Link Prediction in Multi-relational Graphs using Additive Models

Xueyan Jiang[2], Volker Tresp[1,2], Yi Huang[1,2], and Maximilian Nickel[2]

[1] Siemens AG, Corporate Technology, Munich, Germany
[2] Ludwig Maximilian University of Munich, Munich, Germany

Abstract. We present a general and novel framework for predicting links in multirelational graphs using a set of matrices describing the various instantiated relations in the knowledge base. We construct matrices that add information further remote in the knowledge graph by join operations and we describe how unstructured information can be integrated in the model. We show that efficient learning can be achieved using an alternating least squares approach exploiting sparse matrix algebra and low-rank approximations. We discuss the relevance of modeling nonlinear interactions and add corresponding model components. We also discuss a kernel solution which is of interest when it is easy to define sensible kernels. We discuss the relevance of feature selection for the interaction terms and apply a random search strategy to tune the hyperparameters in the model. We validate our approach using data sets from the Linked Open Data (LOD) cloud and from other sources.

1 Introduction

There is a growing amount of data published in multirelational graphs where information elements are represented as subject-predicate-object (s, p, o) triples. Entities (i.e., subjects and objects) are represented as nodes and statements are represented as directed labeled links from subject node to object node. A machine learning task of some generality is the prediction of links between entities using patterns in known labeled links in the knowledge base.

We present a general framework for predicting links in multirelational graphs using a set of matrices describing the various instantiated relations in the knowledge base. We first consider triples in the immediate neighborhood of the triple of interest and then construct matrices that add information further remote in the knowledge graph by performing join operations. We also consider the case that unstructured information is available that can support the link prediction task and we describe how unstructured information can be integrated in the model. Examples of unstructured information are textual documents describing the involved entities (e.g., from the entities' Wikipedia pages). We show that efficient learning can be achieved using an alternating least squares approach exploiting sparse matrix algebra and low-rank approximations. We discuss the relevance of modeling nonlinear interactions and add corresponding model components. We also discuss a kernel solution which is of interest when it is easy to define sensible kernels. We discuss the relevance of feature selection for the interaction terms and apply a random search strategy to tune the hyperparameters in the model. We validate our approach using data sets from the Linked Open Data (LOD) cloud and from other sources.

The paper is organized as follows. The next section discusses related work. Section 3 describes our basic approach. Section 4 describes the cost function and the alternating least squares solution for parameter learning. In Section 5 we discuss aggregation via joint operations, the inclusion of unstructured information and interaction terms. Section 6 contains our experimental results. Section 7 presents our conclusions.

2 Related Work

One of the first line of research where matrix representations were used for link prediction in multirelational graphs is the SUNS framework [10] [5]. A major extension was the probabilistic extension of the SUNS approach reported in [7]. The same paper also describes how information extraction (IE) can be combined with deductive reasoning and machine learning for link prediction, where the combination is implemented as a postprocessing step. The additive approach presented in this paper is novel and has several advantages. It considers a model for a complete knowledge-base of triples and considers dependencies on all triples in the immediate neighborhood of the triple. Whereas in [7], the combination was a postprocessing step, here we optimize the additive model globally. Also the discussion on aggregation by joint operations is novel, as well as the application of alternating least squares for optimizing the penalized cost function.

The winning entries in the Netflix competitions are based on matrix factorization [1]. The main difference is that in those applications unknown ratings can be treated as missing entries whereas in relational prediction, the topic here, they are treated as negative evidence.

Multi-relational graphs also map elegantly to a tensor representation. Tensor models for relational learning have been explored in [9], showing both scalability and state-of-the-art results on benchmark datasets.

