USING SEMANTIC CUES FOR CONTEXTUAL RECOMMENDATION

Maryam Ramezani, Andriy Shepitsen, Runa Bhaumik, Robin Burke, Bamshad Mobasher
DePaul University
School of Computer Science, Telecommunications and Information Systems
mramezani,ashepits,rbhaumik, mobasher, rburke @cs.depaul.edu

Abstract

Recommender systems help users overcome the information overload problem and have been widely used in an ever-increasing number of e-commerce websites. However, most existing recommender systems use simplistic user models. In this paper, we describe how context can be brought to bear on recommender systems. We propose a new approach to integrating user rating vectors with a contextual retrieval cue to generate recommendations. We use a user model consisting of short and long term memory with context playing the role of retrieval and we use semantic information extracted from the domain knowledge as the key cue for distinguishing user context. An evaluation of our recommendation algorithm was carried out using Movielens data by extending standard collaborating filtering algorithm using semantic cues.

1 Introduction

Recommender systems tend to use very simplistic user models. For example, user-based collaborative recommendation [1, 2] generally models the user as a vector of item ratings; content-based recommendation methods [3, 4] tend to use models such as na"ive Bayes or simple feature vectors. The user models also tend to be additive in nature. For example, in collaborative recommendation, as more ratings are provided by the user, they are simply added to the existing set of ratings and used all together to identify peer users from whom to extrapolate ratings. Similarly, content-based techniques simply update their feature probabilities as new items are rated.

This additive approach ignores the notion of “situated action” [5], that is, the fact that users interact with systems within a particular context and items relevant within one context may be irrelevant in another. Consider the following example. The user buys and rates contemporary fiction for himself (“Gravity’s Rainbow”), work-related books on computer science topics (“Programming Python”), and books for his children (“Where’s Waldo?”). It makes little sense to represent this user’s “interest in books” in a single representation that aggregates all of these disparate works. Yet that is precisely what most recommender systems will do. The ideal contextual recommendation system would therefore be able to reliably label each user action with a context. Thus, neighbors with similar tastes in children’s books would be used only when the “children’s book” context is active and would be ignored otherwise.

While little agreement exists among researchers as to what constitutes context, the importance of context is undisputed. In psychology, a change in context during learning has
been shown to have an impact on recall \[6, 7\], suggesting a key role played by context in the structure and processing of human memory. Research in linguistics has shown that context plays the important role of a disambiguation function: that is, it reduces the possible interpretations of a message \[8\].

However, user context has been largely ignored in research in recommender systems. This paper aims to remedy this omission, via a fundamental shift in user modeling, inspired by psychological models of human memory. The long-term component of the model contains a collection of preference models, one for each identified context. A model for the current situation is stored in short-term store. Recommendation is performed using the contents of the short-term model, after it has been augmented by data retrieved from the long-term memory. Cues from the short-term model are used to identify and retrieve those portions of the long-term model appropriate to the current context.

For example, in the case of the book-buying user above, we might imagine three models stored in the long-term profile: one for contemporary fiction \(m_1\), one for children's books \(m_2\), and another for computer books \(m_3\). If the user were browsing the children's section of a book catalog, it would be appropriate to retrieve \(m_2\). This preference information would then be combined with whatever information was currently being gathered from the user's interaction to form the basis for recommendation.

Semantic knowledge is an essential part of the user context. The domain knowledge can be used as the key source to disambiguate different contexts. For example in a movie recommendation system, the semantic features extracted from the domain knowledge could be the movie-genre, director, actor and other related fields. We envision that using the semantic information can help distinguish between different contexts and lead to more accurate recommendation.

In this paper, we first present a framework for contextual recommendation in which semantic knowledge of item can be used. In particular, we look at a movie recommender system and consider genre as a context. Finally, we propose algorithms that use semantic cues in conjunction with collaborative filtering algorithm.
Table 1: User Profile for User 1

2 Contextual Recommendation Framework

Our model is based on Atkinson and Shriffin’s model of human memory [9]. This model consists of three structural components of human memory: the sensory register, the short term store, and the long term store. From the perspective of a recommender system, a user interacts with the system through implicit and explicit input. These inputs can be thought of as being directly input to the short term store. User preferences from previous interactions of the user are stored within the long term store as memory objects. These memory objects should be retrieved and transferred to the short term store to aid in recommendation generation. The cues for this retrieval are generated from the contents of the short-term store. Finally, an updated representation of the context and the user preferences will be extracted and transferred back to the long term store for use in future interactions.

