Modeling and Learning Context-Aware Recommendation Scenarios using Tensor Decomposition

Hendrik Wermser
Technische Universität München
Garching bei München, Germany
Email: wermser@cs.tum.edu

Achim Rettinger
Karlsruhe Institute of Technology
Karlsruhe, Germany
Email: rettinger@kit.edu

Volker Tresp
Siemens Corporate Technologies
Munich, Germany
Email: volker.tresp@siemens.com

***Abstract*—The task of recommending items, like movies, to users is a core feature of many social networks. Standard approaches either use item or user similarity to suggest the next items users might be interested in. Recently, multivariate models like matrix factorization have become popular to combine the advantages of both perspectives. In addition, extensions have been proposed to capture the dynamics of user interests over time, like trends or recurrent user needs. While offering good predictive performance, so far those models do not exploit possibly available rich semantic context. Typically, only one implicit feature, like user ratings, is tracked to give personalized recommendations. However, with semantic data sources, like linked data, wealthy background knowledge becomes available that could be leveraged to improve predictive performance. We argue, that a more flexible framework is needed to model and learn a greater class of recommendation scenarios where rich context is available. Thus, we propose a generic approach which generalizes state-of-the-art methods based on pairwise interaction tensor factorization by leveraging arbitrary background knowledge related to the recommendation situation. Our experiments on streamed semantic data from a social network show that by adding varying sets of context - like user information, sequential information or time information - the ranking of potential items can be personalized and the predictive performance can be improved.**

I. Introduction

The availability of large semantic data sources is constantly increasing. Most notably is the development around the Linked Data[1] (LD) initiative which interlinks data sources via dereferenceable Unique Resource Identifiers (URIs) on the Web. This semantic data can provide valuable context information for applications. It is particularly useful in supporting inductive reasoning and Machine Learning (ML) applications. In LD applications, typically subsets of the LD-cloud are retrieved in repositories and some form of reasoning is applied to infer implicit facts. However, one can certainly assume that there are a huge number of likely facts which are neither known to be true nor can be derived by deductive reasoning. The approach taken in this work is to estimate the truth values of facts by exploiting patterns in the data, i.e. to rank predictions by their likelihood and apply this ranking to recommendation tasks.

[1]http://linkeddata.org/

Recommender systems, such as movie recommenders or tag recommenders, play an increasingly important role in today's social networks on the world wide web. They are for instance used to increase sales on eCommerce sites (see e.g. Amazon[2]) and at the same time facilitate the discovery of relevant content for the user (see e.g. YouTube[3]). In such scenarios it cannot be known for certain that a user will like an item unless the user specifies it explicitly. Thus, most state-of-the-art approaches use ML methods, e.g. to find items similar to the ones that a user liked before. The data usually consists of one relation specifying an interaction of users with items. This interaction is captured by one observable feature, e.g. $likes$, $buys$, $rates$, $views$, etc. Then, the goal is to suggest the most probable items the user might also $like$, buy, $rate$ or $view$ in the future.

The most successful model classes for this task perform a multivariate prediction of the unknown relation-instances. Especially factorization methods based on matrix or tensor decomposition have performed best on various benchmarks, like the Netflix challenge[4] [1]. It is quite apparent that contextual information has the potential to improve the prediction accuracy. An example for considering additional context information is the Pairwise Interaction Tensor Factorization [2] which still exploits only a minimum of semantic information. In contrast, we generalize that approach to a greater class of recommendation tasks where arbitrary contextual information about the recommendation situation can be incorporated. More specifically, any situations where an entity is to be recommended given an arbitrary number of binary context information can be covered. The model subsumes many other factorization models, such as standard matrix factorization (MF). Hence, we call our approach Context-Aware Recommendation Tensor Decomposition (CARTD).

Our main contributions are as follows: We present an intuitive modeling framework for a wide class of semantic recommendation tasks which subsumes popular approaches like [3] (see Sec. II). In Sec. II we also show how to utilize

[2]http://www.amazon.com/
[3]http://www.youtube.com/
[4]http://www.netflixprize.com/

978-0-7695-4375-8/11 $26.00 © 2011 IEEE
DOI 10.1109/ASONAM.2011.56

context knowledge in recommendation tasks in order to model a large class of possible recommendations scenarios. Next, we derive a method to decompose and optimize the resulting tensor according to a modified version of the Bayesian Personalized Ranking (BPR) criterion, which was introduced in [4] (see Sec. III). In Sec. IV we provide examples on how common recommendation scenarios can be modeled using CARTD. We demonstrate the effectiveness of our approach on real world data from a social network which provides a stream of user recommendations accompanied by additional contextual clues. Our results show that processing this kind of semantic data can improve predictive performance (see Sec. V). Finally we report on related work (see Sec. VI) and conclude in Sec. VII.

