Static Preference Models for Options with Dynamic Extent

Thomas Bauereiß, Stefan Mandl, and Bernd Ludwig

Dept. of Computer Science 8 (Artificial Intelligence),
Friedrich-Alexander-Universität Erlangen-Nürnberg,
Haberstraße 2, D-91058 Erlangen, Germany
thomas@bauereiss.name, {Stefan.Mandl,Bernd.Ludwig}@cs.fau.de
http://www8.informatik.uni-erlangen.de

Abstract. Models of user preferences are an important resource to improve the user experience of recommender systems. Using user feedback static preference models can be adapted over time. Still, if the options to choose from themselves have temporal extent, dynamic preferences have to be taken into account even when answering a single query. In this paper we propose that static preference models could be used in such situations by identifying an appropriate set of features.

1 Introduction

Preferences are an important component of personalized information retrieval and recommendation systems (see [4]). Given a recommendation or information retrieval system that interacts with the user such that the user enters queries and the system produces a list of options the user may choose from, the order of the presented items typically is determined by the quality of the item with respect to some *objective* value function. Given that this value function is user-pareto-optimal in the sense, that it cannot be changed to improve from the perspective of one potential user without worsening from the perspective of another potential user, any further improvement of the system can only be achieved by personalized user specific adaptations, hence user models and user preferences.

In this paper, after discussing the use of standard machine learning techniques for classification and regression to represent user preferences in a standard scenario—TV program recommendations—, we focus on a tour recommender. The major difference between tour planning and TV programs is that under the usual granularity of discourse, TV programs typically are considered as singular events (though they have temporal extent) while tours and trips are considered multi-part events.

The goal of this paper is to empirically identify proper features of multi-part options in the tour-planning scenario such that standard models of preferences can be used to enhance user experience.

Section 2 gives a short account on modeling user preferences: Section 2.1 gives a formal definition of preferences, Section 2.2 contains some notes on preference elicitation, Section 2.3 presents various concrete preferences models, and

R. Dillmann et al. (Eds.): KI 2010, LNAI 6359, pp. 99–106, 2010.

Section 2.4 specifies the performance measures we use to evaluate preference models. Section 3.1 describes a user survey and the use of the preference models discussed before in a standard TV program recommendation scenario. Section 3.2 describes a set of features for multi-part tours such that the previous preference models can be used for this multi-part setup. Results from a preliminary study show that the proposed method of using standard preference models with a dynamic set of features is appropriate for the task. Section 4 concludes the paper and discusses future research.

2 Preference Modeling

2.1 Formalizing Preferences

In [12], several formalizations of preferences are discussed. They have in common that preferences are expressed as binary relations on a set of options. If a user prefers an option a to another option b, then the pair (a, b) is an element of her preference relation. In this paper, we define these relations in the following way:

Definition 1. *A preference relation $P \subset O \times O$ on a set of options O is a strict weak ordering with the properties*

- *Irreflexivity:* $\forall a \in O : \neg P(a, a)$
- *Asymmetry:* $\forall a, b \in O : P(a, b) \longrightarrow \neg P(b, a)$
- *Transitivity:* $\forall a, b, c \in O : (P(a, b) \wedge P(b, c)) \longrightarrow P(a, c)$
- *Transitivity of equivalence:* $\forall a, b, c \in O : P(a, b) \longrightarrow (P(a, c) \vee P(c, b))$.

This definition is consistent with properties commonly associated with preferences, e.g. irreflexivity (an option a is not considered to be preferred to itself). It also implies the existence of a numerical representation of preference relations [12, Theorem 6.2]:

Theorem 1. *For a preference relation $P \subset O \times O$ on a finite set of options O, a mapping $g : O \mapsto \mathbb{R}^+$ exists such that $\forall a, b \in O : (a, b) \in P$ iff $g(a) > g(b)$.*

This means that the prediction of preferences may be understood as the prediction of this representation function $g : O \mapsto \mathbb{R}^+$. For every option we aim to calculate a real number such that the ordering induced by the values of g for a set of options matches the user's preferences.

