Noname manuscript No.
(will be inserted by the editor)

Context-aware Tensor Decomposition for Relation Prediction in Social Networks

Achim Rettinger · Hendrik Wermser · Yi Huang · Volker Tresp

Received: date / Accepted: date

Abstract An important task in network modeling is the prediction of relationships between classes of objects, such as friendship between persons, preferences of users for items, or the influence of genes on diseases. Factorizing approaches have proven effective in the modeling of these types of relations. If only a single binary relation is of interest, matrix factorization is typically applied. For ternary relations tensor factorization has become popular. A typical application of tensor factorization would concerns the temporal development of the relationships between objects. There are applications, where models with n-ary relations with $n > 3$ need to be considered, which is the topic of this paper. These models permit the inclusion of context information that is relevant for relation prediction. Unfortunately, the straightforward application of higher-order tensor models becomes problematic, due to the sparsity of the data and due to the complexity of the computations. In this paper, we discuss two different approaches that both simplify the higher-order tensors using coupled low-order factorization models. While the first approach, the Context-Aware Recommendation Tensor Decomposition (CARTD), proposes an efficient optimization criterion and decomposition method, the second approach, the Context-Aware Regularized Singular Value Decomposition (CRSVD), introduces a generative probabilistic model and aims at reducing the dimensionality using independence assumptions in graphical models. In this article, we discuss both approaches and compare their ability to model contextual information. We test both models on a social network setting, where the task is to predict preferences based on existing preference patterns, based on the last item selected and based on attributes describing items and users. The experiments are performed using data from the GetGlue social network and the approach is evaluated on the ranking quality of predicted relations. The results indicate that the CARTD is superior in predicting overall rankings for relations, whereas the CRSVD is superior when one is only interested in predicting the top-ranked relations.

Keywords Relation prediction · Tensor matrix decomposition · Graphical model · Recommendation · Social media analysis

Achim Rettinger
Karlsruhe Institute of Technology, Karlsruhe, Germany
E-mail: rettinger@kit.edu

Hendrik Wermser
Technische Universität München, Munich, Germany
E-mail: wermser@cs.tum.edu

Yi Huang
Siemens AG, Corporate Technology, Munich, Germany
E-mail: yihuang@siemens.com

Volker Tresp
Siemens AG, Corporate Technology, Munich, Germany
E-mail: volker.tresp@siemens.com

1 Introduction

In this paper we discuss methods for predicting the existence of a relation between two or more entities. Examples would be relations describing the interest of a user for items, e.g., *watches(User, Movie)* and friendship relations in a social network, e.g., *isFriendsWith(PersonA, PersonB)*. Relation prediction has been used in various settings to improve user experience on the Web, recommending for example products, tags or news to the user. Initially, the problem was formulated as collectively labeled collaborative filtering. Here, recommendations were generated by comparing users' preference patterns and in this way identifying users with comparable preferences [Linden et al (2003)]. Currently, the best performing techniques for collaborative filtering are based on matrix factorizations, ranging from standard singular value decomposition (SVD) to a multitude of related approaches, including ones that capture temporal dynamics [Koren (2009)]. One prominent commercial example is the movie-rental platform Netflix[1] which uses an approach based on matrix factorization. The particular solution was the the winner of an open competition for the best collaborative filtering algorithm to predict user ratings for films, based on previous ratings.

Current methods utilize little additional context information that is available during recommendation, although context information has great potential to improve prediction accuracy. As an example, let's consider the relation *watches(User, Movie, LastMovieWatchedByUser, Month)* which says that a user watches a movie in a given month and where we also have information about the last movie that the user has watched. Such a relation can be modeled by a $n = 4$-way tensor which would give us, after reconstruction and normalization, $\hat{P}(\text{User, Movie, LastMovieWatchedByUser, Month})$. Information regarded as context may include, e.g., location information, time information and access device information. For example, in a social network setting advertisements that are relevant to a user on a mobile device (such as a smartphone) may be of less relevance to that same user when at home on the stationary computer. Also, a user may be more interested in recommendations of friends from a certain area when she actually is in this area. These types of context sensitivity are typical for recommendation settings as diverse as movie recommendation, music recommendation, news recommendation, tag recommendation, friend recommendation, and follower recommendation.

Matrix factorizations are most suitable for modeling binary relationships and previous work has shown that the power of factorization approaches can be transported to ternary relationships by employing tensor factorization. The task investigated in this paper is to extend factorization approaches to include contextual information which leads to tensors with more than three modes. Unfortunately, a straightforward application of tensor factorization to more than three modes is not feasible, due to data sparsity and to computational issues.

The two models presented in this paper simplify the tensor models by coupling simpler matrix models. The first approach is an extension to the work of Rendle et al (2010) towards higher-order models. The second approach is based on the idea that by properly normalizing the empirical tensor, it provides an empirical estimate of observing a relation, thus a normalized tensor model $P(a_1, \ldots, a_n)$. By having a probabilistic interpretation, we can now exploit independencies in the model to achieve efficient models. In particular, we reduce the model so that only probability distributions involving two variables need to be modeled.

The paper is organized as follows: In the next section, we discuss related work. Section 3 and Section 4 introduce the two models, respectively. Section 5 introduces the application used for evaluation and Section 6 discusses the experimental results. Section 7 presents our conclusions.

2 Related Work

Some of the leading approaches for predicting a single binary relationship are based on matrix completion that is calculated via matrix factorization. The latter can readily exploit structure in relational patterns. In particular, the winning entry to the Netflix competition used matrix completion approaches [Takacs et al (2007); Bell et al (2010); Cands and Recht (2008)]. In Yu et al (2006); Salakhutdinov and Mnih (2007) contextual information was included in matrix completion where a Gaussian noise model was employed which is more suitable for modeling continuous and ordinal quantities, such as a user score for a movie, than for the likelihood of the existence of a relation, as we are dealing with here. Also, those approaches often have difficulties in situations where only positive examples for a relation are available; they need to distinguish between true negatives (e.g., it is known that a user does not like a movie) and missing information (e.g., it is unknown if a user likes a particular movie). Bernoulli and Gaussian sampling approaches have been pursued in Chu et al (2006); Chu and Ghahramani (2009).

Tensors have recently found a lot of attention in modeling evolving networks. A recent overview on tensor models has been provided in Kolda and Bader (2009). Tensor models have been introduced to model ternary relations in Rendle et al (2010). The CARTD approach discussed in Section 3 is an extension of this work. In most cases, matrix and tensor factorization have been implemented in a deterministic interpretation, e.g., simply to complete a matrix or a tensor based on a low-rank approximation.

Our second approach is based on a decomposition of a probabilistic graphical model. Graphical models have a long history in expert systems and statistical modeling [Lauritzen (1996)]. Graphical models have also

[1] http://www.netflix.com

been applied to relational domains. Prominent examples are Probabilistic Relational Models [Koller and Pfeffer (1998); Getoor et al (2007)], Markov Logic Networks [Domingos and Richardson (2007)], and Infinite Hidden Relational Models [Kemp et al (2006); Xu et al (2006)]. Although being very general, the application of these models to a given relational domain might still be tricky: Probabilistic Relational Models require involved structural optimization, Markov Logic Networks depend on the availability of rule sets and logical expressions (approximately) valid in the domain and Infinite Hidden Relational Models require complex inference processes.