II. Utilizing Context Knowledge in Recommendation Tasks

The task in recommender systems is to predict a list of most likely entities, e.g. movies, a user might be interested in. For example a movie recommender would recommend a list of movies. A simple approach used in many applications is the 'most popular' recommender which recommends the same set of most popular movies to every user. Thus, this recommender is not 'personalized' and does not incorporate any context information. Another standard approach, known as Collaborative Filtering (CF), uses only correlation in preference patterns to predict a list of most relevant items [5], [6].

However, in many cases additional information is available that could be utilized in order to improve recommendations. With the Resource Description Framework (RDF) and its broad uptake, e.g. within the Linked Data (LD) initiative, an increasing amount of such graph-structured semantically enriched data can be fetched from different data sources.

In this paper we will introduce the Context-Aware Recommendation Tensor Decomposition (CARTD), which makes use of such context information in a generic way. We focus on context information as a set of entities that is not deterministically dependent on the entity to be recommended but depends on the situation where the recommendation needs to be performed. From the perspective of a movie, context information like this would be the user who watched it, the location where it was watched, the time when it was watched, the movie that was watched before by the user, etc. Deterministically dependent context information would be the genre of the movie (because it deterministically depends on the movie) or the age of the user (because the user is already in the set and the age deterministically depends on the user). Information deterministically dependent like that is not focus of this work. The distinction is illustrated in Figure 1.

A. Formalization

More formally, the task of item recommendation in our scenario is to provide the user with a ranked list of entities from a set $\mathbf{Entity}$, where the goal is to rank the most interesting item in the current situation at the top of the list. Additionally n sets of contexts are given, each belonging to a specific context type, i.e. $\mathbf{Context_i}$, where $i \in 1, \ldots, n$. In the case of standard CF movie recommendation $\mathbf{Entity}$ would be $\mathbf{Movies}$ and there would only be one context, namely

$$\mathbf{Context_1} := \mathbf{User}$$

But one could imagine that also

$$\mathbf{Context_2} := \mathbf{MonthOfYear}$$

which describes the month of the year when the recommendation is supposed to be given could be a correlated factor which might improve the recommendations. Clearly, $\mathbf{MonthOfYear}$ conforms to our definition of not deterministically dependent context information since it neither depends on the movie nor on the user but only on the recommendation relation (see Fig. 1). We give further examples and illustrations of possible choices for these sets in section IV.

The formal task, is to recommend a list of entities $(e_1, e_2, \ldots, e_m)$, where $e_i \in \mathbf{Entity}$, given some context information $(c_1, c_2, \ldots, c_n)$, where $c_i \in \mathbf{Context_i}$. In order to train our model we assume a set D of training data to be given, such that

$$D \subset \mathbf{Entity} \times \mathbf{Context_1} \times \mathbf{Context_2} \times \cdots \times \mathbf{Context_n}$$

We then define multisets (sets which can hold elements more than once) for the individual relations between $\mathbf{Entity}$ and the respective context $\mathbf{Context_i}$:

$$\mathbf{ContextRelation_i} := \left(\left\{ (e, c_i) \;\middle|\; \begin{array}{l} \exists c_1, \ldots, \exists c_{i-1}, \exists c_{i+1}, \ldots, \exists c_n : \\ (e, c1, \ldots, c_{i-1}, c_i, c_{i+1}, \ldots, c_n) \in D \end{array} \right\}, \#_i \right)$$

where $\#_i$ denotes a function from the set (the first element of the tuple) to $\mathbb{N}$, indicating the number of occurrences of (e, c_i) in the training set D for a specific c_i and with all other contexts not considered fixed. Applied to our movie recommendation scenario, this counts e.g. the number of times a movie was watched at a certain location.

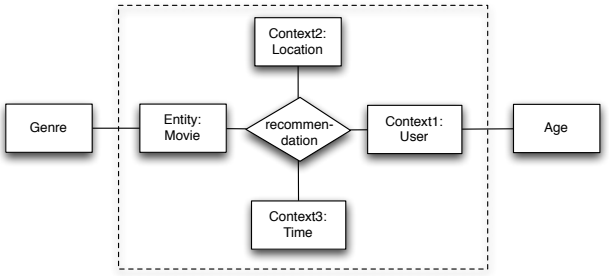

Fig. 1. ER-diagram showing an example of a context-aware recommendation scenario. The dashed rectangle contains the not deterministically dependent context and the elements outside the rectangle illustrate deterministically dependent context.

B. Ranking

So far, we have stated that we would like to recommend a list of entities given certain context information. Following [4]

we now interpret a list of recommended entities as a ranking over these entities. Then

$$>_{c_1,c_2,c_3,\ldots,c_n} \subset \mathbf{Entity}^2$$

is a ranking over all entities $e \in \mathbf{Entity}$ given the respective context information $(c_1, c_2, c_3, \ldots, c_n)$, $c_i \in \mathbf{Context_i}$. We abbreviate this ranking as $>_{\{c\}}$. Usually only the top of the list is interesting for real world applications. In a movie recommender application for example only the 5 highest ranked entities might be presented to the user.

