly with multiple context dimensions – the more features are used, the greater the sparsity of the recommendation data and the more often it will fail to find any neighbors at all. For this reason, researchers typically concentrate on only one or two contextual features, such as only the trip type.

Suppose we relax the context constraint somewhat, for example: C = { `(exact trip_type), (any duration), (contained time_zone origin_city), (exact destination_city), (any month)` }. We can see that this constraint relaxes the original context by ignoring `month` and `duration` and allowing any `origin_city` in a matching time zone. Now there is a match between the two contexts, and we can use Adam as a neighbor when predicting whether or not Betsy will want to stay at Club Quarters during her trip. The most extreme relaxation of the contextual constraint is to ignore context altogether, resulting in the original context-insensitive prediction algorithm given above.

In this research, we ask the question if we can get better predictions by using different relaxations of the context for different purposes in the same algorithm. In other words, instead of narrowing C to just a few variables, which is what researchers typically do to avoid excessive sparsity, we define relaxations of C applicable to different parts of the prediction function. For example, we might use `trip_type` to select neighbors, but `origin_city` to establish a user baseline. Above, we identified three functions of context in Resnick's algorithm: neighbor selection, neighbor baseline, and user baseline. We will notate the relaxed versions of C applied to each of these areas as C_1, C_2, and C_3, respectively.

As a result, the task of identifying influential contexts becomes the task of discovering the best choices as the context relaxation (C_1, C_2 and C_3) for the three components. Insufficient relaxation will not solve the problem of sparsity and may impair predictive performance. Excessive relaxation will fail to capture the contextual effects that make one circumstance different from another.

3.3 Hybrid Contextual Modeling

Based on the ideas above, we introduce the context relaxation to the three algorithm components mentioned previously and come up a hybrid contextual modeling approach.

We start by defining a new prediction algorithm in which these constraints play a role. We need three new terms:

- N_{C_1} is the set of neighbors of user a, filtered such that those neighbors have rated item i in a context that satisfies contraint C_1.
- $\bar{r}_{u,C_2}$ is the baseline for neighbor u, taking into account only those ratings given in contexts that satisfy C_2.
- $\bar{r}_{a,C_3}$ is the baseline for user a, the target user, using only those ratings given in contexts satifying C_3.

With these modifications in place, we can now state a context-sensitive version of the prediction formula as below, where C_1, C_2 and C_3 are relaxed versions of the original constraint C. Note that $P_{a,i,C}$ turns out to be the predicted rating for user a on item i under contexts C, where r_{u,i,C_2} is selected neighbor u's rating on item i under the relaxed contexts C_2.

$$P_{a,i,C} = \bar{r}_{a,C_3} + \frac{\sum\limits_{u \in N_{C_1}} (r_{u,i,C_2} - \bar{r}_{u,C_2}) \times sim(a,u)}{\sum\limits_{u \in N_{C_1}} sim(a,u)} \tag{2}$$

Algorithm 1

Require: Context constraints C_1, C_2, C_3
Require: Ratings database R, training set S and testing set T
Ensure: Coverage score
Ensure: RMSE

```
$E \Leftarrow \emptyset$ (vector of errors)
$k \Leftarrow 0$
for each $t \in T$ do
    $a \Leftarrow t.user$
    $i \Leftarrow t.item$
    $c \Leftarrow t.context$
    $r \Leftarrow t.rating$
    $c_1 \Leftarrow$ Apply$(c, C_1)$
    $c_2 \Leftarrow$ Apply$(c, C_2)$
    $c_3 \Leftarrow$ Apply$(c, C_3)$
    $N \Leftarrow$ Neighbors$(a, c_1, S)$

    if $|N| > 0$ then
        $k \Leftarrow k + 1$
    end if
    $p \Leftarrow \bar{r}_{a,c_3} + \frac{\sum_{u \in N} (r_{u,i,c_2} - \bar{r}_{u,c_2}) \times sim(a,u)}{\sum_{u \in N} sim(a,u)}$
    $E = E$.append $< r, p >$
end for
$Coverage = k/|T|$
Calculate RMSE from the vector of errors $E$
```

Algorithm 1 shows how the experimental methodology is implemented. It assumes that we have a database of ratings R, where a rating consists of a 4-tuple: user, item, rating and contextual features associated with that rating, and that database is split into a training set S and a testing set T. The algorithm takes the three contextual constraints C_1, C_2, and C_3, and applies them to the specific context of the test rating, yielding an instantiation of each constraint. We assume that our neighborhood selection and baseline averaging functions make use of these constraints appropriately. We iterate through all items in the test set, evaluating the context-sensitive predictions with the three constraints each playing their individual role.