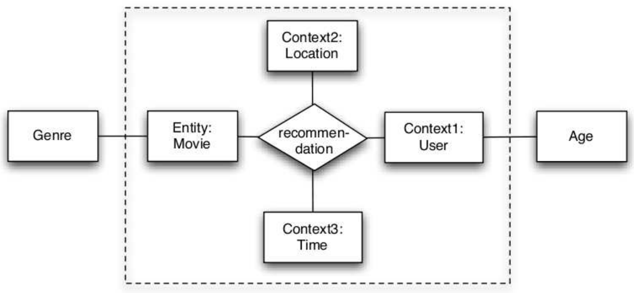


Fig. 1 ER-diagram showing an example of a context-aware recommendation scenario. The dashed rectangle contains the not deterministically dependent context and the elements outside the rectangle illustrate deterministically dependent context.

3 Modeling N-ary Relations using CARTD

In computer science, a tensor simply is a generalization of a matrix to more than two modes. Thus, an element in an n-ary tensor is addressed by N indices. Consider an n-ary relation $(a_1, \ldots, a_n)$, where a_i are finite discrete variables. For the model, we form a tensor with n modes. Now we simply start with a tensor with entries being zero and increment the corresponding entry by one, whenever the corresponding relation is observed. Typically the tensor is very sparse and in order to predict plausible new relations from the empirical tensor we generalize from observed entries to unobserved entries using tensor factorization. Kolda and Bader (2009) provides a recent overview of the leading tensor factorization approaches. The drawback is that with $n > 3$, the empirical tensor becomes increasingly sparse and the computational requirements become infeasible. Rendle et al (2010) showed how a ternary tensor can be described by a coupling of matrices leading to models with much better efficiency and better computational scalability. First, we describe the Context-Aware Recommendation Tensor Decomposition (CARTD), which is a direct generalization of this idea and was first presented in [Wermser et al (2011)].

3.1 Modeling N-Ary Tensors

To achieve meaningful recommendations via the CARTD method, we assume binary rating data is given together with context information that is dependent on the *recommendation* relation. This means the context information considered by CARTD is not determined by the entity to be recommended or by other context information that is already being used (see Fig. 1).

Formally, the task of item recommendation in our scenario is to provide the user with a ranked list of entities from a set **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 collaborative filtering (CF) movie recommendation **Entity** would be **Movies** and there would only be one context, namely

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

But one could imagine that also

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

which describes the month in which the recommendation is supposed to be given could be a correlated factor which might improve recommendations. Clearly, **MonthOfWatching** 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 (again, see Fig. 1).

For example,

(StarWars, Alice, Jan, Munich, E.T.) $\in T$

may be the binary information, stating that Alice has given positive feedback to the movie 'Star Wars' in January 2010, while being in Munich and after having given positive feedback to the movie E.T. before. We assume a user has given feedback to an entity either exactly once or never in one specific context. This however means that a user can rate the same movie in a different context again. This information can be represented by a binary 5-mode tensor T, which usually exhibits extreme sparsity, as in most recommendation settings only very few of all the possible combinations of contexts actually appear in observed data.

StarWars	Jan	Feb	March
Alice	1	0	0
Bob	1	1	0
Charlie	1	1	0

E.T.	Jan	Feb	March
Alice	0	0	0
Bob	1	0	0
Charlie	1	1	1

$$\begin{aligned}(\text{StarWars}, \text{E.T.}, \text{Alice}, \text{Jan}) &\in D\\ (\text{StarWars}, \text{E.T.}, \text{Bob}, \text{Feb}) &\in D\\ (\text{E.T.}, \text{StarWars}, \text{Charlie}, \text{March}) &\in D\end{aligned}$$

Fig. 2 Illustration of the original data, encoded in a ranking-suitable fashion in the set D. The tables correspond to feedback given on the movies 'StarWars' and 'E.T.' respectively. The resulting set D encodes a ranking over these two movies. The grey cells represent the context settings for which the respective movie is ranked above the other one. To simplify the illustration, we did not include information on location and on the previously rated movie.

3.2 Modeling Rankings

We encode the binary feedback information in a set

$$D = \{(i, j, c_1, c_2, ...)|(i, c_1, c_2, ...) \in T \wedge (j, c_1, c_2, ...) \notin T\}$$

which is a representation suitable to the task of recommending a ranked list of entities to each user. The set D retains information about differences in ratings in the set T, but in contrast to T it encodes the information as a ranking, stating which item i is ranked higher than item j. i is ranked higher than j in a certain combination of contexts c exactly if i was given positive feedback in this particular context and j did not receive positive feedback. This is illustrated in Fig. 2.

To overcome the problem of extreme sparsity we can now define the set

$$D_A = \{(i, j, c_1, c_2, ...)|\forall c_k : \#_{c_k}(i) > \#_{c_k}(j)\}, \qquad (1)$$

where $\#_{c_k}(i)$ is defined as the number of occurrences of item i in a given context c_k, not taking into account the other contexts. Formally,

$$\#_{c_k}(i) := \left|\left\{(i, c_k) \;\middle|\; \begin{array}{l}\exists c_1, \ldots, \exists c_{k-1}, \exists c_{k+1}, \ldots, \exists c_n :\\ (i, c1, \ldots, c_{k-1}, c_k, c_{k+1}, \ldots, c_n) \in T\end{array}\right\}\right|.$$

The set D_A then encodes a ranking where an item i is ranked higher than an item j in a certain combination of contexts if i appears more often in all these contexts. This is illustrated in Fig.3.

StarWars	Jan	Feb	March
Alice	1	0	0
Bob	1	1	0
Charlie	1	1	0

E.T.	Jan	Feb	March
Alice	0	0	0
Bob	1	0	0
Charlie	1	1	1

$$\begin{aligned}(\text{StarWars}, \text{E.T.}, \text{Alice}, \text{Jan}) &\in D_A\\ (\text{StarWars}, \text{E.T.}, \text{Alice}, \text{Feb}) &\in D_A\\ (\text{StarWars}, \text{E.T.}, \text{Bob}, \text{Jan}) &\in D_A\\ (\text{StarWars}, \text{E.T.}, \text{Bob}, \text{Feb}) &\in D_A\\ (\text{E.T.}, \text{StarWars}, \text{Charlie}, \text{March}) &\in D_A\end{aligned}$$

Fig. 3 Illustration of the augmented data set D_A. In order to overcome the problem of extreme sparsity in the original data the conditions on the set D are relaxed. It is sufficient if an entity appears more often than another entity in every context compared to appearing in that exact combination of contexts. Again, the grey cells correspond to the context setting in which the respective movie is ranked higher than the other movie in the set D_A.