2.2 Preference Elicitation

Filling the preference model is an important task for systems using preferences. Possible ways to acquire the desired information are letting the user specify rules for her preferences [13] or gathering implicit [7] or explicit feedback [10]. In this paper, we use the latter approach and let users rate options on a scale from 1 (disliked) to 9 (liked). The underlying assumption is that we can infer what the user might currently like by using information about what she liked in the past.

The motivation to use explicit feedback instead of implicit feedback is the fact that in this paper we want to focus on the preference model and its *evaluation*. Therefore, features that enhance user experience at the cost of measurability of system correctness – like implicit feedback – are disregarded here.

2.3 Preference Models

Common ways of using feedback information to predict preferences are collaborative filtering [1] and content-based filtering [11], which can also be combined [3]. In this paper, we focus on content-based filtering. For this approach, options have to be turned into feature vectors, which can then be used as input for standard machine learning methods. In Section 3 we use different kinds of regression and classification models provided by the Weka toolkit for data mining [5].

Regression models try to predict a user's rating of an option as a real number, based on the user's previous ratings. These numbers are interpreted as values of a preference representation function g as mentioned in section 2.1.

Classification models assign a discrete class value to an option. The options previously rated by the user have to be separated into two classes, e.g. by labeling options with a rating below the user's average rating as "disliked" and options with a rating above average as "liked". The classifier can then be trained with this labeled set of examples and used to predict class labels for new options. When two options are assigned the same class label, the class probabilities provided by the classifier are used to order the options. The higher the estimated probability of an option the more it is considered to be preferred: The class probabilites are used as preference scores $g(a)$.

The preference models considered here are static, as the dynamics of preferences changing over time and from one situation to another are not explicitly modeled. Still, static preference models can capture long-term changes in preferences by adapting to user feedback. Additionally, we propose that static models can also be used for preferences regarding dynamic options, where the options themselves have temporal extent.

2.4 Performance Measures for Preference Models

In order to evaluate preference models, we gathered rating data from a set of users, performed 10-fold stratified cross-validation [8] for each user and averaged the results. These results were calculated in terms of the normalized distance-based performance measure (NDPM) [14], which is defined as:

$$ndpm(P_u, P_s) = \frac{2C^- + C^u}{2C} = \frac{2|P_u \cap P_s^{-1}| + |P_u \cap I_s|}{2|P_u|} \tag{1}$$

where C^-, C^u, and C are the numbers of contradicting, compatible and total user preference pairs, respectively, P_u and P_s are the user's actual preference relation and the one calculated by the system, respectively, P_s^{-1} is the inverse relation $\{(a,b) \mid (b,a) \in P_s\}$, and the indifference relation corresponding to the preference relation calculated by the system is denoted by $I_s =$

$\{(a,b) \mid (a,b) \notin P_s \wedge (b,a) \notin P_s\}$. Basically, this measure counts the differences between the user and the system relation, where compatible pairs are counted once and contradictory pairs are counted twice. This count is then normalized with the maximum distance $2|P_u|$, such that values of *ndpm* lie in the interval from 0 to 1, where 0 indicates a perfect match between predicted and actual user preference, 1 is the maximum distance, and 0.5 is equivalent to the performance of a random generator (assigning uniformly distributed random preference scores $g(a)$ to all available options).

Additionally, we used two other measures, one based on the symmetric set difference between the relations and one based on the mean squared error of the rank $rank(o, P)$ assigned to an option o according to a preference relation P:

$$rdpm(P_u, P_s) = \frac{|P_u \Delta P_s|}{\mathrm{E}\left[|P_u \Delta P_r|\right]} = \frac{|(P_u \setminus P_s) \cup (P_s \setminus P_u)|}{\mathrm{E}\left[|(P_u \setminus P_r) \cup (P_r \setminus P_u)|\right]} \tag{2}$$