We consider this algorithm as a contextual hybrid. A hybrid recommendation algorithm [8] combines two or more recommendation techniques to provide recommendations. In our case, we decompose the user-based CF into three components where the 1^{st} component filters out neighbors who never rate item in matched contexts. This is a form of contextual pre-filtering. However, we also introduce contexts to the other two components in the model, where the selections for C_2 and C_3 are allowed to be different than C_1, thus it is a contextual modeling approach in view that contexts are introduced into the process of modeling. The combination of these three applications of context results in hybrid of contextual pre-filtering and contextual modeling.

3.4 Experimental Setup and Design

We selected hotels in the 120 largest cities in USA from Tripadvisor.com, and crawled ratings as well as geographical location information from user and hotel profiles. We assume that the origin location is the same as the user's geographical information as defined in his or her profile. As mentioned above, many profiles are incomplete, either without home city information or without state information. We opted only to include state and time zone as features for origin location. Users from outside of the USA were removed. The final dataset includes user id, user's state of residence with time zone information, hotel id, hotel city as well as the associated state and time zone information, trip type and user's rating. In order to conduct 3-fold cross-validations, we only include users who have at least 3 ratings in the dataset and then split the dataset into 3 folds using 1/3 of each user's profile. In the process of selecting neighbors, we selected neighbors from the users who have rated the same hotel in the training set, and user similarity is measured by Pearson correlation coefficient.

A closer look at the data revealed that users were selecting multiple contexts for a single review.[2] For example, a business traveler might annotate one stay at a hotel with both "business" and "solo". In these cases, we really have only one rating for the hotel, not multiple ones in different contexts. Therefore, we filtered the context information in this and similar cases. If a rating listed both "business" and "solo" as trip types, we kept only "business" under the assumption that "business" is a stronger descriptor of the traveling situation in those cases. We handled the pairs "business" / "couples" and "business" / "family" by selecting "couples" and "family" respectively, based on

[2] The current version of the Tripadvisor.com website only allows a single trip type to be chosen for each rating/review, but there is still legacy data in the system from when multiple trip type entries were permitted.

the understanding that any time other parties are along for the trip, their needs will influence the acceptability and hence the rating for a hotel. This filtering step ensures that we have only one trip type for each rating and no double counting.

After the pre-processing above, the full dataset before the split contains 2,562 users, 1,455 hotels and 9,251 ratings. 45% of users have multiple ratings in at least one city, and 13% of users had multiple reviews of the same hotel under different trip purposes.

In our experiments, we consider three different contextual variables for hotel recommendation. Trip type is the reason that the user gives for his or her trip. Destination city is the location of the hotel being rated, and origin city is the location from which the user's travel begins. We examine two different types of `contained` constraints for the geographic features (state and time zone) in addition to the `exact` and `any` possibilities that we consider for type type. So, the space of possible relaxations theoretically has 2 possibilities for trip type, 4 for destination, and 4 for origin, giving 32 possibilities for each of C_1, C_2, and C_3, a total space of $32^3 = 32k$ possible constraints to search for our optimal set of relaxations.

The characteristics of our data and the algorithm allow us to reduce the combinatorial complexity considerably. Recall that we select neighbors from users who have rated the same item, thus the item features in the contextual vector C_1 is unnecessary. Similarly, C_2 and C_3 are applied to ratings coming from one user, so user demographic information is redundant as a constraint. With these considerations, the search space becomes greatly reduced in size – $6 \times 8 \times 8 = 384$ possibilities – quite tractable for complete search.

We run an exhaustive search to fully explore the performance of different context relaxations in terms of RMSE and coverage, where coverage indicates the percentage of instances in which neighbors can be found for collaborative prediction.

In our experiments, we compare our algorithm to two classes of competitors. The first type is user-based collaborative recommendation without incorporating context. The second is a typical contextual recommendation approach using contextual pre-filtering. In our formalism, contextual pre-filtering is achieved by applying contextual constraints to C_1 only.