3 Link Prediction in Multi-relational Graphs

3.1 Relational Adjacency Matrices

In this paper we assume that labeled links are represented as triples of the form (s, p, o) where subject s and object o stand for entities in a domain and where p is the predicate, i.e. the link label. We define a variable $x_{i,j,k}$ that is associated with the triple (s $= i$, p $= j$, o $= k$). We set $x_{i,j,k} = 1$ when the triple is known to exist, otherwise $x_{i,j,k} = 0$. In the multirelational graph, the entities form the nodes and the existing triples form labeled links.

We now consider a domain with N entities and P predicates. For the predicate p$=j$ in the domain we define a relational adjacency matrix $X_j \in \mathbb{R}^{N \times N}$

where $(X_j)_{i,k} = 1$ if $x_{i,j,k} = 1$ and $(X_j)_{i,k} = 0$ otherwise. The matrix of concatenated relational adjacency matrices $X = (X_1, \ldots X_P)$ describes all existing and all potential triples involving all known entities in the knowledge base.

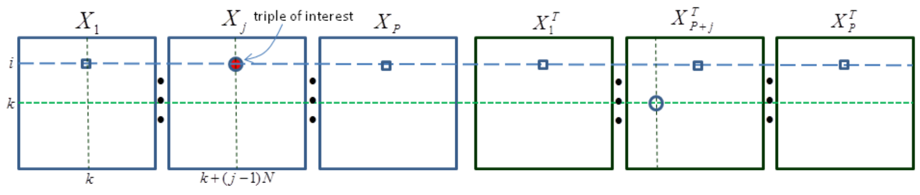

Fig. 1. The figure shows the matrix M and illustrates the terms in Equation 1. We assume that the goal is to predict the matrix entry $\hat{x}_{i,j,k}$ in matrix X_j indicated by the small red circle. The terms in the first sum (subject term) are represented by the upper dashed blue line, the terms in the second sum (object term) are represented by the lower green dotted line, and the terms in the third sum (subject-object term) correspond to the small rectangles.

3.2 The Basic Model

In the basic model we assume that the truth value of a triple (s= i, p= j, o= k) can be estimated as a linear combination of directly related triples, defined as all triples (s= i, p= j', o= k') where i is the subject, all triples (s= i', p= j', o= i) where i is the object, all triples (s= k, p= j', o= k') where k is the subject and all triples (s= i', p= j', o= k) where k is the object. Finally we consider triples with arbitrary predicates but where two entities from the target triple are involved, i.e., (s= i, p= j', o= k), and (s= k, p= j', o= i). If $x_{i,j,k} = 0$, the predicted $\hat{x}_{i,j,k} \geq 0$ can then be interpreted as a likelihood that the triple is true based on the immediate context of the triple.

We now form the matrix $M = (X, X^\dagger)$ where $X^\dagger = (X_1^T, \ldots X_P^T)$ denotes the in-place matrix transposed of X (Figure 1). Let $(M)_{i,l} = m_{i,l}$.

Following the discussion we form the model

$$\hat{x}_{i,j,k} = \sum_{l=1}^{2PN} w_{l,k+(j-1)N}\, m_{i,l} + \sum_{l=1}^{2PN} r_{l,i+(j-1)N}\, m_{k,l} + \sum_{l=1}^{2P} h_{l,j}\, m_{i,k+N(l-1)} \quad (1)$$

For further reference, we call the first term in the sum in Equation 1 the subject term, the second one the object term, and the last one the subject-object term.[3] The $w_{.,.}$, $r_{.,.}$, and $h_{.,.}$ are model parameters to be estimated.

[3] Note that we get nontrivial solutions by using regularized parameter fits with low-rank constraints, as described in Section 4.