We now want to optimize the posterior probability of some set of parameters given a ranking, i.e.

$$p(\Theta \mid >_{\{c\}}) \propto p(>_{\{c\}} \mid \Theta) p(\Theta)$$

as described in more detail in [4]. In order to optimize for a ranking specifically we define the augmented training data as follows:

$$D_A := \left\{ (e, c_1, \ldots, c_n, j) \;\middle|\; \begin{array}{c} (e, c_1, \ldots, c_n) \in D \\ \wedge\, \forall i : \#_i(e, c_i) > \#_i(j, c_i) \end{array} \right\}$$

That means we define the training data to include a second entity which less often appears in all respective context relations. In the example from above this would mean the additional movie in the training data was less often watched by the respective user as well as less often watched in the respective month.

If we assume that the contexts are independent of each other and also that the ordering of each pair of items (e, j), where $e, j \in \mathbf{Entity}$, for a given set of contexts $\{c\}$ is independent of the ordering of all other pairs, we obtain the maximum a posteriori (MAP) estimator

$$\begin{aligned} \hat{\Theta}_{MAP} &= \arg\max_{\Theta} \prod_{\{D_A\}} p(>_{\{c\}} \mid \Theta) \\ &= \arg\max_{\Theta} \prod_{(e,c_1,\ldots,c_n,j) \in D_A} p(e >_{\{c\}} j \mid \Theta) p(\Theta) \end{aligned}$$

of Θ .

In order to extend the BPR criterion [4] to our scenario we need to further define $p(e >_{\{c\}} j|\Theta)$. Therefore, item e is ranked higher than item j according to $>_{\{c\}}$ if and only if some $\hat{x}_{\{c\},e} >_{\mathbb{R}} \hat{x}_{\{c\},j}$, i.e.

$$e >_{\{c\}} j :\Leftrightarrow \hat{x}_{\{c\},e} >_{\mathbb{R}} \hat{x}_{\{c\},j}$$

and

$$\begin{aligned} p(e >_{\{c\}} j|\Theta) &= p(\hat{x}_{\{c\},e} >_{\mathbb{R}} \hat{x}_{\{c\},j}) \\ &= p(\hat{x}_{\{c\},e} - \hat{x}_{\{c\},j} >_{\mathbb{R}} 0) \\ &:= \sigma(\hat{x}_{\{c\},e} - \hat{x}_{\{c\},j}) \end{aligned}$$

where the $\hat{x}_{\{c\},e}$ depends on Θ, i.e. $\hat{x}_{\{c\},e} = \hat{x}_{\{c\},e}(\Theta)$ and where σ is the logistic sigmoid $\sigma(z) := \frac{z}{1+e^{-z}}$.

Furthermore, we use a normally distributed prior over the parameters, i.e. $p(\Theta) \sim (0, \Sigma_\Theta)$ where we choose $\Sigma_\Theta = \lambda_\Theta I$ in order to reduce the number of hyper-parameters. Ultimately our goal is to estimate $\hat{\Theta}$ via the log likelihood:

$$\arg\max_{\Theta} \sum_{(e,c_1,\ldots,c_n,j) \in D_A} \ln \sigma(\hat{x}_{u,e} - \hat{x}_{u,j}) - \lambda_\Theta ||\Theta||^2_{\text{Fro}},$$

where $||\cdot||_{\text{Fro}}$ denotes the Frobenius norm of a matrix.

Note that this approach is generic and allows for an arbitrary parametric model to be optimized according to this ranking criterion. Only the $\hat{x}_{\{c\},e}$ have to be defined by the model. We will propose Context-Aware Recommendation Tensor Decomposition (CARTD) as our particular model to be optimized (cf. section III). Similarly to [4] we use bootstrapped stochastic gradient descent as the optimization procedure.

C. Interface

We have above defined the sets that the model works on. These sets can be interpreted as an "interface" between the data and the model. In order to use the model for recommendation tasks the problem has to be representable in form of these sets in a meaningful manner. As we will see in section IV, a great set of problems can be formulated in such a way.

III. Context-Aware Recommendation Tensor Decomposition

Our goal is to model an entity together with multiple contexts. The standard matrix decomposition models popular for Collaborative Filtering approaches can be generalized to higher-order relations in the form of tensor factorizations (TF) [7]. In this section we show how context-aware recommendation tensors can be decomposed and optimized efficiently.