$$rsre(P_u, P_s) = \frac{\frac{1}{|O|} \sum_{o \in O} \left(rank(o, P_u) - rank(o, P_s)\right)^2}{\mathrm{E}\left[\frac{1}{|O|} \sum_{o \in O} \left(rank(o, P_u) - rank(o, P_r)\right)^2\right]} \tag{3}$$

where P_r is a randomly generated preference relation and $\mathrm{E}[\cdot]$ is the expected value (estimated by calculating a sample mean). In case of partial preference orders, we estimated the expected value regarding randomly generated linear extensions to total orders by calculating a sample mean. With these measures, an ideal match between user and system corresponds to a value of 0, the performance of a random generator to 1. We call these measures relative distance-based performance measure (RDPM) and relative squared rank error (RSRE), respectively. Because of the normalization with the empiric performance of a random generator, we expected these two measures to be less optimistic than NDPM.

In the following section, we evaluate several preference models for two application domains, one with static and one with dynamic options. Although the domains are very different in nature, we show that after finding suitable features, the same methods can be applied to predict preferences in both cases.

3 Evaluation of Static Preference Models in Static and Dynamic Domains

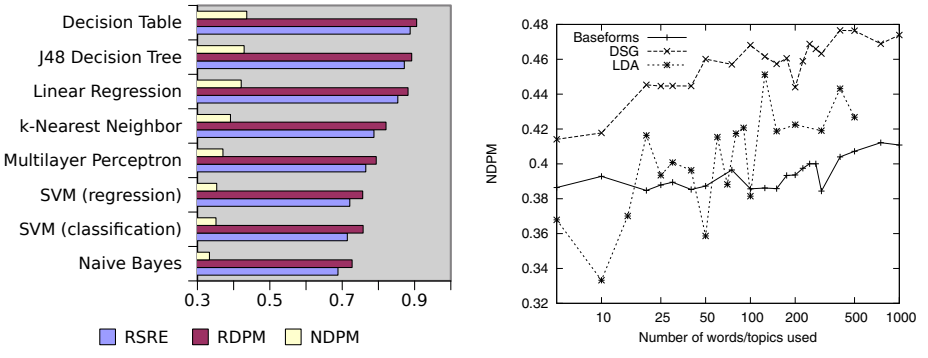
3.1 TV Program Recommender—A Domain with Static Options

We extend the TV Program Recommender that is presented in [9]. Queries have the form of free text terms. Ranking is performed by following the standard *bag of words* approach. For titles and descriptions of TV programs, the following feature sets are used:

- Baseform – every (normalized) word is a dimension in the feature vector
- DSG (Dornseiff group) – baseforms are mapped to topics which are defined in the Dornseiff lexicon for German language.

– LDA (Latent Dirichlet Allocation [2]) – baseforms are mapped to corpus-specific topics using a probabilistic model

Additionally, we use other features available from electronic program guide (EPG) data, namely the channel the program is broadcast on, the weekday and time of broadcast and the duration of the program. We gathered data for evaluating different preference models using an online questionnaire. 41 participants gave ratings (i.e. explicit feedback) for an average number of 45.6 programs.



(a) Comparison of preference models

(b) Evaluation of textual features

Fig. 1. Evaluation of models and features for the TV domain (smaller values are better)

The results displayed in Fig. 1(a) show a high consistency between the three performance measures. We evaluated several models provided by Weka: probabilistic methods (Naive Bayes), instance-based methods (k-nearest neighbor), neural networks (multilayer perceptron), decision trees, decision tables and support vector machines. For our data, Naive Bayes achieves the best performance.

We also experimented with different ways of representing and selecting features, in particular for the textual features of program titles and descriptions. Fig. 1(b) shows the results of using three different kinds of textual features with a Naive Bayes classifier. We evaluated the use of baseforms, Dornseiff groups and LDA topics to represent title and description of programs. Additionally, we evaluated different choices for the number of words or topics to be used. We employed a feature selection method that ranks the features according to their mutual information with the class and selects a fixed number of features with the highest scores. The results show that for our application, the best performance is achieved with relatively low numbers of selected words or topics. For LDA, the number of topics must be configured at the time of topic generation. The performance when using LDA features is highly dependent on the choice of this number. In our case, setting the number of topics to 10 produced the best results. LDA features with 10 topics also performed better than the other two kinds of textual features for our data.