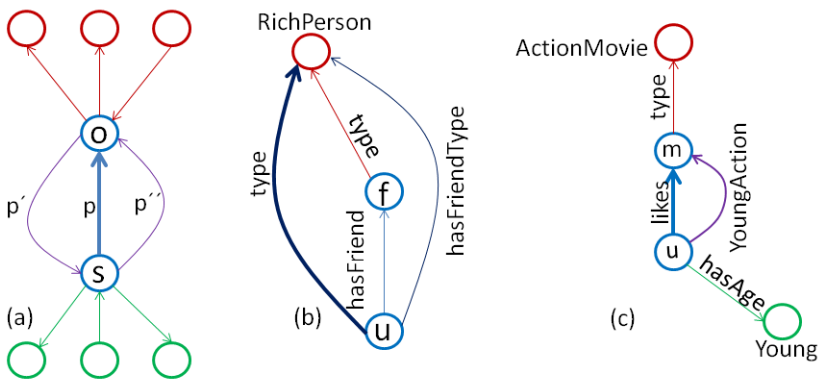

Fig. 2. (a): The goal is to predict the likelihood of the triple (s, p, o). In Equation 1, the triples attached to s correspond to the subject term, the triples attached to o correspond to the object term, and the triples linking s and o correspond the subject-object term. (b): From the triple (u, hasFriend, f) and (f, type, richPerson) we derive via aggregation (s, hasFriendType, RichPerson) which can be useful to predict (u, type, richPerson). (c): From (u, hasAge, Young) and (m, type, ActionMovie) we derive (u, youngAction, m) which is useful for predicting (u, likes, m).

The subject term represents triples where i is the subject, when $l = 1, \dots P$ or where i is the object, when $l = P+1, \dots 2P$. Similar the object term represents triples where k is the subject, when $l = 1, \dots P$ or where k is the object, when $l = P+1, \dots 2P$. The subject-object term considers all triples that involve both i and k with any predicate (see Figure 2 (a)).

3.3 Model Discussion

Note that we assume that the rows in M are exchangeable such that the weights in object term $w_{l,k+(j-1)N}$ are independent of i, and the weights in the subject term $r_{l,i+(j-1)N}$ are independent of k. The parameters $h_{l,j}$ in the subject-object term are independent of both i and k.

There are of course also other ways to segment the parameter space. For example, one might decide that the semantics of the predicate “like” is very different when subject is a person than if the subject is a dog and the object is a bone. Technically this could mean that, e.g., we write $w_{l,k+(j-1)N,type(i)}$ and the model correspondingly would have more parameters.

As a special case, we only have one predicate, i.e. ”like“ and entity types users and movies. If we apply the learning procedure as described in Section 4 we obtain a solution that only exploit correlations between triples with the same predicate, i.e., intrarelational correlations. In effect, we obtain a regularized low-rank approximation of the relational adjacency matrix which is a model often used in collaborative filtering applications. Additional relational adjacency matrixes, for example representing user and movie attributes, can then help to support the prediction of "like"-triples.

One might want to think of the last equation in terms of an if-then-rule where the right side of the equation describes the condition and the left side describes the conclusion. In this view the subject ?s and the object ?o would be variables[4] and the subject term describes relations including the first variable, the object term describes relations including the second variable, and the subject-object term describes relations including both variables. All these variables are universally quantified, which means that the expression is valid for all subjects ?s and all objects ?o. We can introduce additional variables in the condition part for certain aggregation operations, as described in Section 5, and these variables would be existentially quantified (as in Horn clauses).

4 Cost Function and Parameter Optimization

4.1 Penalized Cost Function

We can write the model of Equation 1 efficiently in matrix form as

$$\hat{X} = MW + (MR)^{\dagger} + \mathrm{matrix}_{(N \times PN)}\left(\tilde{M}\ H\right) \tag{2}$$

Here, $\tilde{M} = (\mathrm{vect}(X_1), \ldots, \mathrm{vect}(X_P), \mathrm{vect}(X_1^T), \ldots, \mathrm{vect}(X_P^T))$ is an $N^2 \times 2P$ matrix where the column vector vect(.) contains all elements of the corresponding relational adjacency matrix. Furthermore, $W \in \mathbb{R}^{2PN \times PN}$, $R \in \mathbb{R}^{2PN \times PN}$, and $H \in \mathbb{R}^{2P \times P}$ are parameter matrices. The operation $\mathrm{matrix}_{(N \times PN)}(.)$ transforms the result of the matrix product into a $N \times PN$ matrix.