A. Decomposing Context-Aware Recommendation Tensors

Well-studied TFs include the Tucker decomposition as well as Parallel Factor Analysis (PARAFAC) [7]. Since the ratio of relations existent in the training data to possible relations is extremely small (meaning the resulting tensor would be extremely sparse) we use a special case of PARAFAC that reduces to modeling pairwise interactions between the entity and the corresponding context (cf. [2]):

$$\hat{x}_{\{c\},e} := \sum_{i=1}^{n} \left\langle v_e^{E,C_i}, v_{c_i}^{C_i,E} \right\rangle,$$

so that an entry $x_{\{c\},e}$ is reconstructed as $\hat{x}_{\{c\},e}$ from the respective factorization matrices $\Theta = \{V^{E,C_1}, V^{C_1,E}, V^{E,C_2}, V^{C_2,E}, \ldots, V^{E,C_n}, V^{C_n,E}\}$. Here v_e^{E,C_i} denotes the eth column of the factorization matrix V^{E,C_i} and analogously with the other matrices. The entries in the factorization matrices Θ then are the parameters at our disposal for learning the rankings. Obviously each pair of matrices has to have corresponding inner dimensionality, i.e. if $V^{E,C_i} \in \mathbb{R}^{|E| \times k_{E,C_i}}$ then $V^{C_i,E} \in \mathbb{R}^{|C_i| \times k_{E,C_i}}$.

B. Optimizing Context-Aware Recommendation Tensors

Finally, we need to optimize CARTD by the BPR criterion. Usual stochastic gradient descent will cycle trough the training data and after every instance update the parameters accordingly. The gradients of BPR with respect to a model parameter θ and an instance of the augmented training data $(e, c_1, c_2, \ldots, c_n, j) \in D_A$ are

$$\begin{aligned}&\frac{\partial}{\partial\theta}\left(\ln\sigma(\hat{x}_{\{c\},e} - \hat{x}_{\{c\},j}) - \lambda_\theta\theta^2\right)\\ &= \left(1 - \sigma(\hat{x}_{\{c\},e} - \hat{x}_{\{c\},j})\right)\frac{\partial}{\partial\theta}(\hat{x}_{\{c\},e} - \hat{x}_{\{c\},j}) - 2\lambda_\theta\theta,\end{aligned}$$

which can be calculated analytically for the CARTD and matrix factorization models. Instead of cycling through the training data bootstrapped stochastic gradient descent optimization is applied, i.e. $(e, c_1, c_2, \ldots, c_n, j) \in D_A$ is sampled uniformly at random from the training data. As has been shown in [4] this dramatically improves the convergence speed of the algorithm.

IV. Encoding Common Context-Aware Recommendation Scenarios

As mentioned, CARTD suits itself best to tasks where binary context information can be exploited for the training phase and is also available during prediction/recommendation. We will show in the following that many common recommendation tasks fall under this definition.

A. Time Information

In commercial recommender settings timestamps of the time of purchase are usually recorded. These can be utilized as additional context in several ways. In our evaluation we used these timestamps to generate two contexts for a purchase (time and sequence information). Here we consider the obvious case of using the timestamps to define the hour of the day, the day of the week or the month of the year as context when the item was purchased. We define the generic sets of our model as follows:

$$\begin{aligned}\mathbf{Entity} &:= I && \text{all the available items}\\ \mathbf{Context_1} &:= U && \text{all the users}\\ \mathbf{Context_2} &:= M && \text{the months a of year}\end{aligned}$$

The context relation sets are constructed as stated in section II-A:

$$\mathbf{ContextRel_1} := (\{(i,u) \mid \exists m \in M : (i,u,m) \in D\}, \#_u)$$

$$\mathbf{ContextRel_2} := (\{(i,m) \mid \exists u \in U : (i,u,m) \in D\}, \#_m)$$

where D is the training data set and $\#_u$ indicates how often a user has bought a certain item and $\#_m$ indicates how often an item has been bought in the respective month.

B. Sequential information

In the previous section it was shown how to define sets for time information gained from purchase timestamps. Here we demonstrate how sequential information can be added. We have the sets

$$\begin{aligned}\mathbf{Entity} &:= I && \text{all the available items}\\ \mathbf{Context_1} &:= U && \text{all the users}\\ \mathbf{Context_2} &:= M && \text{the months a of year}\\ \mathbf{Context_3} &:= I && \text{all the available items (as last items)}\end{aligned}$$

$\mathbf{ContextRelation_3}$ can then be used to model interaction between an item and the item previously purchased and is defined as follows:

$$\mathbf{ContextRel_3} := (\{(i,l) \mid \exists u \in U, \exists l \in I : (i,u,m,l) \in D\}, \#_l)$$

with $\#_l$ now indicating how often an item has been purchased after another item (termed last item, because it was the last item before the current).