3.3 Predicting Rankings

Next, we apply a method similar to Rendle et al (2010) which can predict unknown rankings over entities in the currently observed context combination from the given feedback data (and thus reconstructs the set D_A with the help of the chosen model parameters). The model of choice is pairwise interaction tensor decomposition, which is a special case of parallel factor analysis (PARAFAC). Here, the entire tensor is decomposed into additive factorized matrices, one pair of factorization matrices for each context:

$$\hat{x}_{i,\{c\}} := \sum_{k=1}^{n} \left\langle v_i^{I,C_k}, v_{c_k}^{C_k,I} \right\rangle,$$

where then $\hat{x}_{i,\{c\}} > \hat{x}_{j,\{c\}}$ signifies that entity i is considered to be ranked higher than entity j given the combination of context information ($\{c\}$ abbreviates $\{c1, c2, c3, \ldots\}$) in the trained model. v_i^{I,C_k} denotes the ith column in the factorization matrix V^{I,C_k}, which models the interaction of the kth context with the respective entity. Obviously, each pair of factorization matrices V^{I,C_k}, $V^{C_k,I}$ has to have corresponding inner dimensionality. This inner dimensionality guides the number of available parameters per context in the model.

Note that the respective context information may be of very diverse nature. For example, there may be time, sequence and location information available for the given feedback, like in the example in section 3.1.

Finally we optimize the available model parameters according to the ranking criterion BPR, introduced in

Rendle et al (2009), which is differentiable and similar to the area under the curve for ranking tasks (see Yan et al (2003), Ling et al (2003)). This optimization is performed via bootstrapped gradient descent as this has been shown to be much more efficient than cycling through the model parameters in a linear fashion [Rendle et al (2010)]. Since the optimization is done in a Bayesian setting, we can choose priors for all the model parameters. We chose a Gaussian prior $v \sim \mathcal{N}(0, \sigma)$ for every matrix entry in V^{I,C_k} and $V^{C_k,I}$ for all contexts C_k respectively and sampled the initial parameters from this prior at the beginning of the optimization. The optimization has proven to be rather insensitive to the choice of σ and a value of $\sigma = 0.2$ has been set for our experiments.

4 Modeling N-ary Relations using CRSVD

A set of instantiated relations $\{r(a_1, \ldots, a_n)\}$ corresponds to a set of statements. The task investigated in this paper is to predict new relations based on this data. We argue that a statistical framework for relation prediction, as pursued in this second approach, has advantages in that the assumptions behind the statistical model are made explicit and generalization to new instantiated relations is well understood. In essence, to derive a probabilistic model from a set of instantiated relations, we need to make assumptions about the way that the instances were generated. In object-oriented sampling one attribute identifies an object and the other attributes correspond to object properties. Here, one might assume that the objects were randomly selected out of a population. In this case, we can use statistical inference to generalize from the available data to the properties of objects in the population. Graphical probabilistic models using this object-oriented sampling assumption are discussed in the following subsection. Object-oriented sampling is not the only one possible and, indeed, might not be well suited for relational domains with multiple objects. Another reasonable sampling assumption, and the one used in our approach, is to assume that instantiated relations are a random sample out of a population of instantiated relations. We can now use statistical inference to generalize from the available data to likely new instantiated relations. We will discuss this model in Section 4.2.

4.1 Standard Object-Oriented Sampling Assumption

Traditionally, statistical units, i.e. data points, are associated with objects and statistical models are concerned with the statistical dependencies between attributes of those objects. A typical example is a medical application where one analyzes the dependencies between the attributes of a population of patients, for example in form of a Bayesian network. In a data matrix the patients would define the rows and would act as unique identifiers and the attributes would define the columns. A fundamental task is then to predict if a novel object belongs to the same population (density estimation), or what values a variable has to assume such that the likelihood that the object belongs to the same population is maximized (predictive modeling).

	I1	I2	I3	I4
U1	1	1	0	1
U2	1	0	0	1
...	...	...	...	...

	U	I
ID1	U1	I1
ID2	U1	I2
ID3	U2	I4
...	...	...

Fig. 4 Left: In a more traditional view, each row is defined by a user and the columns represent the different items. A one indicates that a user has purchased an item. Right: Each row is defined by an event user-buys-item, which is the sampling assumption used in this paper.

This approach is also quite common in modeling relational domains. For example, one might analyze the preferences of a population of U users based on user attributes and based on known preferences for I items, e.g., *buy(User, Item)*, where the preferences are essentially also treated as attributes of the users (Figure 4, Left). In Breese et al (1998) a Bayesian network is described where a binary node x_j represents an item and the state of the node indicates if a user has bought an item ($x_j = 1$) or not ($x_j = 0$). The Bayesian network then models

$$\hat{P}(x_1, \ldots, x_I). \tag{2}$$

A problem one encounters in these models is that one needs to distinguish between relationships known not to exist and relations that are unknown. For example, in the Bayesian networks in Breese et al (1998) and in the Dependency Networks [Heckerman et al (2000)], missing relations are treated as not-to-exist whereas in Koller and Pfeffer (1998); Xu et al (2006); Domingos and Richardson (2007); Getoor et al (2007) Gibbs sampling and loopy belief propagation are used for dealing with unknown relationships.

4.2 Relation-Oriented Sampling Assumption

In our relation-oriented view, an instance is defined by an observed relation, i.e., a tuple, typically describing the relationship between two or more objects (Figure 4, Right). The population then consists of all true tuples and a sample is a random subset of those true tuples. Thus, whereas in the previous subsection we assumed that either users or items define the rows in the data matrix, here we assume that each observed instantiated relation (tuple) defines a row.

Considering again the relation *buy(User, Item)*, the data matrix would contain two columns encoding the user and the item, respectively, and a model would estimate

$$\hat{P}(User = u, Item = i). \tag{3}$$

Note that whereas Equation 2 describes a probability distribution over I binary variables, this equation describes a multinomial model with two variables where the two variables have U and I states, respectively.

Considering now that we generalize from two to A attributes that describe a relation, i.e., are informative for determining the existence of a relation, the basic problem is to evaluate $P(x_1, \ldots, x_A)$, i.e., the probability that a novel relationship with attributes $x_1, \ldots, x_A$ exists. Alternatively, it might be interesting to predict the most likely value of one of the attributes given other attributes, such as $P(x_1|x_2, \ldots, x_A)$, e.g., the probability of an item x_1 given a user x_2 and given contextual information $x_3, \ldots, x_A$.

In object-to-object relationships, variables typically contain many states and a contingency table involving all variables can be very sparse. In high-dimensional domains graphical models have been quite effective in the past [Lauritzen (1996)] and so in this paper we will apply them as well. As discussed earlier, the novelty of the proposed approach is that we apply graphical models in domains where the relations form the instances and where we model just a single relation instead of a whole network of entities and their relationships.