We define a penalized least squares cost function as

$$\|X - \hat{X}\|_F^2 + \lambda_W \|W\|_F^2 + \lambda_R \|R\|_F^2 + \lambda_H \|H\|_F^2$$

where $\|\|_F$ is the Frobenius norm. The last three terms are used to regularize the solution to avoid overfitting.

4.2 Alternating Least Squares

We optimize the parameter matrices W, R, and H using an alternating least squares procedure as described in this subsection.

To reduce computation and to further regularize the solution, we first decompose using singular value decomposition (SVD)

$$M = UDV^T \qquad \tilde{M} = \tilde{U}\tilde{D}\tilde{V}^T$$

and only use the leading singular values and corresponding singular vectors in the model. Another benefit of this low-rank approximation is that we implicitly

[4] We use the common notation of indicating a variable by a question mark in front of a symbol.

benefit from a sharing of statistical strengths leading to performance improvements, as it is well known from Latent Semantic Analysis (LSA).

We define $\hat{X}^{(-W)}$ as X minus the estimate in Equation 2 using the parameter estimates in the current iteration step, except that $W = 0$, i.e. we remove the subject term in the sum. Similarly, we define $\hat{X}^{(-R)}$ as X minus the estimate in Equation 2 using the parameter estimates in the current iteration step, except that $R = 0$, i.e. we remove the object term in the sum. Finally, we define $\hat{X}^{(-H)}$ as X minus the estimate in Equation 2 using the parameter estimates in the current iteration step, except that $H = 0$, i.e. we remove the subject-object term in the sum.

In the alternating least squares steps we iterate until convergence

$$MW = U_r \,\mathrm{diag}\left\{ \frac{d_i^2}{d_i^2 + \lambda_W} \right\}_{i=1}^{r} U_r^T \hat{X}^{(-W)} \tag{3}$$

$$MR = U_r \,\mathrm{diag}\left\{ \frac{d_i^2}{d_i^2 + \lambda_R} \right\}_{i=1}^{r} U_r^T \left(\hat{X}^{(-R)} \right)^{\dagger} \tag{4}$$

$$\tilde{M}H = \tilde{U}_{\tilde{r}} \,\mathrm{diag}\left\{ \frac{\tilde{d}_i^2}{\tilde{d}_i^2 + \tilde{\lambda}_H} \right\}_{i=1}^{\tilde{r}} \tilde{U}_{\tilde{r}}^T \tilde{X}^{(-H)} \tag{5}$$

where in $\tilde{X}^{(-H)} = \mathrm{matrix}_{N^2 \times P} X^{(-H)}$ each relational adjacency matrix is written as a column vector. In particular for the update in Equation 5, a solution in terms of the V-matrices might be more efficient (see the Appendix).

4.3 Computational Costs

Considering domains with several million entities, the computations seem to be expensive. Fortunately, in the computations one can explore the extreme sparsity of all relational adjacency matrices in many domains of interests. For example, due to type constraints, nonzero elements are often restricted to one or a small number of blocks in the matrices. For example if entities are users and movies and X stands for "likes" then only the submatrix with users as rows and movies as columns contains nonzero elements, reflecting the fact that users like movies but, e.g., movies do not like users. Also, if one is only interested to predict entries in one particular relational adjacency matrix, we only need to calculate the parameters relevant to predicting the entries in that particular matrix.

We also want to point out that one could also apply the SVD to each relational adjacency matrix separately or to blocks of relational adjacency matrices, instead of M; essentially one should make this decision on the expected performance benefits of the matrix decompositions and the computational costs.