The overall framework for contextual recommendation is depicted in Figure 1. The user model is divided between a short-term store $M_S$ and a long-term store $M_L$. $M_L$ consists of a set of preference models: $\{m_1, \ldots, m_k\}$, where each model is labeled with a representation $L_i$ of the interaction context to which that model is relevant. These will be analogous to the memory objects stored in the long term memory. The $M_S$ is the working memory, where user preference data provided by the user is stored during on-line interaction. The contents of $M_S$ are processed to generate contextual cues (see Section 2.2), which are probabilistically associated with the contextual labels in $M_L$. The most relevant preference models can then be retrieved from the long-term store to augment $M_S$.

2.1 Recommendation Generation

The standard formulation of the recommendation problem is as follows. We have a set of $m$ users, $U = \{u_k : 1 \leq k \leq m\}$, and a set of $n$ items, $I = \{i_j : 1 \leq j \leq n\}$. Let $u_a \in U$, referred to as the active user, represent the user whose navigation through $I$ needs to be personalized. In previous interactions, $u_a$ will have provided explicit or implicit ratings for a set of items $I_a \subset I$. Typically, systems assume that the user $u_a$’s preferences on items in $I$ are represented via a rating function $r_a$, defined as $r_a : I \rightarrow [0, M]$, where $M$ is some maximum rating value. We refer to the set $Q_a = I - I_a$ as the candidate item set for the user $u_a$. The goal of the recommendation engine is to select a set of items, $R_a \subseteq Q_a$ consisting of items of interest to the user. This is achieved by approximating $r_a$ from the ratings in $I_a$ and any other data made available to the system, typically, an item knowledge base and ratings by other users in $U$. In the standard formulation, the collective ratings of the user represented by $r_a$ is considered to be the only memory object in the long term store.

We can describe the task of generating contextual recommendations as follows. Data
generated in the active interaction, such as ratings or other inputs, are stored in the STM. Contextual cues are then derived from this data and used to retrieve some preference models from the user’s LTM that are deemed to belong to the same context as the active interaction. These are merged with the information stored in the STM and the whole contents of the short-term store is used to generate recommendations. For example, consider Table 1, which contains information about the users short and long term previous preferences. In the case of contextual recommendation, the recommender system takes movies from the STM and then tries to find an appropriate context in the LTM. In Table 1 the user rated two movies in STM: The Sixth Sense and Paycheck. The recommender system using item semantic cues determines that the interaction 2 from LTM is very close to the one in the STM, because movies Star Wars and Cliff Hanger belong to the Action and Adventure genre. Then the STM is going to be extended only by part of LTM which belongs to the same context and only the current context is used for generating recommendations.

2.2 Types of Contextual Retrieval Cues

In this section we identify four different types of cues that can be generated from data stored in STM and discuss how these cues are generated. The key requirement that these cues must meet is that they must reflect different user contexts. One of the techniques that is used for retrieving information from LTM is collaborative cues. The main idea behind this algorithm is that it uses every item as \( m \)-dimensional vectors consisting of ratings for the item by the \( m \) users of the system. During recommendation generation the STM is extended by similar items from LTM which exceeds some predefined similarity limit.

Another approach in combining STM and LTM is through the usage of behavioral cues. This algorithm uses the similarity of the user’s current behavior on the web-site with his previous activities and choose the most alike interactions for extending current user STM. An alternative approach, when an item ontology is available, is to extract latent factors that drive user choice, for example, impact values extracted using Kullback-Leibler’s Information Divergence [10] and use these as the basis for describing user behavior.

Some types of recommendation interaction uses more straightforward approach for getting information about the user favorites and may involve extended recommendation dialogs in which user’s preferences and constraints are incrementally elicited. A critiquing dialog [11] is an example of such a recommendation design. Contextual cues may be gathered as part of such an interaction. The system may, for example, suggest constraints or present alternatives to discriminate between possible contexts.