For our purpose, Bayesian networks and decomposable models are most suitable. For a Bayesian network model, the probability distribution factors as

$$\begin{aligned} P(x_1, \ldots, x_A) &= \prod_{i=1}^{A} P(x_i|\mathbf{par}(x_i)) \\ &= \prod_{i=1}^{A} \frac{P(x_i, \mathbf{par}(x_i))}{P(\mathbf{par}(x_i))} \end{aligned}$$

Typically a Bayesian network is depicted as a directed graphical model without directed loops. In this model, $\mathbf{par}(x_i)$ denotes the direct parents of x_i.

Given a Bayesian network structure, the task is then to model $P(x_i|\mathbf{par}(x_i))$, or equivalently, $P(x_i, \mathbf{par}(x_i))$. If the involved variables have many states, matrix and tensor completion methods have been successful in the past and we also apply those in our approach, as described in Section 5.2.

4.3 Context-Aware Regularized Singular Value Decomposition (CRSVD)

As a next step, we consider the modeling of the probabilistic tables used in the Bayesian network. Let's consider that in the Bayesian network a node only has one parent. Since the attributes in the relation are typically high dimensional but sparse, the empirical contingency table of counts is a many state sparse matrix. Consequently, a maximum likelihood estimate would hopelessly over-fit the data and assign zero probability to most states. This situation is common to text models where the standard approach is to apply some form of smoothing. For example in Laplace smoothing one adds a couple of ones to each table entry. Certainly this would not be helpful in our application, where we are interested in the precise order of probabilities. Thus, we use matrix factorization as a means of smoothing the contingency tables, which, after proper normalization, can be interpreted as probabilities. This idea is discussed in more detail in context of the application in Section 5. Since we apply matrix factorization in form of a regularized singular value decomposition, we named our approach Context-Aware Regularized Singular Value Decomposition (CRSVD). In case that a node has more than one parent, we would get a contingency tensor to which smoothing via factorization might be applied as well. Naturally, we run into the problem of sparsity and computations complexity, as discussed before.

5 Application of Context-aware Relation Prediction to Social Networks

The two context-aware relation prediction methods are evaluated using a real-world data-set from the social network GetGlue[2]. GetGlue allows users to share the experience of navigating the Web and has a number of features enabling social information exchange. GetGlue uses semantic recognition technologies to automatically identify books, music, movies, wines, stocks, movie stars and many other similar entities, and thus a user generates a continuous data stream of object annotations. Users can observe the data stream and can receive recommendations from GetGlue about interesting discoveries by their friends. Both, the social network data and

[2] http://getglue.com

the real-time streams are accessible via a REST API[3]. For the purpose of our experiments we used a set of wrappers [Barbieri et al (2009)] that export GetGlue data in the form of Linked Data (LD).

We continuously sampled data sets through the GetGlue's network API. To ensure a focused analysis we gathered movie ratings only. We sampled from 50 to 400 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 (for example the "like button" available on the *facebook* social network).

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 Sec. 5.1.2) is inferred from the timestamps that are available as is the month in which the movie was rated (see Sec. 5.1.3).

5.1 Applying CARTD

In the following we recommend movies to users based on the GetGlue data described in Section 5 using the CARTD method. We will describe how to achieve a collaborative filtering setting first and describe how to add additional context information next.

5.1.1 Modeling User-Movie Events

In the formalism of abstract sets described in Section 3.1, we define:

Entity $:= I$	all the available items
Context$_1$ $:= U$	all the users

This corresponds to a usual collaborative filtering setting. The user preference similarities are shared between users by means of matrix factorization. In the generic CARTD model, the user information is also considered as context information just like all other not deterministically dependent context (see Sec. 3.1).

[3] http://getglue.com/api

5.1.2 Adding Information on the Last Movie Watched

Next, we add to the user-movie recommendation setting the available sequential context information, i.e., on which movie the user has last given positive feedback. This is not deterministically dependent context in the sense of Fig. 1. It is not determined by either the user or the movie and it is available when the recommendation is performed. The sequential information is calculated from the GetGlue data by means of the available timestamps of every rating. Our experiments (see Sec. 6) show that the added sequential information is able to significantly improve predictive performance of the model.

Context$_2$ $:= I$ all the available items (as last items)

5.1.3 Adding Time of the Event

We also add time information directly, as the month in which the movie was rated. The total rating period of our dataset is 43 months. Again, in the experiments we will see that the added time information also results in a gain in predictive performance of the model.

Context$_3$ $:= M$ all the months of the rating period

5.1.4 Implementation

The CARTD model is implemented in Java including all the extraction tools needed in order to gain the relevant information and the tensor from data coming from the RDF wrapper (see 3.1). This makes extensive use of Apache's implementation of a http client in Java[4] and for temporary storage the eXist XML database [5] is used. The matrices and tensors are represented in the Colt matrix library from CERN [6] and computations have been done on an off-the-shelf 2.53 GHz MacBook Pro (4GB RAM). For computation times needed for the predictions see Section 6.

5.2 Applying CRSVD

In the following we model the user-movie recommendation by using CRSVD. We incrementally increase the model complexity by first describing a user-movie model, and by then adding information about the last movie watched by a user and the time of the event.

[4] http://hc.apache.org/httpcomponents-client-ga/
[5] http://exist-db.org
[6] http://acs.lbl.gov/software/colt/

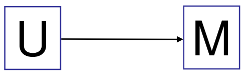


Fig. 5 A graphical model for the dependencies between users U and movies M.

5.2.1 Modeling User-Movie Events

We model the event that a user watches a movie. The graphical model consists of two attributes, i.e., the user and the movie (Figure 5). The rows in the data matrix are then defined by known user-movie events and the columns consist of two variables with as many states as there are users and movies, respectively. A contingency table C is formed. Entry $c_{u,m}$ counts how often user u has watched movie m. By dividing the entries by the overall counts, we can interpret the entries as estimates for the probabilities of observing a user-movie pair under this sampling assumption, i.e. as a maximum likelihood estimate of $P(u,m)$. This matrix will contain many zero entries and the maximum likelihood estimates are notoriously unreliable. We follow common practice and smooth the matrix using a matrix factorization approach. We perform a singular value decomposition $CC^T = \mathcal{U}D\mathcal{U}^T$ and obtain the low-rank approximation [Huang et al (2010)]:

$$\hat{C} = \mathcal{U}_s \operatorname{diag}_s \left(\frac{d_l}{d_l + \lambda} \right) \mathcal{U}_s^\top C,$$

where $\operatorname{diag}_s \left(\frac{d_l}{d_l+\lambda} \right)$ is a diagonal matrix containing the s leading eigenvalues in D and where $\mathcal{U}_s$ contains the corresponding s columns of U. λ is a regularization parameter. After proper normalization $\hat{C}$, the entries can be interpreted as $\hat{P}(u,m)$, i.e., an estimate of the probability of observing the relation that user u watches movie m.[7]

It should be noted that matrix completion is an active area of research and many other matrix completion methods are applicable as well.

Recommendations for users can now be based on $\hat{P}(u,m)$.