C. Location Information

Apart from context gained from time there is another common example for context information, namely location information. In this case then we have sets

$$\begin{aligned}\mathbf{Entity} &:= I && \text{all the available items}\\ \mathbf{Context_1} &:= U && \text{all the users}\\ \mathbf{Context_2} &:= L && \text{all the locations .}\end{aligned}$$

Here the location information is assumed to be given for discrete entities, i.e. *at home*, *at work*, *in a super market*, etc. for each purchase. Note, that it is straight forward to combine location context with the above mentioned time and sequence contexts. In general it can be expected that this would further improve recommendations.

D. Tag Recommendation

Tag recommendation is another related example. Here, tags get recommended while resources (e.g. text documents that can be tagged) and users provide context information. The sets of our model can for example be defined as follows:

$$\begin{aligned}\mathbf{Entity} &:= T && \text{all the available tags}\\ \mathbf{Context_1} &:= U && \text{all the users}\\ \mathbf{Context_2} &:= R && \text{all the resources}\end{aligned}$$

Available tags are tags that have been used before by other users. The context relation sets are defined accordingly. In [2] a very similar approach has been applied to this problem. Use cases like these motivate the definition of a generic model like CARTD which does not need to be redesigned depending on domain knowledge and the task at hand.

V. Empirical Evaluation

In this section we investigate how CARTD is applied to several combinations of possible contexts in a real world recommendation scenario.

A. Get Glue Dataset

Our real-world context-aware recommendation scenario is provided by the social network Glue[5]. Glue allows users to share the experience of navigating the Web; as any social network, it also allows users to connect to each other. Glue uses semantic recognition technologies to automatically identify books, music, movies, wines, stocks, movie stars and many other similar entities, and thus generates a continuous data stream of such objects. Users can observe every data stream and can receive recommendations from Glue about interesting discoveries by their friends. Both, the social network data and the real-time streams are accessible via a REST API[6]. For the purpose of our experiments we used a set of wrappers [8] that export Glue data in the form of Linked Data (LD).

We continuously sampled datasets through the Glue network's API. To ensure a focused analysis we gathered movie ratings only. We sampled from 50 to 200 items per user to experiment with different levels of data sparsity. Also, we restricted samples to positive feedback on items by users (leaving out the possible explicit negative feedback) in order to deal with a pure binary ratings matrix - a scenario that is very common in real world applications[7]. The specific dataset on which we report our results has 3,076 users and 9,707 movies and a single user gave up to 400 times positive feedback (i.e. *likes*) on different movies. On average a user gave feedback on 170 movies. The first date in the data lies in July 2006 and users gave ratings over an average time interval of 98 days. The median date in the data is 26th of October 2009. We pruned the data to the 3-core, meaning every item was rated at least 3 times and each user rated at least 3 items. The resulting training data is 98.21 % sparse, or in other words 98.21% of the entries of the resulting user-movie-rating matrix are zero entries representing missing/unknown information. Sequential information (see section IV-B) is inferred from the timestamps that are available as is the month in which the movie was rated (see section IV-A).

B. Evaluation

We report results on the dataset described above for two different recommendation settings. In the first scenario we split 5% of the data randomly as test data, without any preference for any particular entities or contexts. This evaluation is very generic and works for all the settings described in section IV. In the second scenario we explicitly split those movies as test data for which each user last gave positive feedback. This simulates a likely real world scenario where the next movie in a sequence of ratings is to be recommended. In both cases the remaining data is taken as the training data. Items that were in the training data do not appear in the predicted ranking, as they are known already (we don't want to recommend known movies). This could be different for other applications, where repeated ratings or purchases are possible (e.g. online shopping).

[5]http://getglue.com
[6]http://getglue.com/api
[7]e.g. the "like button" available on facebook

For each setting the ranking performance is evaluated by two accuracy measures, the standard machine learning measure *Area Under the ROC Curve* (AUC) and the HitRatio measure. For the task of ranking, the AUC reduces to $AUC(i) = \frac{|I|-pos(i)}{|I|}$, where $pos(i)$ gives the position of item i in the ranking, 0 being the best position and $|I|$ the worst. We report the average AUC over all test items. The HitRatio determines whether the test movie is in the top k of the given ranking. We then report the percentage of test movies for which this was the case. We chose $k = 10$, as it seems reasonable that a real world application might present its users a list of 10 movies.

The particular combinations of context information that were used in the different scenarios are described in the following and depicted in Fig. 2. Note that the model (as described in section III) is the same for all combinations, what is altered is the context information taken into account.

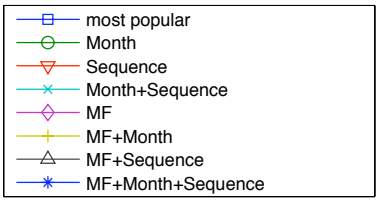

Fig. 2. Different approaches compared in the experiments.