3.2 Tour Planner—A Domain with Dynamic Options

The second application domain we investigated is a tour planner that was designed for big events like the "Long Night of Sciences" in Erlangen, Nuremberg and Fürth. Throughout this "long night", hundreds of talks, workshops, film presentations and information booths are offered on a single evening. The system allows the user to select a set of individual events that she is interested in. It then recommends a detailed tour plan consisting of a sequence of events and the details of the journeys from one event location to the next one. In its current version, the system recommends the tour with the maximum number of events.

However, it is desirable to take the preferences of individual users into account. In contrast to the static options in the case of TV programs, the options in this application domain are dynamic in nature. Not only are they generated dynamically for each user query, but the temporal extent and dynamics of the tours themselves like sequence and lengths of events, journeys and pauses are essential for user preferences. Still, we propose static preference models as discussed before to be used for these dynamic options by finding suitable features.

Table 1. Feature evaluation in terms of NDPM for individual users (smaller values are better; bold features are selected with the method described in [6])

Feature	User 1	User 2	User 3
Number of events	**0.374**	0.286	**0.175**
Average event duration	0.264	**0.102**	0.274
Event descriptions	**0.356**	**0.461**	**0.424**
Topic spread of event descriptions	0.520	0.311	0.363
Number of visited places	**0.307**	0.142	0.228
Ratio of events with fixed start and end time	**0.240**	**0.130**	**0.266**
Percentage of time spent at events	0.217	**0.344**	0.596
Ratio of journey duration to event duration	0.228	0.285	0.573
Elapsed time at begin of first event	0.424	0.363	0.395
Number of nearby events during pauses	0.580	0.667	0.476
Longest waiting time	0.710	**0.376**	0.302
Shortest waiting time	0.543	0.500	0.458
Longest foot walk	0.563	0.512	0.432
Longest bus ride	0.373	0.435	0.407
Number of bus changes	0.383	0.254	0.353
Number of different bus lines used	0.358	0.251	0.322
Distance between events	0.287	**0.172**	**0.172**
Performance with feature selection [6]	0.285	0.119	0.217
Performance with all features used	0.180	0.109	0.148

In search of these features, we asked several users for aspects that might be important for them when rating a tour, and formalized these ideas as features. Table 1 lists the features that resulted from this process. They represent different aspects of the composition and temporal dynamics of a tour, like the selection of events (e.g. number of events, descriptions or topic spread of events in the tour), travel modalities (e.g. number of bus changes, duration of longest bus ride or total distance travelled throughout the evening) and pauses (e.g. longest waiting time before an event, average ratio of waiting time to duration of the event or number of nearby events while waiting for another event to begin).

In order to evaluate these features, we created a set of test items by constructing nine different queries and collecting the six top results returned by the tour planner. These 54 tours were labeled according to the preferences of three different users[1]:

- User 1 prefers talks and film presentations to other kinds of events, like workshops and information stands. She also prefers to stay in the same place when possible, and avoids bus travels.
- User 2 wants to visit as many different places as possible, even if this means that she can only visit few and short events at each location.
- User 3 wants to attend as many events as possible throughout the evening. She prefers to have a tight schedule and as little waiting time as possible.

We then compared the same preference models with each other as for the TV recommender domain by performing cross-validation for all models with the labeled sets of tours for each user. The results are shown in Fig. 2. Naive Bayes, the best model for the TV recommender, also does well for the tour planner, but the best performance in this case is achieved with SVM classification. We therefore used SVM classification to calculate performance measures for each of the proposed features. Table 1 lists the results of the evaluation of these features for the three users individually. The results show that all features have predictive value (i.e. an NDPM value less than 0.5, which would be equivalent to a random generator) for at least one of the users. The feature selection method described in [6] does a good job at selecting small subsets of features while retaining performance, but it doesn't improve performance in our case. Overall, the model based on Support Vector Machines in combination with our proposed set of features achieves good performance at predicting preferences for tours, even better results than in the TV recommender scenario.