5 Extensions

5.1 Aggregation by Join Operations

The triples represented in Equation 2 only consider the immediate neighborhood of the triples (s, p, o) under consideration. It is easy to extend the formalism to also consider triples further away in the graph. As an example, consider the case that the likelihood that a person likes a movie is increased if at least one friend likes the movie. The latter information can be represented by the matrix, representing a join operation, formed by $X_{\text{friendLikesMovie}} = \min(1, X_{\text{friendOf}} X_{\text{likes}})$ where min is applied component wise. Then $X_{\text{friendLikesMovie}}$ (and its transposed) is simply added as an additional relational adjacency matrix. Now we can model (via the subject term in Equation 2) that a person might like "Action Hero 3" if at least one friend likes "Action Hero 3"; the subject-object term in Equation 2 can even model the more general dependency that a person likes any movie if at least one friend likes that movie.

The general form of an aggregated adjacency matrix is $X_a = \min(1, \prod_i M_i)$ where $M_i \in (X_1, \ldots X_P, X_1^T, \ldots X_P^T)$. Naturally, it is not feasible to consider an infinite set of matrix products. Possible approaches are that the user defines a small set of interesting candidates or that one applies structural search, e.g., by using approaches borrowed from the field of Inductive Logic Programming. Many more forms of aggregation are possible. For example one might not apply the min operation and, e.g, count how many friends liked a movie, or what percentage of friends liked a movie.

Here are two interesting examples involving join operations. First, let's assume that a person tends to be rich if this person has a rich friend: The triple of interest is (?u, type, RichPerson). We join (?u, hasFriend, ?f) and (?f, type, RichPerson) and obtain a matrix that indicates if anybody of a person's friends is rich (see Figure 2 (b)). Second, let's assume that a person often prefers restaurants of the nationality of that person: The triple of interest is (?u, likes, ?r). We join (?u, hasNationality, ?c) and (?r, hasNationality, ?c) and obtain a matrix that indicates if the user and the restaurant have the same nationality.

5.2 Contextual and Unstructured Data

Sometimes there is contextual information available, often in textual form, that describe entities and relationships and can be exploited for link prediction [7]. For example, one can use keywords in an entity's Wikipedia articles as attributes of that entity. The triples (s, itsWikiPageHasKeyword, Keyword) can simply be added as an additional relational adjacency matrix in the approach. If a keyword can be identified as an entity, then this information is even more valuable. Information extraction (IE) can also be used to extract triples from text and these triples can then presented in matrix form as well. In the latter case, the subject-object term in Equation 2 can be expected to be most valuable: if, for example, the IE system extracts with high confidence that (Jack, knows, Jane) this could be information for predicting that (Jack, hasFriend, Jane).

5.3 Interaction Terms

In Equation 2 we used a linear system, which is suitable in many high-dimensional domains. Of course, one can apply more general models such as neural networks as predictive models. Often this is unsuitable since the computational costs would explode. In our approach we stay with a model linear in the parameters but add nonlinear interaction terms. As an example, assume that young users prefer action movies. We define a new triple (?u, YoungAction, ?m) that is true if (?u, hasAge, Young) is true and (?m, type, ActionMovie) is true (see Figure 2 (c)). In general, the subject-object term in Equation 1 can be expected to be most valuable here as well. To keep the number of these interaction terms small, we apply a feature selection procedure, as described in Section 6.

5.4 Kernel Formulation

So far our discussion focussed on a representation in feature space. Here we discuss a representation in kernel space. A kernel formulation is appropriate for data in a multirelational graph and suitable kernels are described in [11,4,3,8].

From Equation 2 one can see that two kernels are involved in our approach, the first one $k(.,.)$ involving two entities, either two subjects $k(s, s')$ or two objects $k(o, o')$. The second kernel $\tilde{k}((s, o), (s', o'))$ involves two subject-object pairs. Given the corresponding kernel matrices K and $\tilde{K}$ we can decompose using a singular value decomposition

$$K = UDD^TU^T \quad \tilde{K} = \tilde{U}\tilde{D}\tilde{D}^T\tilde{U}^T$$

and use the resulting terms in the update Equations 3 to 5.