Month refers to "month of year" information as described in section IV-A, gained from the timestamps available in the Get Glue datasets. The month context set obviously consists of 12 different contexts only, compared to 3,076 users and 9,707 movies. Thus the month is the context and the item-month matrix is factorized.

Sequence refers to the last rated item as context related to the current item. The sequence context set consists of all 9,707 movies. Thus the only context is the last item and the item-lastItem matrix is factorized.

MF refers to standard Matrix Factorization (i.e. the user is the context and as in most collaborative filtering approaches the item-user matrix is factorized). In our case the Matrix Factorization is optimized by the BPR criterion (as described in section II-B).

Combinations The other scenarios are combinations of the three contexts mentioned above. Thus, for example **MF+Month** indicates that not only the user was taken as context but also the month of the year in which the rating took place (this would be exactly the use case given in section IV-A). The factorizations above are special cases where only one single context is considered.

most popular refers to the most popular baseline which ignores any context and simply recommends movies in order of the number of positive feedback they received.

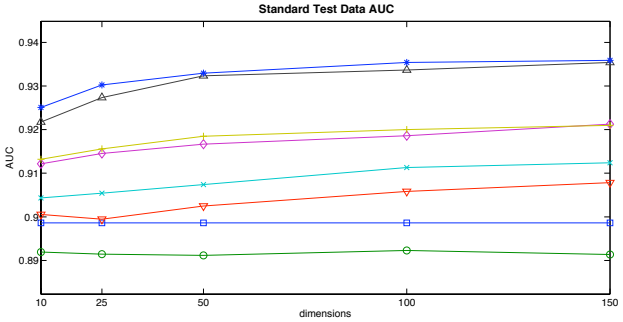

Fig. 3. Comparison of CARTD for different context information (see fig. 2) on standard test data. The AUC is reported for inner factorization dimensions 10, 25, 50, 100 and 150.

Figure 3 shows that for standard test data evaluated via the AUC measure every context information utilized by the CARTD model apart from the **Month** alone, improves performance compared to the most popular baseline. Furthermore each additionally utilized context adds to the performance, although the **Month** context is able to improve the performance of the **MF** and **MF+Sequence** context settings only slightly.

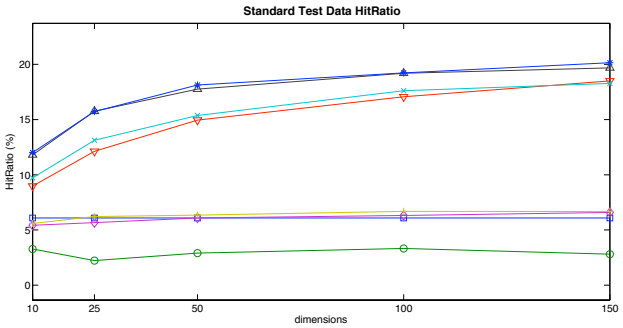

Fig. 4. Comparison of CARTD for different context information (see fig. 2) on standard test data. The HitRatio ($k = 10$) is reported for inner factorization dimensions 10, 25, 50, 100 and 150.

In figure 4 the HitRatio on the standard test data is reported. Intuitively the **most popular** baseline should perform very well for the HitRatio measure, because this baseline simply recommends the most liked movies in every context and many users will of course have *liked* exactly such a movie. This is indeed the case, as even standard matrix factorization does not significantly outperform the baseline for this measure. The **Month** context information alone receives very low scores in this setting and is not able to gain a lot in combination with the other contexts either.

In figure 5 we report results of the context information utilized by CARTD on the most current movie feedback test data. This means for each user the most recently rated movie is used as test movie and should appear in a high position in the ranking given the context the user rated this movie in. Please note, that **Month+Sequence** performs similarly to the matrix factorization (**MF**), although the contexts used are fairly different. Similarly for **MF+Month** and **MF+Sequence**.

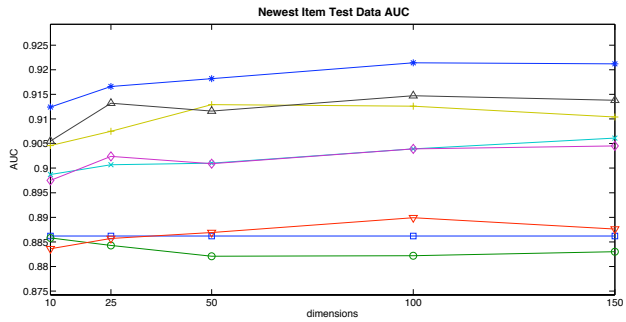

Fig. 5. Comparison of CARTD for different context information (see fig. 2) on most current movie feedback test data. The AUC is reported for inner factorization dimensions 10, 25, 50, 100 and 150. For a legend see fig. 2.

Not surprisingly, the combination of all the available context information then outperforms all other settings.