5.2.2 Adding Information on the Last Movie Watched

Certainly, there is a sequential nature to the user-watches-movie process that the model so far cannot capture. In particular we might consider the last movie that a user has watched as additional information [Rendle et al (2010)]. Note that we now obtain a truly ternary relation *watches(u,m,l)* consisting of user, movie and last movie l watched by the user. The approach followed in Rendle et al (2010) is to consider a three-way contingency table and apply tensor factorization as a tensor smoothing approach. There it was argued that general tensor factorization, such as PARAFAC or Tucker [Kolda and Bader (2009)], are too difficult to apply in this situation since the contingency table is very sparse and a simplified additive model is applied instead. In our approach we suggest an appropriate graphical model as shown in Figure 6 (left).[8] The model indicates that the last movie watched by a user directly influences the next movie that a user watches but that given that information, last movie and user are independent. The great advantage now is that we do not need to readapt the user-movie model but can model independently the movie-last-movie dependency. Again we calculate empirical probabilities based on the contingency table, smooth the table using matrix factorization and obtain $\hat{P}(m,l)$. We combine both models and form

$$\hat{P}(u,m,l) = \frac{\hat{P}(u,m)\hat{P}(m,l)}{\hat{P}(m)}.$$

Note that in contrast to Rendle et al (2010), we do not obtain a *sum* of local models but a *product* of local models.

5.2.3 Adding Time of the Event

Next we consider the instance of time t when a movie is watched. Certainly, the preference for movies changes through time and at certain points in time a movie might be very popular and then decrease in popularity. Also, a movie can only be watched after it is released. Time of watching in units of month is added to the model. Again we formed an empirical estimate based on the movie-time of watching contingency table. The graphical model is shown in Figure 6 (right).

We now obtain

$$\hat{P}(u,m,l,t) = \frac{\hat{P}(u,m)\hat{P}(m,l)\hat{P}(m,t)}{(\hat{P}(m))^2}.$$

[7] Normalization takes care that all entries are non-zero and are smaller than one. Incidentally, this step turns out to be unnecessary in the regularized reconstruction, since after matrix completion all entries already obeyed these constraints. A second step ensures that the sum over matrix entries is equal to one.

[8] A link from the last movie to movie might appear more plausible. If one does that change, the link between user and movie would need to point from movie to user, such that no collider (more than one link pointing to the same node) appears. With a collider one would need to use a tensor model as a local model.

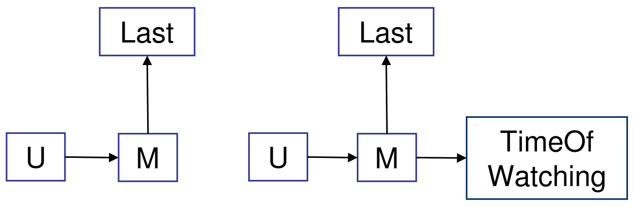


Fig. 6 Left: As additional information, the last movie, which the user has watched, is added. Right: The month when the user watches the movie is added.

5.2.4 Implementation

We implemented the CRSVD model in Matlab where we used the svds function[9] to compute the largest singular values and associated singular vectors of data matrices. We ran the evaluation on a laptop with Windows 7, Intel i5 CPU 2.5 GHz and 8 GB RAM. The computation times needed for training models are reported in Section 6.

6 Experiments

In this section we report experimental results of applying the two proposed approaches CRSVD and CARTD on the data set described in Section 5. First, we describe two settings of our experiments. Then, we introduce a baseline which we used to compare both approaches and different evaluation measures used, i.e., HitRatio, area under the curve (AUC) and normalized discounted cumulative gain (nDCG). We evaluate the quality of recommendation with only user-movie ratings and by integrating contextual information.

6.1 Settings

We tested the approaches on two different scenarios. In the first scenario, the *newest movie setting*, we used for testing the lastest movie for which the respective user provided positive feedback. This setting corresponds to a very realistic application: usually we would want to recommend to a user movies to watch next given a sequence of movies that were rated positively before. In the second scenario, the *random movie setting*, we randomly used one movie as test data for each user. This setting corresponds more to traditional tasks for collaborative filtering, where context information and especially sequential information is not considered. In both settings movies that a user already rated are not recommended again to the same user (i.e., we do not recommend watched movies). This might be handled differently in other applications, for instance in an e-commerce application where each user would reasonably want to buy the same items twice or more. In the latter setting we repeated the experiment 5 times such that error bars can be produced (see figures below).

[9] http://www.mathworks.de/help/techdoc/ref/svds.html

6.2 Methodology

The particular approaches and context information for which we report evaluation results are:

most popular: The *most popular* baseline recommends for every user the same list of movies which are sorted by the number of positive ratings in descending order. The most positively rated movies are on top of the list.

CARTD(UM): The Context-Aware Recommendation Tensor Decomposition approach, described in Section 3.1 using only the user-movie ratings (see Sec. 5.1.1). This corresponds to traditional collaborative filtering with a ranking-based optimization criterion.

CARTD(UM+Sequence+Month): The CARTD approach, using additionally sequence and month context information. This information is taken into account as described in Section 5.1.2 and 5.1.3.

CRSVD(UM): The Context-aware Regularized Singular Value Decomposition approach, described in Section 4 using only the user-movie ratings.

CRSVD(UM+Sequence+Month): The CRSVD approach, using additionally sequence and month context information as described in Sections 5.2.2 and 5.2.3.

In our experiments we utilized three evaluation methods. First, we generated a recommended list of k items for each user. If the single test item appears in the recommended list, then for that user we consider the recommendation successful. The ratio of successful recommendations vs. all recommendations is called *HitRatio*. Mathematically, the HitRatio is defined as

$$\text{HitRatio}(k) = \frac{1}{|U|} \sum_{u \in U} \delta(T_u \in R_k),$$

where δ is the indicator function, T_u is the user's test item, R_k are the top k recommended items given the contexts which in our case are the last movie watched and the month of watching.

Secondly, we evaluated the tested approaches according to the area under the curve (AUC) measure which for ranking tasks can be simplified to the form

$$\text{AUC}(u) = \frac{|I| - pos(T_u)}{|I| - 1},$$

where T_u is the test item for the user u and $pos(\cdot)$ gives the position of an item in the ranked list of recommended items, 1 being the best position and $|I|$ the worst position. Ataman et al (2006) describes the equivalence relation between the AUC and the Wilcoxon-Mann-Whitney statistic from which the simplified formulation can easily be derived, since in both settings there is only one single test movie for each user.

Finally, we applied normalized discounted cumulative gain (nDCG) to evaluate the quality of the recommendation. NDCG is calculated by summing over all the gains in the ranked list R with a log discount factor as

$$\mathrm{nDCG}(R) = \frac{1}{Z}\sum_{k}\frac{2^{r(k)}-1}{\log(1+k)},$$

where $r(k)$ denotes the target label for the k-th ranked item in R, and r is chosen such that a perfect ranking obtains value 1. To focus more on the top ranked items, one can also consider the nDCG@n which only counts the top n items in the ranked list for a user.

