Joint Similarity Learning for Predicting Links in Networks with Multiple-type Links

Majid Yazdani
Idiap Research Institute and EPFL
Centre du Parc, Rue Marconi 19
1920 Martigny, Switzerland
majid.yazdani@idiap/epfl.ch

Andrei Popescu-Belis
Idiap Research Institute
Centre du Parc, Rue Marconi 19
1920 Martigny, Switzerland
andrei.popescu-belis@idiap.ch

ABSTRACT

This paper addresses the problem of link prediction on large multi-link networks by proposing two joint similarity learning architectures on nodes' attributes. The first model is a similarity metric that consists of two parts: a general part, which is shared between all link types, and a specific part, which learns the similarity for each link type specifically. The second model consists of two layers: the first one, which is shared between all link types, embeds the objects of the network into a new space, while the second one learns the similarity between objects for each link type in this new space. The similarity metrics are optimized using a large-margin optimization criterion in which connected objects should be closer than non-connected ones by a certain margin. A stochastic training algorithm is proposed, which makes the training applicable to large networks with high-dimensional feature spaces. The models are tested on link prediction for two data sets with two types of links each: TED talks and Amazon products. The experiments show that jointly modeling of the links given our frameworks improve link prediction performance significantly for each link type. The improvement is particularly higher when there are fewer links available from one link type in the network. Moreover, we show that transfer learning from one link type to another one is possible using the above frameworks.

1. INTRODUCTION

The problem of new link prediction in networks is particularly relevant to networks that are collaboratively built over time, whether they are document-based (such as Wikipedia, or databases of products or multimedia records) or person-based (social networks such as Facebook or LinkedIn) or both. In such situations, some nodes become connected over time due to similarities or affinities that are noticed and validated by users. Predicting such connections is a functionality which is valuable to speed up network construction, for instance by presenting these connections as recommendations to users. This is particularly challenging when links are of multiple types. Moreover, accurate modeling of the connections in the network can assist prediction of the phenomena like evolution of the network or even marketing via the network.

We introduce here two joint link prediction models for networks with multiple link types. Link prediction is formulated as a learning to rank problem: given a query node, all other nodes must be sorted according to a score that is related to the likelihood of creating a link between them and the query node. The proposed link prediction models in this paper learn a similarity metric according to which connected nodes are closer than non-connected nodes.

Two different joint models are proposed. The *shared similarity model* consists of two parts: the general part, which is a shared similarity function between all types of links, and the specific part, which learns similarity specifically for each type of links. The *two-layer similarity model* consists of two layers: in the first layer, objects in the network are embedded into a new space, while the second layer learns the similarity between objects for each link type specifically in this new space. The first layer is shared between all link types and can also be considered as a representation of objects which is common across all link types.

The training algorithm is applicable to large networks with high-dimensional features and was tested on two data sets: one built from TED talks (1150 items) and one built from Amazon products (10,000 items), both with two types of links. The results show that our joint modeling and training frameworks improve link prediction for each link type, in comparison to several state-of-the-art models.

This paper is organized as follows. In Section 2 we review previous related work and highlight the main differences with our proposal. In Section 3 we introduce the joint models for multiple-type links, first by defining the shared similarity model, and then by defining the two-layer similarity model. We then propose a large-margin method for training the similarity functions (3.4). We also show how to generalize the proposed frameworks for the related problems including jointly model link prediction and classification in a network (3.5). The experiments described in Section 4 on the above-mentioned networks demonstrate the predictive power of the joint model, in comparison with separate models, as well as with SVM Rank and cosine similarity. Moreover, transfer from one link type to another one enables prediction when no training data from one type is available.

2. RELATED WORK

Related work on this topic falls into two main categories: learning to rank and link prediction which both have been studied in the recent past. The main advantage of our model over previous ones is that we model large networks with multiple-type links, also called *multi-link networks*, by sharing information between different link types.

2.1 Link Prediction

Authors in [13] discuss the link prediction task as a ranking task over pairs of nodes, based on link structure similarity metrics, and demonstrate the difficulty of the link prediction task. They show in particular that the Adamic and Adar [1] similarity measure, which makes use of common neighborhoods, yields high performance in comparison to other link-based approaches for link prediction. A shortcoming of this work is that all studied approaches are based on the link structure and, therefore, attribute information – for nodes and for edges – is not exploited. This is more problematic when the query node does not have many known links in the network. Authors in [2] proposed a supervised learning method for ranking the objects in a graph. Their method uses a random walk model and learns the transition probabilities from the ordered pairs of objects in the training data. A transition probability is learned for each link, which makes overfitting likely and is not effective for large-scale graphs with many edges, given that there are many parameters to learn. Supervised random walk [3] overcomes one of the main shortcomings of the link structure based similarity metrics by using attribute information to make predictions. A weight function for each edge is trained over nodes' attributes in a way that the probability resulting from random walk for the connected nodes be higher than non-connected nodes. This approach, along with the other approaches above, is not yet applicable when the query object is not part of the network (i.e. it does not have any link) as it is supervised generalization of random walk model.

Authors in [14] are pursuing a generative Bayesian nonparametric approach to simultaneously infer the number of latent features and to learn which entities have each feature. The method is difficult to train, and inference for large scale graphs is time-consuming. Relational Topic Models (RTM) [8] consider both the documents and the links between them. For each pair of documents, an RTM models their link as a binary random variable that is conditioned on their contents. The inference and learning algorithms are based on variational methods. The original RTM algorithm is not applicable to large graphs, although it is possible to consider an online algorithm for training the model.