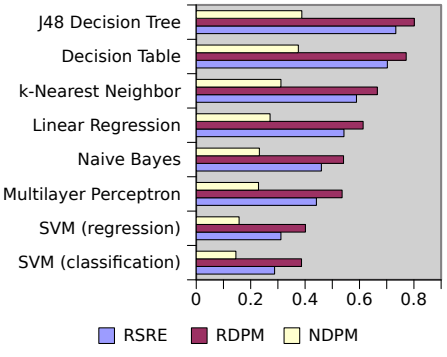


Fig. 2. Comparison of preference models for the tour planner (smaller values are better)

4 Conclusions

In this paper, we described an approach to content-based, static preference modeling. It is based on representing the options available to the user with features

[1] The data set can be downloaded from `http://www8.informatik.uni-erlangen.de/en/datasets.html`

and then using machine learning methods to adapt a preference model over time by incorporating feedback from the user. We introduced two application domains, and discussed the differences between them. The options to which preferences refer in the TV recommender scenario can be considered singular events, while in the tour planner domain, preference models need to deal with options consisting of multiple parts with temporal extent and dynamics. We presented empirical results from preliminary studies showing that the approach to preference modeling described in the paper can be applied successfully to application domains with static and dynamic options by finding suitable sets of features. In the future, we intend to validate our findings for other application domains and larger data sets, for example by gathering ratings from users at future events like the upcoming "Long Night of Music" in Munich.

References

1. Balabanović, M., Shoham, Y.: Fab: content-based, collaborative recommendation. Communications of the ACM 40(3), 66–72 (1997)
2. Blei, D., Ng, A., Jordan, M.: Latent dirichlet allocation. The Journal of Machine Learning Research 3, 993–1022 (2003)
3. Burke, R.: Hybrid web recommender systems. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.) Adaptive Web 2007. LNCS, vol. 4321, pp. 377–408. Springer, Heidelberg (2007)
4. Chen, P.M., Kuo, F.C.: An information retrieval system based on a user prole. The Journal of Systems and Software 54 (2000)
5. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.: The WEKA data mining software: An update. ACM SIGKDD Explorations Newsletter 11(1), 10–18 (2009)
6. Hall, M.A.: Correlation-based Feature Subset Selection for Machine Learning. Ph.D. thesis, University of Waikato (1998)
7. Kelly, D., Teevan, J.: Implicit feedback for inferring user preference: a bibliography. In: ACM SIGIR Forum, vol. 37, pp. 18–28. ACM, New York (2003)
8. Kohavi, R.: A study of cross-validation and bootstrap for accuracy estimation and model selection. In: International Joint Conference on Artificial Intelligence, Citeseer, vol. 14, pp. 1137–1145 (1995)
9. Ludwig, B., Mandl, S.: Centering information retrieval to the user. Revue D'intelligence Artificielle 24, 96–120 (2010)
10. McNee, S., Lam, S., Konstan, J., Riedl, J.: Interfaces for eliciting new user preferences in recommender systems. In: User Modeling 2003, pp. 178–187. Springer, Heidelberg (2003)
11. Mooney, R., Roy, L.: Content-based book recommending using learning for text categorization. In: Proceedings of the Fifth ACM Conference on Digital Libraries, pp. 195–204. ACM, New York (2000)
12. Öztürk, M., Tsoukiàs, A., Vincke, P.: Preference modelling. Tech. Rep. 2003-34, DIMACS (2003)
13. Stefanidis, K., Pitoura, E., Vassiliadis, P.: Adding context to preferences. In: Proc. ICDE, Citeseer, pp. 846–855 (2007)
14. Yao, Y.: Measuring retrieval effectiveness based on user preference of documents. Journal of the American Society for Information Science 46(2), 133–145 (1995)