Each of these evaluation measures focuses on different aspects of the ranking. NDCG especially emphasizes the very top positions in R due to the log function on the position k, which means that the nDCG score increases when the position of a test item is lifted from the second position to the first one, much more than when its position is improved from 100 to 99. In contrast, AUC evaluates the quality of the entire ranking list. The HitRatio measures all positions in the ranking list equally, similar to AUC, but it is specified to focus on the top k items in the ranking list only.

6.3 Results

Figure 7 shows the AUC performance of the evaluated approaches in the *newest movie setting* and the *random movie setting*, respectively. The AUC values are plotted against the factorization dimension which is $10, 50, 100, 200, 400$ and 1000 respectively. In the *random movie setting* we also report error bars which show the standard error.

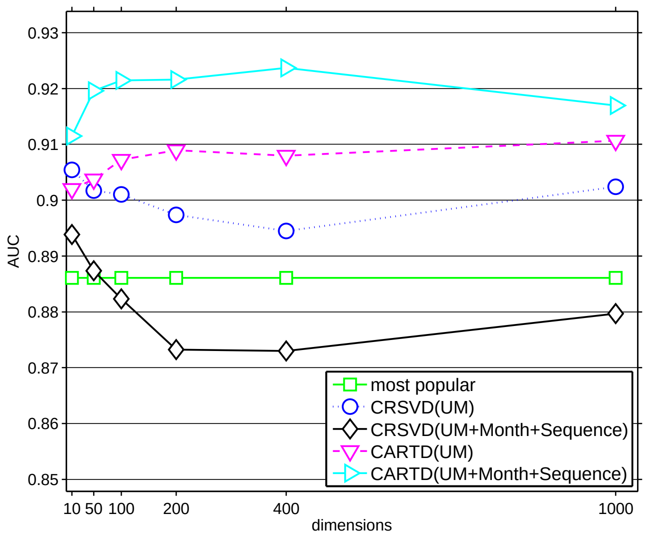
The CARTD approach clearly outperformed the most popular baseline in both settings. The performance of CARTD increases with increasing dimensions and reaches its best AUC value at 400 dimensions. Integrating the contextual information significantly improved its performance when compared to only using rating information. CRSVD(UM) outperformed the most popular baseline independent of the factorization dimension. In particular, in the *random movie setting* its performance was even close to CARTD(UM+sequence+month). Adding contextual information could not improve AUC values for the CRSVD approach. CRSVD(UM+sequence+month) was better than the baseline only at low dimensions, i.e., 10 and 50, and always worse than CRSVD(UM). This scenario occurred again in comparisons using the HitRatio measure when the number of the top items k was reasonably large (see Figures 8 (c) and (d)). As explained before, AUC evaluates the quality of the entire recommended list. For the CRSVD approach we did not observe any improvement with increasing factorization dimension.

Figures 8 (a) and (b) show the HitRatio for the top 10 movies achieved by the two approaches in the *newest movie setting* and the *random movie setting* respectively. We ran each of the approaches including the different context information with factorization dimensions $10, 50, 100, 200, 400$ and 1000. Again, for the *random movie setting* we also report the standard error.

Both figures show that CARTD with only user-movie rating information was not able to perform better than the *most popular* baseline for a top 10 movies recommendation on this dataset. The additional time and sequence context information clearly improved the recommendations given by the CARTD model, although it was outperformed in both settings by the CRSVD approach even without additional context information. Clearly, also the performance of the CRSVD approach improved significantly if the time and sequence context information was taken into account. It performed especially well in the *random movie setting*, where it was able to correctly recommend the test movie with close to 28% of the users with the top 10 recommendations, when the factorization dimension was equal to 400.

Figures 8 (c) and (d) show the HitRatio of the tested approaches with different values of k for a fixed factorization dimension of 400. The figures show that for low values of k, the CRSVD approach outperformed CARTD, as has been seen in Figures 8 (a) and (b). For k greater than 100, CARTD with all context information outperformed all other approaches. Especially in the random movie recommendation setting we see that the CRSVD approach was only able to gain 20% from $k = 100$ up to $k = 1000$, while CARTD with all context information remained able to outperform the most popular baseline throughout.

Figure 9 shows nDCG scores achieved by the CRSVD approach in the *newest movie setting* and the *random movie setting* respectively. The results are plotted against dimensions, again, from 10 to 1000. The most popular baseline generated the worst scores. CRSVD with only rating information performed much better and CRSVD with all contextual information produced the overall



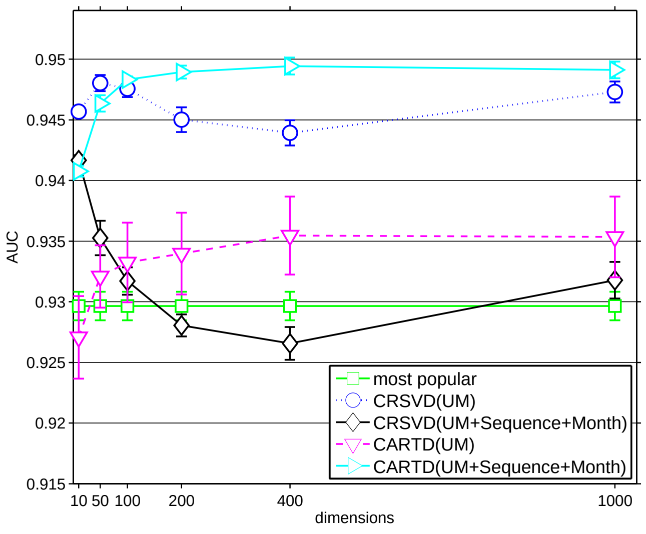


Fig. 7 The AUC values for different factorization dimensions. Left: The *newest movie setting*. Right: The *random movie setting*. Details in Sec. 6.1.

`Time [min]`	10	50	100	200	400	1000
CARTD(UM)	2.17	2.68	3.57	5.58	13.47	33.93
CARTD(full)	2.85	4.06	5.58	10.69	23.90	61.76
CRSVD(UM)	0.01	0.04	0.11	0.35	0.97	4.74
CRSVD(full)	0.04	0.11	0.25	0.74	2.46	14.32

Fig. 10 The computation times for different settings on common laptops, where “full here means that all available context information was used, i.e. the user, sequence information and month information.

best results. Here, we do not plot the nDCG scores of the CARTD approach, as the performance of CARTD(UM) was more or less the same as that of the baseline and CARTD(UM+sequence+month) performed only slightly better than the baseline.

The time needed for computations of the two methods with different factorization dimensions (columns) are reported in Table 10. CRSVD is computationally more efficient but CARTD also proofs to be scalable to higher-order relations in most real-world scenarios.