4 Experimental Results

In order to compare how context relaxation performs, we introduce three context strategies in the model: 1) use trip type only (as previous authors have done with TripAdvisor data); 2) use strict contexts without relaxation; or, 3) use relaxed contexts for each component – so that we can compare the performance of trip type, strict contexts and relaxed contexts. Keep in mind that the contextual pre-filtering in our experiments is the one which we only introduce contexts to the 1^{st} component for neighbor filtering. The results of the experiment are shown by Figure 1 and Table 1.

In terms of RMSE, experimental results in Figure 1 show that introducing contexts always outperform the standard CF without contexts. The differences in RMSE between all conditions are significant at the 10^{-3} level using paired Student's t-test, and a p-value of less than 0.01 if using the Wilcoxon signed-rank test. Applying context relaxation outperforms situations where we use trip type only or use strict contexts. Our

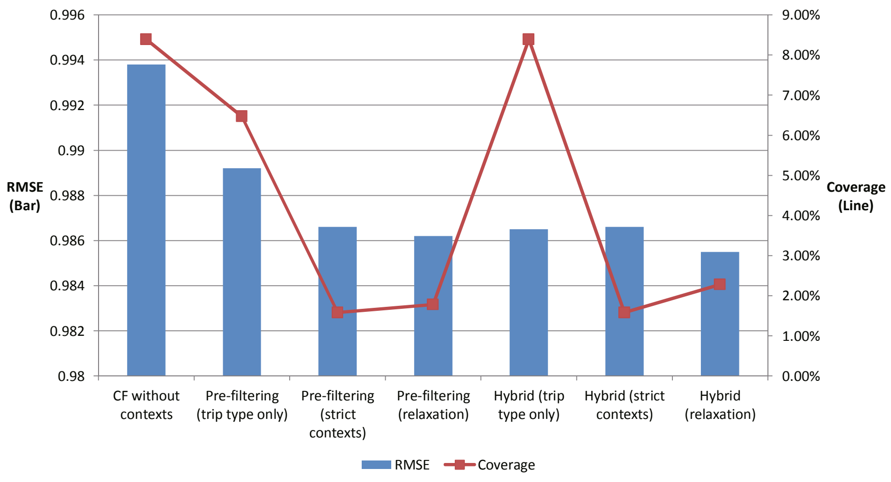

Fig. 1. Comparison of Algorithms in terms of RMSE

hybrid contextual model actually performs better than contextual pre-filtering when using the same context strategy. The best performing one is our hybrid contextual modeling approach using context relaxation, where the RMSE is improved from 0.9938 (the baseline without contexts) to 0.9855.

The coverage in all cases is low (see Figure 1 and Table 1), which means that the default rating is assigned in the vast majority of cases. Recall that C_3 is the constraint that controls which items are chosen to be averaged to create this default. This part of the calculation is the same for the baseline and the optimum algorithms, which means that improved accuracy on a relatively small number of possible predictions accounts for all of the difference in accuracy, which although small are significant.

Recall that when we cannot find a neighbor in our algorithm, we use default predicted ratings where no collaborative predictions and no contextual effects (because the best selection for C_3 is empty without contexts, see Table 1. Thus it is necessary to evaluate the effects contributed by the valid collaborative efforts only – the situation that we can find neighbors and use our hybrid contextual model to make predictions. If we compare the algorithms just on their collaborative predictions – rule out the default predictions and calculate the RMSE on the remaining predictions only, we see a much larger difference between the baseline (RMSE 1.1453) and the optimal relaxation (RMSE 1.0703), an improvement of 6.6%. This suggests that in a dataset with greater density would see an even greater benefit from this technique.

From the Table 1, we can see the optimum set of relaxed constraints can be summarized as follows.

- C_1 = Neighbors filtered based on the state of origin.
- C_2 = Neighbors' baselines computed based on trip type.
- C_3 = Users' baseline computed over all prior ratings.