6 Experiments

6.1 Tuning of Hyperparameters

We have several hyperparameters that need to be tuned (rank of approximations; regularization parameters). We follow the approach described in [2] and perform a random search for the best hyperparameters using cross-validation sets (i.e. they are not tuned on the test set).

6.2 Synthetic Data

The synthetic data has been generated according to our modeling assumptions. We define a target predicate of interest and call the triples involving the target predicate the target triples. In addition we have triples related to the subject, i.e., describing subject attributes, and triples related to the object, i.e., describing object attributes. In addition we use interaction triples generated by conjunctions on subject and object triples.

Figure 3 shows the results of using different relational adjacency matrices. The proposed model that uses all sources of information (M_{global}) performs best. Also if we only exploit subject attributes and object attributes (F_{all}) we obtain significant predictive power. A model only using intrarelational information M_{CF} is quite strong. The reason is that, if sufficient amount of target triples are known to be true, the information on subject and object attributes is implicitly modeled in M_{CF} as well. This is a result also confirmed in the remaining experiments: if M_{CF} is quite strong, adding subject and object information does not improve the model further, even when the latter might have predictive power. The proposed model (M_{global}) is significantly better than the reference model (M_{HBS}) described in [7] that used a hierarchical Bayesian combination scheme.

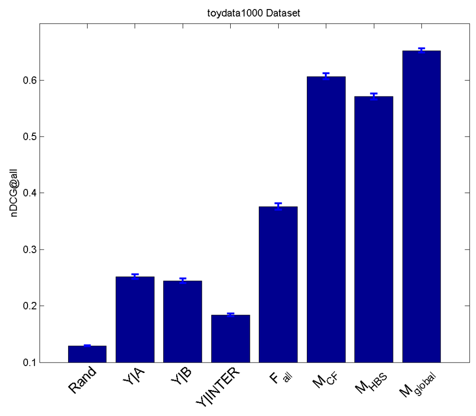

Fig. 3. Test results on synthetic data. For each subject entity in the data set, we randomly selected one true relation to be treated as unknown (test statement). In the test phase we then predicted all unknown relations for the entity, including the entry for the test statement. The test statement should obtain a high likelihood value, if compared to the other unknown entries. The normalized discounted cumulative gain (nDCG@all) [6] is a measure to evaluate a predicted ranking. We see that the proposed method (M_{global}) is significantly better than the model that only relies on intrarelational correlations (M_{CF}). The reference model (M_{HBS}) does not significantly improve w.r.t. M_{CF}. Predictions only based on subject attributes $F_{Y|A}$ only based on object attributes $F_{Y|B}$, and only based on interaction terms $F_{Y|INTER}$ are much better than random. F_{all} uses subject attributes, object attributes and interaction terms, but not intrarelational correlations in the target triples.

6.3 Associating Diseases with Genes

The task here is to predict diseases that are likely associated with a gene based on knowledge about gene and disease attributes and about known gene-disease patterns. In our experiments we extracted information on known relationships between genes and diseases from the LOD cloud, in particular from Linked Life Data and Bio2RDF, forming the triples (Gene, related_to, Disease). In total, we considered 2462 genes and 331 diseases. We retrieved textual information describing genes and diseases from corresponding text fields in Linked Life Data and Bio2RDF.

We have 49801621 potential interaction terms which we reduced to 1132 by using a fast feature selection procedure evaluating the Pearson correlation between targets and interaction term.

Figure 4 (left) shows the results for predicting diseases for genes. Our proposed model gives very good results, although the reference model is slightly stronger. Figure 4 (right) shows the results for predicting genes for diseases. Due to sparsity, this task is more difficult and our proposed model performs best.