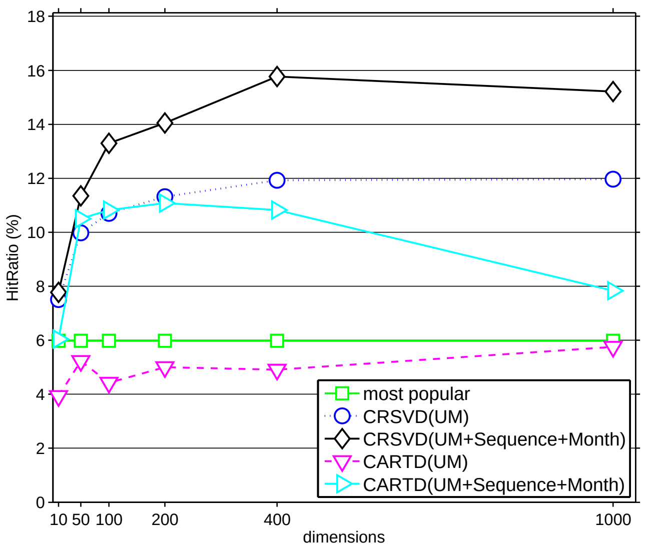
6.4 Discussion

Based on the experimental results described in the previous section we observed the following points.

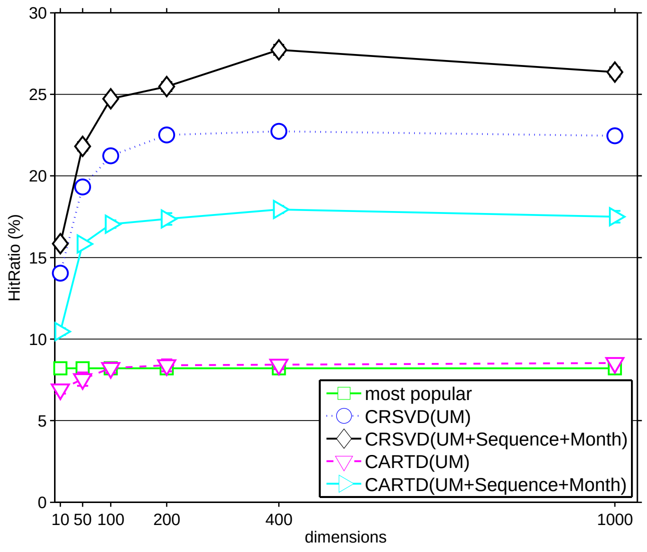
First, both CRSVD and CARTD outperformed significantly the most popular baseline in most cases. Second, modeling contextual information improved the quality of the recommendation to a great extent, when compared to considering only user-item ratings. That confirms the main contribution of the methods presented in this paper. Third, for the recommendation task emphasizing the quality of the very top ranked items, the CRSVD approach performs usually better, while the CARTD approach is rather suitable for general relation prediction tasks, since it provides a higher quality of recommendation regarding the whole ranking list. This can be explained by the nature of CRSVD that tends to better model both, more popular items and such users who own a great amount of ratings and can be viewed as opinion leaders. Both popular items and opinion leaders strongly affect the behavior of the whole community, in particular in social networks and recommendation systems. Forth, CRSVD and CARTD are both, robust and insensitive to the number of latent variables. This can be explained by the fact that CRSVD, in contrast to other matrix factorization approaches such as SVD, is regularized and CARTD performs implicit model averaging. The insensitivity to parameter tuning improves the usability of both approaches. Fifth, these two approaches are efficient and are capable to scale up to large data sets. Again, we carried out our experiments on a regular laptop.

7 Conclusion and Future Work

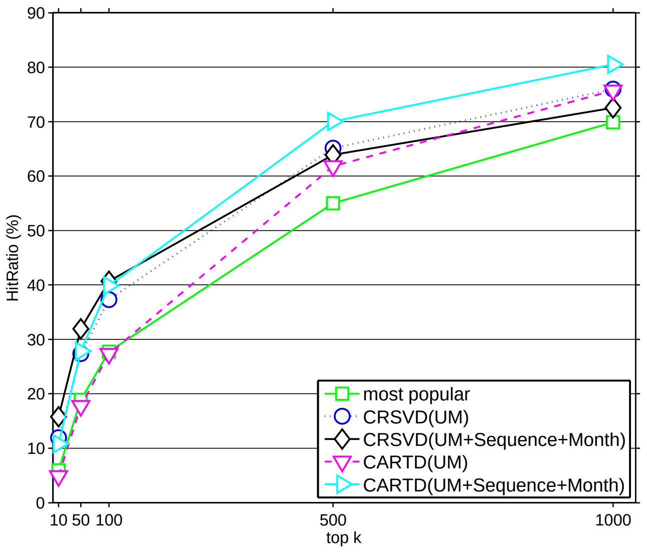
In this paper we covered the modeling of context-aware relations for the task of relation prediction in social networks, e.g., recommending movies to users. We consider relational information to be context-aware if it provides supplementary information for predicting the relation of interest, as in *watches(User, Movie, LastMovieWatchedByUser, Month)*. We have argued that an n-ary relation can be elegantly modeled by a tensor with n modes but that sparsity and computational com-



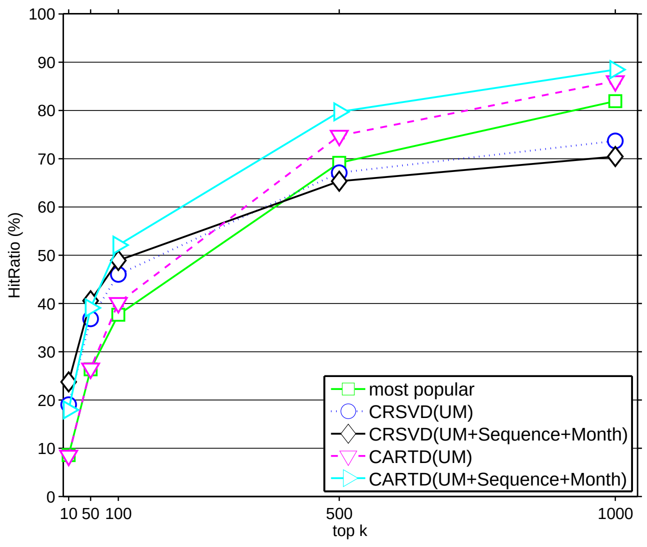
(a)



(b)



(c)



(d)

Fig. 8 Top: (a) and (b) report the HitRatio for a recommended list of top $k = 10$ movies vs. the factorization dimension. Bottom: (c) and (d) report the HitRatio for different values of k while the factorization dimension is fixed at 400. Left: (a) and (c) plot HitRatio values in the *newest movie setting*. Right: (b) and (d) plot HitRatio values in the *random movie setting* with error bars showing the standard error.

plexity make tensor factorization impractical for $n > 3$. We introduced two different approaches to simplifying the tensor model, both relying on factorization methods to estimate unknown tensor elements. The main difference lies in the way each method handles the extreme sparsity of a high dimensional tensor. While the first approach, the Context-Aware Recommender Tensor Decomposition (CARTD), proposes an efficient ranking and optimization criterion, the second approach, the Context-aware Regularized Singular Value Decomposition (CRSVD), introduces a generative probabilistic model and aims at reducing the dimensionality using independence assumptions in graphical models.