Figure 6 shows the HitRatio results for the most current movie feedback test data. Again, the percentage of test movies that appeared in the top 10 for the particular context is recorded. Once more, the **most popular** baseline performs very well on this measure. It clearly outperforms the **Month** context, the matrix factorization **MF** and even their combination. As should be expected the sequence information is especially helpful in recommending the most current movie. Thus, the settings involving the sequence perform especially well, with the combination of all contexts again outperforming all simpler combinations.

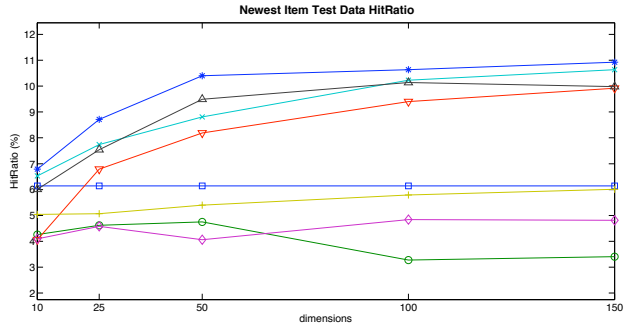

Fig. 6. Comparison of CARTD for different context information (see fig. 2) on most current movie feedback test data (details see text). The HitRatio ($k = 10$) is reported for inner factorization dimensions 10, 25, 50, 100 and 150. For a legend see fig. 2.

VI. Related Work

As mentioned before there are two main research directions in ML which try to expand the common feature vector representation to more complex representations as desired for the context-aware recommendation task in social networks.

First, there is the attempt to use semantic data representations - like graph-based (e.g., RDF(S)) or logic-based (e.g., OWL) formalisms - as the input to ML algorithms to capture the context for e.g., the recommendation scenario [9]. The ML

area concerned with those kind of statistical models is known as Statistical Relational Learning (SRL) [10]. The range of techniques deployed in SRL is wide. This includes for instance kernel methods for structured representations [11], multivariate prediction models [12], [13], relational graphical models (e.g., *probabilistic relational models* [14] or *infinite hidden relational models* [15]) and first-order probabilistic learning approaches (e.g., *bayesian logic programs* [16] or *markov logic networks* [17]). For the task of item recommendation multivariate prediction models like matrix decompositions have been especially successful [1]. However, the mentioned approaches are not capable of intuitively modeling context-aware recommendation scenarios by integrating arbitrary contexts like sequential data. The approach proposed here is based on tensor decomposition and offers a flexible framework for modeling and learning in such situations.

Second, there exist ML approaches that are concerned mainly with the statistical analysis of ordered sequences of observations where common tasks are a) classification of observations, b) prediction of future observations and analysis of the temporal dynamics of the stream. Common applications are concerned with anomaly detection, (long term vs. short term) trend detection, changes in the regularities in the data stream and evolution of social networks.

In our case we expect user preferences in a social network to change over time. For example, in a movie recommender system, users may change their preferred genre or adopt a new viewpoint on an actor or director. In addition, they may alter the assessment of their feedback. Besides that general trends might influence the preferences of users in certain time frames.

Such dynamics are for instance investigated in [18]. Here, a matrix factorization approach is applied in small time steps on movie rating sequences and the authors show, that there is a significant change in user preference over time. However, they do not provide an integrated learning algorithm for the dynamics and don't handle additional relations or features besides the ratings.

Just recently a combination of Markov chains with matrix factorization was proposed by [3] and applied to next-basket recommendation in eCommerce applications. In their work the authors introduce a generic way to handle the temporal information by factorizing a transition cube. Also there has been work on context-aware collaborative filtering via Tensor Factorization in [19].The approach proposed here is an extension to [3] and generalizes it to a larger class of recommendation scenarios which can model any context like spatial and temporal data related to the recommendation situation.

However, in the application presented here - besides the dynamics - we are faced with the added complexity of ML in the context of relational domains. The user might be involved in a number of relationships such as interests in movies, books and other items, the friendship relationships and the follows relationship. In addition, relations of items can be modeled, like genre or director of a movie.

Another related work is introduced by [20]. They study event-based networks such as email traffic, telephone calls and research publications (co-authorship events), and predict future event co-participation of entities. In particular, they answer the question given the previous event data, will a pair of entities co-participate in at least one event in a future specified time interval. This is modeled as a classification problem with two classes co-participation and not co-participation. The features are defined with respect to the entity pair and include two categories: network-based (e.g., neighborhood features and shortest weighted path features) and entity-based (e.g., entity attribute similarity measures and geographic proximity). The method is applied to analyze a publication event network spanning 6-year time period, with 128,000 papers (events), and 310,000 authors.

Both examples of time-varying link prediction models can not be directly applied to sequential and personalized item recommendation. Especially the model of [20] seems to be specifically designed for their task. In contrast, we try to propose a more generic model to predict relational temporal data.