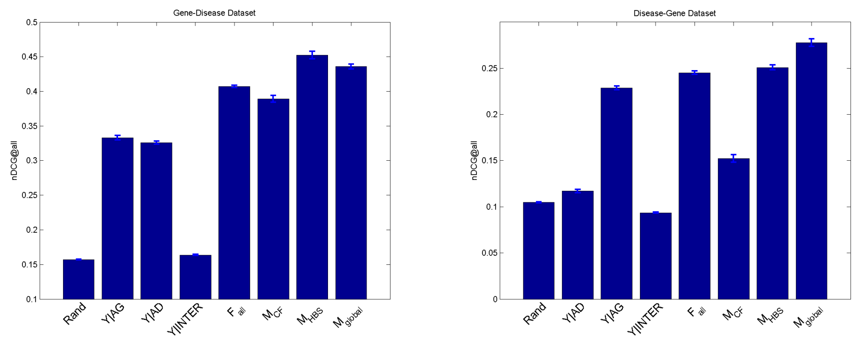

Fig. 4. The goal is to predict the relationship between genes and diseases. On the left we ranked recommended diseases for genes and on the right we ranked genes for diseases. In the left experiment, the subject attributes of the genes $F_{Y|AG}$, and of the object attributes of the diseases $F_{Y|AD}$ are comparable in strength. F_{all} that uses gene attributes, disease attributes and interaction terms in combination gives strong results. Our proposed model (M_{global}) can exploit both contextual information and intrarelational correlations. The reference model (M_{HBS}) is slightly stronger than our proposed model. The right plot shows results from the second experiment where we rank genes for diseases. This task is more difficult due to the large number of genes and our proposed system gives best results.

6.4 Modeling MovieLens Data

We used 943 users and 1600 movies from the MovieLens data set[5] and evaluated if a user has seen a movie or not. 99 user attributes were derived from age (5 classes), gender (2 classes), occupation (21 classes), and the first two digits of the ZIP code. The 89 movie attributes were derived from genre, release month and release year. Figure 5 (left) shows the results. Although the attribute information on the movies and the users have predictive power (significantly above random), a model exploiting intrarelational correlations (M_{CF}) gives very good performance and the proposed model and the reference model cannot improve beyond the performance of M_{CF}. As in the experiment on the synthetic data,

[5] http://www.grouplens.org/node/73

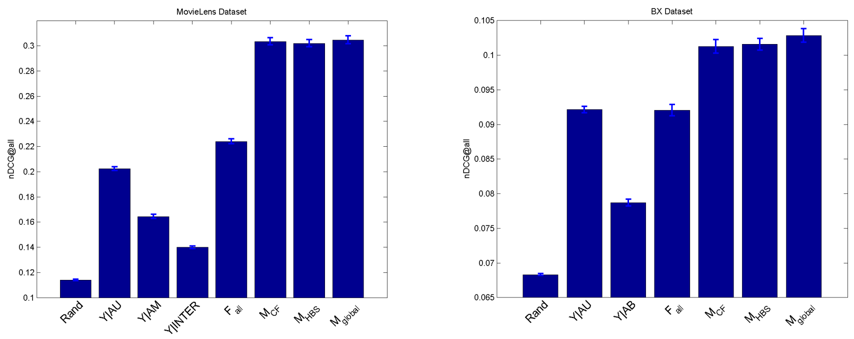

Fig. 5. Left: Experiments on movielens data. $F_{Y|AM}$ describes the modeling performance using only the attribute information on the movies and $F_{Y|AU}$ describes the modeling performance using only the attribute information on the users. Although the attribute information on the movies and the users have predictive power (significantly above random), a model exploiting intrarelational correlations (M_{CF}) gives very good performance and the proposed model and the reference model cannot significantly improve beyond the performance of M_{CF}. As in the experiment on the synthetic data, there is information in the attribute data but this information is also represented in M_{CF}. Right: Experiments on the book-crossing data set. We see the same trends as in the movielens experiments although the proposed approach seems to improve on the model M_{CF}, which only exploits intrarelational correlations.

there is information in the contextual data but this information is also represented in M_{CF}. We have 8811 potential interaction terms which we reduced to 200 by using a fast feature selection procedure evaluating the Pearson correlation between targets and interaction term.