Trip type and point of origin turn out to be useful aspects of the context in view of the optimal context relaxation for both contextual pre-filtering and our hybrid contextual

Table 1. Comparison of Algorithms

Algorithm	Optimal Contextual Constraints	RMSE	Coverage
CF Without contexts	{}	0.9938	8.39%
Pre-filtering (trip type only)	C_1= {`exact` trip_type}	0.9892	6.47%
Pre-filtering (strict contexts)	C_1= all contexts	0.9866	1.58%
Pre-filtering (relaxation)	C_1= {(`exact` trip_type), (`contained` origin_city state)}	0.9862	1.78%
Hybrid (trip type only)	C_1= {}, C_2={`exact` trip_type}, C_3={}	0.9865	8.39%
Hybrid (strict contexts)	C_1= all contexts, C_2={}, C_3={}	0.9866	1.58%
Hybrid (relaxation)	C_1= {`contained` origin_city state} C_2= {`exact` trip_type}, C_3= {}	0.9855	2.28%

modeling approach – the only difference is they are applied into different components. Our hybrid approach reveals that placing the state of origin in the 1st component and trip type in the 2nd component can help achieve the best RMSE. Destination information seems not to be important here, perhaps because the choice of hotel already subsumes this information. We suspect that if we were considering hotel chains (Hilton, Mariott, etc.) with multiple locations, destination would be much more useful. We also find that "state" is the most effective level of abstraction at which to consider origin information. This is the most specific level at which we have origin data.

To more fully understand these results and the dependencies between each aspect of the algorithm, we performed a sensitivity analysis of this optimum condition, looking at the results in which two of the constraints are held fixed and the other is varied. Holding C_2 and C_3 fixed at their optimal values allows us to see the shape of the optimization space as C_1 changes. See Table 2.

The RMSE difference is not large between these relaxation options. Because C_1 controls which neighbors are allowed to contribute to the recommendation, the coverage goes down significantly when the constraint is strict and increases at the more relaxed settings. In all, we see that there is benefit to using origin information, even though some coverage is sacrificed in doing so.

Table 2. Top 3 Relaxations for C_1 in Hybrid Approach

Optimal Relaxation for C_1	Contextual Constraints	RMSE	Coverage
Top-1	C_1 ={ (`contained` origin_city state)}	0.9855	2.28%
Top-2	C_1 ={ (`exact` trip_type), (`contained` origin_city state)}	0.9857	1.78%
Top-3	C_1 ={ (`contained` origin_city timezone)}	0.9859	5.48%

We performed a similar analysis for C_2 and C_3. The results are shown in Table 3. Coverage is not impacted by these constraints, so it is omitted from the tables. As suggested in prior research, matching trip type is important for establishing a useful neighbor baseline via C_2. The destination city has only a weak impact on accuracy.

For C_3, we find that there is a large difference between using all of the user's data (empty constraint) and more focused calculation of the user baseline. This constraint shows the sharpest RMSE "valley" in the constraint space.

Table 3. Top 3 Relaxations for C_2 and C_3 in Hybrid Approach

Optimal Relaxation for C_2	Contextual Constraints	RMSE
Top-1	C_2 ={ (exact trip_type)}	0.9855
Top-2	C_2 ={ (exact trip_type), (contained dest_city timezone)}	0.9857
Top-3	C_2 ={ (exact trip_type), (contained dest_city state)}	0.9858
Optimal Relaxation for C_3	**Contextual Constraints**	**RMSE**
Top-1	C_3 ={ }	0.9855
Top-2	C_3 ={ (exact trip_type), (exact dest_city)}	1.0172
Top-3	C_3 ={ (exact dest_city)}	1.0220

Therefore the remaining contexts in our optimal model are trip type and the state of origin, which is consistent with Klenosky and Gitelson [13]'s findings in their research, where they point out that origin is functionally related to the distance and the time users needed for travel. In this case, origin usually plays an important role for personal travel – the non-business trip, which occupies the largest percentage in our dataset.

The tradeoff between coverage and context-sensitivity appears in this study as it has in many others. Our baseline algorithm already has very poor coverage (below 10%) and our optimal variant has only about 1/4 of that. However, the fact that accuracy can be improved at that level of coverage (and without altering the user baseline C_3) means that we are greatly improving recommendations for the small number of cases in which neighbors matching the origin state can be found. We believe this is a very good sign because it suggests that even small increases in density (more neighbors) may yield substantial accuracy improvements for our contextual recommendation approach.