The other method of extending STM is through the usage of item semantic cues for finding relevant items in LTM. Semantic cues are similar to collaborative cues in that they measure the similarity of the user preference model from the active interaction with those stored in the user’s LTM and retrieve those interactions from LTM that have a similarity, greater than a pre-defined threshold, with the active ratings. However these cues assume the existence of an item knowledge base and use item semantics to compute similarity between items. If items of interest are text-based documents, then textual features and weights can be obtained using methods such as the standard tf*idf approach commonly used in information retrieval. In our opinion this approach is particularly effective for the movie recommender systems as the users preferences frequently are influenced by movie semantic information such as actors,
directors, genres, etc. In this article we described in details the usage of item semantic cues and evaluated the usage of the genre as the divider for the user contexts. The idea behind our approach is that the user could have dissimilar tastes in different type of movies. For example, if we find nearest neighbors for the target user using movies belonging to the “action and adventure” genre their tastes in drama type movies could be different. In order to increase the precision of the recommender system and overcome the above stated problem we propose finding neighbors and generating recommendations inside of particular genres to which target movie belongs or using similarity among genres as normalization factor during recommendation generation.

These cues need not be used in isolation during recommendation generation. In Section 3 we propose algorithms that use semantic cues in conjunction with collaborative filtering algorithm.

3  Contextual Collaborative Filtering Using Semantic Cues

The standard collaborative filtering algorithm is based on user-to-user similarity [12]. This $k$NN algorithm operates by selecting the $k$ most similar users to the target user, and formulates a prediction by combining the preferences of these users. $k$NN is widely used and reasonably accurate. The similarity between the target user, $u$, and a neighbor, $v$, can be calculated by the Pearson’s correlation coefficient defined below:

$$
sim_{u,v} = \frac{\sum_{i \in I} (r_{u,i} - \bar{r}_u) * (r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I} (r_{u,i} - \bar{r}_u)^2} \times \sqrt{\sum_{i \in I} (r_{v,i} - \bar{r}_v)^2}}
$$

where $I$ is the set of all items rated by both users, $r_{u,i}$ and $r_{v,i}$ are the ratings of some item $i$ for the target user $u$ and a neighbor $v$, respectively, and $\bar{r}_u$ and $\bar{r}_v$ are the average of the ratings of $u$ and $v$ over $I$, respectively. Once similarities are calculated, the most similar users are selected. In our implementation, we have used a value of 20 for the neighborhood size $k$. We also filter out all neighbors with a similarity of less than 0.1 to prevent predictions being based on very distant or negative correlations.

Once the most similar users are identified, we use the following formula to compute the prediction for an item $i$ for target user $u$.

$$p_{u,i} = \bar{r}_v + \frac{\sum_{v \in V} sim_{u,v} (r_{v,i} - \bar{r}_v)}{\sum_{v \in V} |sim_{u,v}|}
$$

where $V$ is the set of $k$ similar users and $r_{v,i}$ is the rating of those users who have rated item $i$, $\bar{r}_v$ is the average rating for the target user over all rated items, and $sim_{u,v}$ is the mean-adjusted Pearson correlation described above.

Consider a user who wants movie recommendation. In the existing collaborative approach, the recommendation depends on user profiles, generally in the form of ratings given by the user in past interactions. All past ratings equivalently build user profile and are used to make prediction for a new item. However, when a user rates a movie highly it could be influenced by different aspects of that movie such as movie genres, actors, directors or probably by
combination of all different semantic features related to that movie. These semantic features can be used to distinguish user context. Neighbors of active user can be based on rating within the current user context. In this paper, we extend the standard collaborative filtering algorithm by incorporating genre as one semantic feature of movies. In future we will use other semantic features like actor, director etc.