6.5 Modeling Book Preferences

We used the BookCrossing data set[6] to predict if a user rated a book. The data set consisted of 105283 users and 340554 books. A user is described by 5849 attributes (derived from age and city, province and country) and a book is described by 24508 attributes (authors, publication year, publisher). The goal is to predict if a user would rate (i.e., read) a book. The results in Figure 5 (right) show that the proposed modeling approach gives best results.

7 Conclusions

We have presented a general framework for predicting links in multirelational graphs. We showed that efficient learning can be achieved using an alternating least squares approach.

[6] http://www.bookcrossing.com

The approach can be extended in several directions. First, for the entries in the relational adjacency matrices one can use real numbers, e.g., between zero and one, and the user can represent the certainty that a triple is true [7]. Second, we can exploit deductive reasoning by calculating the deductive closure prior to learning [7]. Third, the prediction in Equation 1 can be applied recursively permitting global information flow through the relational graph. Finally, we can easily generalize to entities not in the training set, either by using Equations 3 to 5 directly or by transforming these equations into appropriate equivalent forms.

References

1. Robert M. Bell, Yehuda Koren, and Chris Volinsky. All together now: A perspective on the netflix prize. *Chance*, 2010.
2. James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 2012.
3. Stephan Bloehdorn and York Sure. Kernel methods for mining instance data in ontologies. *ESWC*, 2007.
4. Thomas Gärtner, John W. Lloyd, and Peter A. Flach. Kernels and distances for structured data. *Machine Learning*, 2004.
5. Yi Huang, Markus Bundschus, Volker Tresp, Achim Rettinger, and Hans-Peter Kriegel. Multivariate structured prediction for learning on the semantic web. In *ILP*, 2010.
6. Kalervo Järvelin and Jaana Kekäläinen. IR evaluation methods for retrieving highly relevant documents. In *SIGIR'00*, 2000.
7. Xueyan Jiang, Yi Huang, Maximilian Nickel, and Volker Tresp. Combining information extraction, deductive reasoning and machine learning for relation prediction. In *ESWC*, 2012.
8. Ute Lösch, Stephan Bloehdorn, and Achim Rettinger. Graph kernels for RDF data. *ESWC*, 2012.
9. Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *ICML*, 2011.
10. Volker Tresp, Yi Huang, Markus Bundschus, and Achim Rettinger. Materializing and querying learned knowledge. In *IRMLeS*, 2009.
11. S. V. N. Vishwanathan, Nic Schraudolph, Risi Imre Kondor, and Karsten Borgwardt. Graph kernels. *Journal of Machine Learning Research - JMLR*, 2008.

8 Appendix

We only derive Equation 3. The derivations for Equations 4 and 5 are equivalent. We start with the regularizes least squares solution for estimating $\hat{X}^{(-W)}$ based on M

$$MW = M(M^T M + \lambda_W I)^{-1} M^T \hat{X}^{(-W)}$$

If we use the low-rank approximation $M \approx U_r D_r V_r^T$, where $D_r = \text{diag}\,\{d_i\}_{i=1}^r$, we get

$$MW = U_r D_r V_r^T (V_r D_r U_r^T U_r D_r V_r^T + \lambda_W I)^{-1} V_r D_r U_r^T \hat{X}^{(-W)}$$

$$= U_r \text{ diag} \left\{ \frac{d_i^2}{d_i^2 + \lambda_W} \right\}_{i=1}^r U_r^T \hat{X}^{(-W)} = \hat{X}^{(-W)} V_r \text{ diag} \left\{ \frac{d_i^2}{d_i^2 + \lambda_W} \right\}_{i=1}^r V_r^T$$