5 Conclusion

Researchers in context-aware recommendation have long known that the danger of sparsity arises if contextual information is applied too strictly. The usual response is to eliminate most contextual variables from consideration, and to focus on the one or two most salient.

In this work, we introduce the ideas of algorithm decomposition and context relaxation, and apply these ideas to a user-based collaborative recommendation algorithm. We break the algorithm into three components and show how contextual constraints can be applied independently in each component. Through exhaustive search, we locate optimal relaxations for our travel data set, and show that error can be significantly reduced. This benefit comes at the cost of reduced coverage, but coverage is not as low as it is when the context is applied strictly. Our approach can be easily applied to other algorithms and datasets, and we plan to do so in future research. We expect that, in other datasets, we will not be able to use exhaustive search over all possible combinations of constraints, and in our future work, we will need more efficient techniques to search the constraint space.

References

1. Abowd, G.D., Dey, A.K., Brown, P.J., Davies, N., Smith, M., Steggles, P.: Towards a Better Understanding of Context and Context-Awareness. In: Gellersen, H.-W. (ed.) HUC 1999. LNCS, vol. 1707, pp. 304–307. Springer, Heidelberg (1999)
2. Adomavicius, G., Mobasher, B., Ricci, F., Tuzhilin, A.: Context-aware recommender systems. AI Magazine 32(3), 67–80 (2011)
3. Adomavicius, G., Tuzhilin, A.: Context-aware recommender systems. In: Recommender Systems Handbook, pp. 217–253 (2011)
4. Anand, S.S., Mobasher, B.: Contextual Recommendation. In: Berendt, B., Hotho, A., Mladenic, D., Semeraro, G. (eds.) WebMine 2006. LNCS (LNAI), vol. 4737, pp. 142–160. Springer, Heidelberg (2007)
5. Baltrunas, L., Amatriain, X.: Towards time-dependant recommendation based on implicit feedback. In: ACM RecSys 2009, the 1st Workshop on Context-Aware Recommender Systems, CARS 2009 (2009)
6. Baltrunas, L., Ludwig, B., Peer, S., Ricci, F.: Context relevance assessment and exploitation in mobile recommender systems. Personal and Ubiquitous Computing, 1–20 (2011)
7. Böhmer, M., Bauer, G., Krüger, A.: Exploring the design space of context-aware recommender systems that suggest mobile applications. In: ACM RecSys 2010, the 2nd Workshop on Context-Aware Recommender Systems, CARS-2010 (2010)
8. Burke, R.: Hybrid recommender systems: Survey and experiments. User modeling and User-adapted Interaction 12(4), 331–370 (2002)
9. Chan, E., Wong, S.: Hotel selection: when price is not the issue. Journal of Vacation Marketing 12(2), 142–159 (2006)
10. Domingues, M., Jorge, A., Soares, C.: Using contextual information as virtual items on top-n recommender systems. In: ACM RecSys 2009, the 1st Workshop on Context-Aware Recommender Systems, CARS 2009 (2009)
11. Hariri, N., Mobasher, B., Burke, R., Zheng, Y.: Context-aware recommendation based on review mining. In: Proceedings of the 9th Workshop on Intelligent Techniques for Web Personalization and Recommender Systems (ITWP 2011), p. 30 (2011)
12. Huang, Z., Lu, X., Duan, H.: Context-aware recommendation using rough set model and collaborative filtering. Artificial Intelligence Review, 1–15 (2011)
13. Klenosky, D., Gitelson, R.: Travel agents destination recommendations. Annals of Tourism Research 25(3), 661–674 (1998)
14. Liu, L., Lecue, F., Mehandjiev, N., Xu, L.: Using context similarity for service recommendation. In: 2010 IEEE Fourth International Conference on Semantic Computing (ICSC), pp. 277–284. IEEE (2010)
15. Lombardi, S., Anand, S.S, Gorgoglione, M.: Context and customer behavior in recommendation. In: ACM RecSys 2009, the 1st Workshop on Context-Aware Recommender Systems, CARS 2009 (2009)
16. Vargas-Govea, B., González-Serna, G., Ponce-Medellín, R.: Effects of relevant contextual features in the performance of a restaurant recommender system. In: ACM RecSys 2011, the 3rd Workshop on Context-Aware Recommender Systems, CARS 2011. ACM (2011)