The most straightforward algorithm to incorporate genre in collaborative filtering algorithm is considering the user ratings in that specific genre to find the neighbors. More specifically, to generate prediction for a particular movie we first identify the genre of the target movie (the movie we want to make prediction for). We calculate the similarity between users using the formula $sim_{u,v}$ described above. In this case, $I$ is the set of all items rated by both users belonging to the target movie genre. We then compute the prediction using the formula $p_{u,i}$. In movielens data, the movies are categorized to 19 different genres and each movie can belong to up to 5 different genres. In this algorithm, if a movie belongs to multiple genres, we calculate prediction for each genre and average them. In practice this algorithm will not produce better results than the standard algorithm. We foresee two major problems with the this algorithm. The first problem is data sparsity already existed in collaborative filtering algorithm. If we want to consider the ratings in each genres separately the rating data will be even more sparse which will make the predictions less accurate. The second problem is that the genres of different movies are not independent of each other. There may be overlap in different genres and one movie can be categorized in different genres. In addition many of the genres may be correlated to each other. In order to solve the above mentioned problems we propose four different algorithms which are based on the baseline simple algorithm described above.

**Algorithm 1 : Disjunction of Contexts**

In our base line algorithm, if one movie belongs to multiple genres, we first compute the prediction for each genre and then average the results. Considering each genre separately, the rating data will be sparser. To overcome this problem, we consider the disjunction (OR) of those genres. More specifically, if one movie belongs to drama and action genre, then the set $I$ is the set of all items rated by both users belonging to these two genres.

**Algorithm 2 : Weighted Average**

In this algorithm instead of taking average of all predictions in baseline algorithm, we follow an alternative approach here. For producing the final prediction for a movie, we multiply each genre prediction by a constant factor. This factor is the user’s average rating for that particular genre. Suppose a movie belongs to 5 different genres and $p_1, \cdots, p_5$ are the prediction for a movie belonging to each of these genres. Our final prediction can be computed as, $P = \sum_{i=1}^{5} p_i \ast C$, where the term $C$ is the user’s average rating for genre $i$.

In the other two algorithms we propose to consider the correlation between different contexts in addition to taking the overlapping contexts into account. Our method of computing correlation between different contexts is described as follows. Consider $U_{m \times n}$, $I_{n \times c}$ and $C_{m \times c}$
as user-item, item-context and user-context matrices, where \( m \), \( n \) and \( c \) are the number of users, items and contexts respectively. In context of a movie recommender system, the user-item matrix is the user rating data where each user has rated different movies and the item-context matrix is a binary matrix where genre represents as context. Each element \( i_{nc} \) in item-context is 1 if movie \( n \) belongs to genre \( c \), and 0 otherwise. We first compute the user-context matrix by multiplying the user-item matrix with item-context matrix as shown below.

\[
[C]_{m \times c} = [U]_{m \times n} \times [I]_{n \times c}
\]

The user-context matrix incorporates both rating and semantic information. The correlation between each pair of contexts is then computed from the user-context matrix. Algorithm 3 and Algorithm 4 describe the use of correlated contexts.

**Algorithm 3 : Top Correlated Contexts**

For recommendation purpose we first find the top correlated contexts to the target context and we take into account not only the target context but also the top-correlated ones. By considering genre as context, we first calculate the genre correlation matrix from user-genre matrix as described above. To make a prediction for a target movie we first find the top 3 correlated genres with the target movie genre. We then take into account the ratings of these correlated genres in order to find the neighbors to the target user.

**Algorithm 4 : Generalized Context, Correlations as Weight**

This approach suggests that rather than using the related context and the top correlated ones, we can use all contexts with a weighting factor. The weight factor is the correlation between each context and the active context. In this approach, we used the weights in calculating the similarity between users. Formally, the user-based formula will be modified as:

\[
sim_{u,v} = \frac{\sum_{i \in I} (r_{u,i} - \bar{r}_u) \times (r_{v,i} - \bar{r}_v) \times \text{corr}(C_i, C_u)}{\sqrt{\sum_{i \in I} (r_{u,i} - \bar{r}_u)^2} \times \sqrt{\sum_{i \in I} (r_{v,i} - \bar{r}_v)^2}}
\]

where \( C_i \) is the context of the \( i \)-th item and \( C_u \) is the context of the target item(i.e. context of active item which the system wants to make prediction for). \( \text{corr}(C_i, C_u) \) is the correlation factor between the active user context and other contexts. The new similarity measure is a combined measure which consists of both rating data and semantic data. We use this new similarity measure to find the nearest neighbors and the prediction formula is remain unchanged.
4 Evaluation

4.1 Data Sets

We use MovieLens data set containing 100,000 explicit numeric ratings on 1682 movies from 943 users. Each user has rated 20 or more movies with a rating scale of 1 to 5. Content information associated with these movies will be extracted using our own wrapper agent that extracts movie instances from the Internet Movie Database \(^1\) based on a simple movie ontology.