VII. Conclusion

In this paper we proposed a generalization of a state of the art matrix factorization technique in order to extend its applicability to a larger class of context-aware recommendation scenarios like the ones needed in social networks. This kind of rich spatial and temporal information is becoming increasingly available on the Semantic Web and - as our experiments show - can be leveraged to improve predictive performance. SW data typically is extremely sparse - in our experiments 99.58% facts are missing/unkown for the relation to predict. We counter this, by exploiting one of the advantages of Linked Data (LD), namely the easily accessible context. By extracting additional related information we can reduce the sparsity of the transition cube which contains the recommendations per user and item. While our approach can be applied intuitively to many recommendation situations, in our dataset this is achieved by modeling recommendations of a user while considering the movies the user was interested in before or the time when movies were watched. Our experimental results on a large real world data set suggest, that our proposed generalization offers modeling capabilities for various recommendation scenarios and can improve predictive performance. In general, our application demonstrates, that LD can provide important additional context to make Machine Learning (ML) problems more context aware and efficient.

Our future work is concerned with providing a detailed theoretical analysis of our approach and ultimately extending the algorithm to an arbitrary LD graph.

REFERENCES

[1] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2008, pp. 426–434.

[2] S. Rendle and L. Schmidt-Thieme, "Pairwise interaction tensor factorization for personalized tag recommendation," in *WSDM 2010: Proceedings of the 2010 ACM International Conference on Web Search and Data Mining*. ACM, 2010.

[3] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing personalized Markov chains for next-basket recommendation," in *Proceedings of the 19th international conference on World wide web*. ACM, 2010, pp. 811–820.

[4] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: bayesian personalized ranking from implicit feedback," in *UAI '09: Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, 2009.

[5] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-Item collaborative filtering," *IEEE Internet Computing*, vol. 7, no. 1, pp. 76–80, 2003.

[6] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *IEEE International Conference on Data Mining (ICDM 2008)*, 2008, pp. 263–272.

[7] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, September 2009.

[8] D. F. Barbieri, D. Braga, S. Ceri, E. D. Valle, and M. Grossniklaus, "Continuous queries and real-time analysis of social semantic data with c-sparql," in *Second Workshop on Social Data on the Web (SDoW2009)*, 2009. [Online]. Available: http://CEUR-WS.org/Vol-520/paper02.pdf

[9] A. Rettinger, M. Nickles, and V. Tresp, "Statistical relational learning with formal ontologies," in *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases, ECML/PKDD2009*. Springer, 2009, pp. 286–301.

[10] L. Getoor and B. Taskar, Eds., *Introduction to Statistical Relational Learning*. MIT Press, 2007.

[11] N. Fanizzi, C. d'Amato, and F. Esposito, "Learning with kernels in description logics," in *Proceedings of the 18th International Conference on Inductive Logic Programming, ILP2008*, ser. LNAI, F. Železný and N. Lavrač, Eds., vol. 5194. Springer, 2008, pp. 210–225.

[12] C. Lippert, Y. Huang, S. H. Weber, V. Tresp, M. Schubert, and H.-P. Kriegel, "Relation prediction in multi-relational domains using matrix factorization," Siemens, Tech. Rep., 2008.

[13] Y. Huang, V. Tresp, M. Bundschus, and A. Rettinger, "Multivariate structured prediction for learning on semantic web," *Proc. of the 20th International Conference on Inductive Logic Programming (ILP 2010)*, 2010.

[14] D. Koller and A. Pfeffer, "Probabilistic frame-based systems," in *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 1998.

[15] Z. Xu, V. Tresp, A. Rettinger, and K. Kersting, "Social network mining with nonparametric relational models," in *Advances in Social Network Mining and Analysis*, ser. LNCS, H. Zhang, M. Smith, L. Giles, and J. Yen, Eds. Springer, 2009.

[16] K. Kersting and L. De Raedt, "Bayesian logic programs," Albert-Ludwigs University at Freiburg, Tech. Rep., 2001.

[17] M. Richardson and P. Domingos, "Markov logic networks," *Journal of Machine Learning Research*, vol. 62, no. 1-2, pp. 107–136, 2006.

[18] Y. Koren, "Collaborative filtering with temporal dynamics," *Communications of the ACM*, vol. 53, no. 4, pp. 89–97, 2010.

[19] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver, "Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering," in *Proceedings of the fourth ACM conference on Recommender systems*, ser. RecSys '10. New York, NY, USA: ACM, 2010, pp. 79–86. [Online]. Available: http://doi.acm.org/10.1145/1864708.1864727

[20] J. O'Madadhain, J. Hutchins, and P. Smyth, "Prediction and ranking algorithms for event-based network data," *ACM SIGKDD Explorations Newsletter*, vol. 7, no. 2, p. 30, 2005.