Our experimental results show that both models can successfully handle the high dimensionality, making them computationally efficient and providing accurate predictions. Also, both approaches provide a robust performance making them usable for non machine learning experts. However, our evaluation also suggests that CARTD has an advantage if a complete ranking is to be predicted, whereas CRSVD is especially suited for tasks where only a small number of top recommendations is needed.

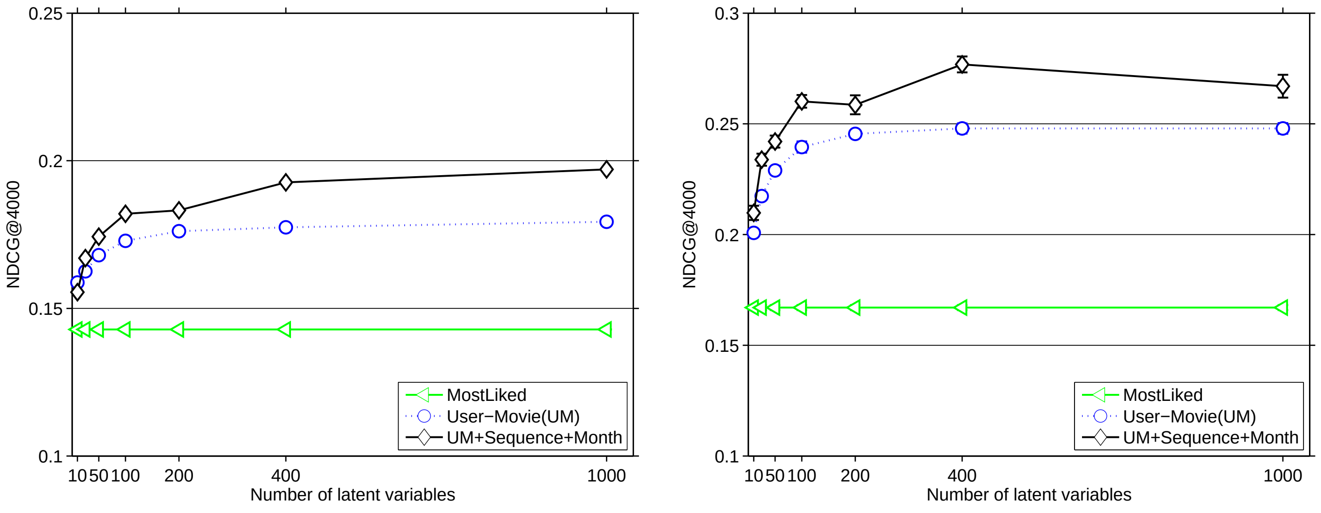


Fig. 9 The nDCG scores at top 4000 for different factorization dimensions. Left: The *random movie setting*. Right: The *newest movie setting*. Details in Sec. 6.1.

We suggest that tensor decomposition for relation prediction has great potential for future work. Examples are the extension of models to scenarios with multiple relations, social network analysis on latent factor matrices and the analysis of the dynamical developments in social networks.

References

Ataman K, Nick W, Member S, Zhang Y (2006) Learning to rank by maximizing auc with linear programming. In: IEEE International Joint Conference on Neural Networks (IJCNN) 2006

Barbieri DF, Braga D, Ceri S, Valle ED, Grossniklaus M (2009) Continuous queries and real-time analysis of social semantic data with c-sparql. In: Second Workshop on Social Data on the Web (SDoW2009)

Bell RM, Koren Y, Volinsky C (2010) All together now: A perspective on the netflix prize. Chance

Breese JS, Heckerman D, Kadie C (1998) Empirical analysis of predictive algorithms for collaborative filtering. In: Uncertainty in Artificial Intelligence

Cands EJ, Recht B (2008) Exact matrix completion via convex optimization. Computing Research Repository - CORR

Chu W, Ghahramani Z (2009) Probabilistic models for incomplete multi-dimensional arrays. In: AISTATS

Chu W, Sindhwani V, Ghahramani Z, Keerthi SS (2006) Relational learning with gaussian processes. In: NIPS

Domingos P, Richardson M (2007) Markov logic: A unifying framework for statistical relational learning. In: Getoor L, Taskar B (eds) Introduction to Statistical Relational Learning, MIT Press

Getoor L, Friedman N, Koller D, Pferrer A, Taskar B (2007) Probabilistic relational models. In: Getoor L, Taskar B (eds) Introduction to Statistical Relational Learning, MIT Press

Heckerman D, Chickering DM, Meek C, Rounthwaite R, Kadie CM (2000) Dependency networks for inference, collaborative filtering, and data visualization. Journal of Machine Learning Research

Huang Y, Tresp V, Bundschus M, Rettinger A, Kriegel HP (2010) Multivariate structured prediction for learning on the semantic web. In: Proceedings of the 20th International Conference on Inductive Logic Programming (ILP)

Kemp C, Tenenbaum JB, Griffiths TL, Yamada T, Ueda N (2006) Learning systems of concepts with an infinite relational model. In: Poceedings of the National Conference on Artificial Intelligence (AAAI)

Kolda TG, Bader BW (2009) Tensor decompositions and applications. SIAM Review

Koller D, Pfeffer A (1998) Probabilistic frame-based systems. In: Proceedings of the National Conference on Artificial Intelligence (AAAI)

Koren Y (2009) Collaborative filtering with temporal dynamics. In: KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, New York, NY, USA

Lauritzen SL (1996) Graphical Models. Oxford Statistical Science Series

Linden G, Smith B, York J (2003) Amazon.com recommendations: Item-to-Item collaborative filtering. IEEE Internet Computing 7(1):76–80

Ling CX, Huang J, Zhang H (2003) Auc: a statistically consistent and more discriminating measure than accuracy. In: Proceedings of the 18th international joint conference on Artificial intelligence, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp 519–524

Rendle S, Freudenthaler C, Gantner Z, Schmidt-Thieme L (2009) BPR: bayesian personalized ranking from implicit feedback. In: Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI)

Rendle S, Freudenthaler C, Schmidt-Thieme L (2010) Factorizing personalized markov chains for next-basket recommendation. In: World Wide Web Conference

Salakhutdinov R, Mnih A (2007) Probabilistic matrix factorization. In: NIPS

Takacs G, Pilaszy I, Nemeth B, Tikk D (2007) On the gravity recommendation system. In: Proceedings of KDD Cup and Workshop 2007

Wermser H, Rettinger A, Tresp V (2011) Modeling and learning context-aware recommendation scenarios using tensor decomposition. In: International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2011), IEEE CPS

Xu Z, Tresp V, Yu K, Kriegel HP (2006) Infinite hidden relational models. In: Uncertainty in Artificial Intelligence (UAI)

Yan L, Dodier R, Mozer M, Wolniewicz R (2003) Optimizing classifier performance via the wilcoxon-mann-whitney statistics. In: Proceedings of the 20th International Conference on Machine Learning (2003)

Yu K, Chu W, Yu S, Tresp V, Xu Z (2006) Stochastic relational models for discriminative link prediction. In: Advances in Neural Information Processing Systems 19