4.2 Evaluation Metrics

There are a variety of metrics for evaluating recommender systems [13]. As our goal is to measure the predictive accuracy of recommender systems, we are interested particularly in a commonly used metric Mean Absolute Error (MAE) which accomplishes this task. This metric measures the average absolute deviation between a predicted rating and the user’s true rating. There are two advantages to Mean Absolute Error. First, the mechanics of the computation are simple and easy to understand. Second, it is well-studied statistical properties that provide for testing the significance of a difference between the mean absolute errors of two systems.

4.3 Experimental Results

Figure 2 shows the results of generating recommendations using movielens data. We compare our different contextual approaches described in Section 3 with the standard User-Based collaborative algorithm. As seen from the result, only the generalized approach which takes the genre correlations as weight factor performs slightly better than the standard collaborative filtering.

Due to sparsity, our experimental results did not perform well. As we focus on only one genre, we are taking into account only the ratings of that specific genre which means we do not use all of available data. So, we are losing some information which should be considered but with a lower weight. In the disjunction of genres we try to overcome the sparsity problem by taking into account not only one genre but by using a bigger portion of data which contains all ratings in overlapping genres. However, as we can see in the results this approach does not produce better results. This is due to the fact that we are considering all overlapping genres with the same weight though they should have different weight factors considering their correlation to the target genre. We obtain better results by considering three top correlated genres rather than simply taking the disjunction of overlapping genres. Interestingly, this result is not still better than the standard collaborative filtering algorithm because different genres are highly correlated. Therefore, by considering only top correlated ones we lose some information.

In the generalized approach we weighted users rating on a particular movie based on the correlation of that movie genre to the target movie genre. This approach produces slightly better results than the standard approach and shows that the basic idea of using semantic information as a factor to distinguish user context can work. Our results suggest that genre

\(^1\)www.imdb.com
by itself is not sufficient to distinguish different contexts. In our future work we intend to take into account other semantics such as actors, directors and etc.

5 Conclusions

In this paper we presented an algorithmic framework to use semantic cues as a key to distinguish context and we proposed different algorithms to use genre as one semantic domain knowledge beside the user rating data in a movie recommender system. We presented an approach to find user ratings for different contexts rather than ratings for items and we used correlation between different contexts as a weight factor to find user nearest neighbors. Our results suggest that genre by itself is not sufficient to distinguish different contexts. We believe that the ideas presented within this paper provide a new direction in recommender system research and will lead to new techniques and improved recommender performance. In future work we are going to take more semantic features to extract user context. How to weight different semantic information to extract user context and how to develop heuristics for identifying contextual interactions is the key research questions which we are working in our future work.

References


Using this information, a recommendation system can be constructed to generate full processing chains in a Digital Audio Workstation (DAW). Discover the world's research. 15+ million members. 118+ million publications. 700k+ research projects. Join for free. Figures - uploaded by Nicholas Jillings. Author content. Previous studies have investigated the automation of parameter settings based on external semantic cues. In this study, we provide an analysis of the ways in which participants apply full processing chains to musical audio. We identify trends in audio effect usage as a function of instrument type and descriptive terms, and show that processing chain usage acts as an effective way of organising timbral adjectives in low-dimensional space. Latent semantic models, such as latent semantic analysis (LSA) and its extensions, are able to map a query to its relevant documents at the semantic level (e.g.,[2]). However, most latent semantic models still view a query (or a document) as a bag of words. Therefore, they are not effective in capturing fine-grained contextual structures for information retrieval. Compared with DSSM, C-DSSM has a convolutional layer that projects each word within a context window to a local contextual feature vector. Semantically similar words-within-context are projected to vectors that are close to each other in the contextual feature space. In psychology, contextual cueing refers to a form of visual search facilitation which describe targets appearing in repeated configurations are detected more quickly. The contextual cueing effect is a learning phenomenon where repeated exposure to a specific arrangement of target and distractor items leads to progressively more efficient search. In a global context, massive amounts of sensory input are received on a daily basis that would require an unrealistic amount of cognitive resources for it all.