In contrast to the above-mentioned methods, the main advantage of our proposed models is jointly modeling of the links in the multi-link networks, which is not clear how to address in those frameworks. Furthermore, learning similarity over query and target nodes' attributes makes our models effective in comparison to random walk based methods when the query object does not have many known links in the network .

2.2 Distance Metric Learning for Learning to Rank

Perception Ranking [9] is an online algorithm for ordinal classification which can be employed for ranking as a pointwise method. Its main idea is to learn several parallel perceptron models which make classification between the neighboring grades. The main difference of this algorithm with our ranker is that we use a pairwise method, which is shown to be more effective than pointwise methods.

IR SVM [7] is a pairwise method which formulates the ranking problem as an SVM classification , and adapts this to the document retrieval problem. The feature selection that transforms the ranking problem into a classification one – building features for classification from the query and each target document – is not effective on all data sets. This is especially problematic for a task such as link prediction, where we have many non-linked examples. Similarly SvmRank [11] is a pairwise ranking algorithm which transforms the pairwise ranking to svm binary classification. For SvmRank, batch optimization on large graphs is not possible considering the number of non-connected pairs.

Authors in [4] perform supervised training of a nonlinear model over vectors of words, to preserve a pairwise ranking between documents. Their approach scales well to large data sets with a large number of words. Similarly, authors in [16] train a distance metric by stochastic gradient descent over a hinge loss function that preserves the network structure. We will follow the same general line to build our ranking method based on similarity learning, by keeping in mind that the framework should be applicable to large graphs. In addition, we allow different link-type rankers sharing information among them.

3. JOINT SIMILARITY FOR MULTIPLE-TYPE LINK PREDICTION

Link prediction can be viewed as a ranking problem in which all objects in the network are ranked based on their similarity score with respect to a query object. A "better" ranking is one that places objects that are actually linked to the query object at the top of the ranked list. The score between the two objects in the network can be considered as a similarity metric (or, conversely, a distance one), and link prediction can be interpreted as a task in which linked objects should be closer to the query object, in comparison to the non-linked objects.

We model the link structure of a network by learning a similarity measure which, for each object in the network, assigns larger scores to the objects that are linked to it than to those that are not. To model N different types of links, we train N different similarity functions, so that we model the link structure of each type separately. However, in many real-world networks, links of a certain type are not entirely independent from links of other types. To consider this dependency, each link-type similarity model shares information with the other models. We will show in Section 4 that the joint modeling increases generalization abilities, hence link prediction performance, especially when there are few links from one specific link type.

3.1 Similarity Learning Framework

In this section, we build a similarity based ranker to model links in the network. The ranker returns a score for a given query object q and a target object t by using a similarity function over their features. If $f_i(q,t)$ represents the ranker for link type i, and x_o is the feature vector of an object o from the network, then we have:

$$f_i(q,t) = similarity_i(x_q, x_t)$$

The $similarity_i$ function computes a similarity for link type i between objects q and t using their attributes, following a classical approach for learning to rank methods. Many similarity functions (or, equivalently, distance metrics) with various learning abilities have been studied in the literature, for instance RankNet [6], polynomial semantic indexing [5] or structure preserving metric learning [16] (see also Section 2 above). In this study, we use inspiration from these previous studies and define the similarity function as follows:

$$similarity_i(x_q, x_t) = x_q \times M_i \times x_t'$$

where matrix M_i is a $Z \times Z$ matrix, Z being the size of the features, and x_t' is the transpose of x_t. The $similarity_i$ function is not necessarily symmetric, which makes it compatible with networks with directed links. In practice, to make training and storage possible, particularly when dealing with high-dimensional data such as text in a word vector representation, a low-rank factorization of M is considered for training [5, 16]. Each ranker in the above formulation is ignorant about the other link types, therefore we call this model separate model and it is used in the experiments section as a baseline. We introduce two models by extending this approach to share information among rankers.

3.2 Shared Similarity Model

To introduce joint modeling among rankers, and share information between them, we assume that each matrix M_i consists of two parts: a general matrix noted G, which is shared between all rankers, and a specific matrix noted S_i, which is learned separately for each link type i. Hence, $M_i = G + S_i$. The similarity function can thus be formulated as:

$$similarity_i(x_q, x_t) = x_q \times (G + S_i) \times x_t' =$$
$$= \underbrace{x_q \times G \times x_t'}_{\text{General Similarity}} + \underbrace{x_q \times S_i \times x_t'}_{\text{Specific similarity for the type } i}$$

The similarity function thus in its turn consists of two parts: first, a general similarity measure which is shared (and identical) across link types and second, the specific similarity measure which is adapted to each link type. Matrices G and S_i s are the parameters of the similarity function which are going to be trained.

Matrix G would represent the correlation between link types. To better explain the role of G let us consider a network with two link types and extreme hypothetical cases: when the two link types are independent, and when they are identical. If the two link types are independent, then the matrix G will be the 0 matrix and the model would be equivalent to the separate model introduced in the previous section. On the other hand, if the two link types are identical, then matrix G is identical to the link specific matrices S_i. For the other situations between these two extreme cases, matrix G would be trained to represent the correlation between the two link types.

However, this model is able to handle dependencies between the link types only if they are positively correlated. In cases where two link types in the network are negatively correlated, this correlation would not be modeled by G since having a link of the first type between two objects prevents us from having the other type of link between them and G would be simply 0. For example consider a network consisting of researchers and two types of collaboration between them: first intra-institute collaboration and second, inter-institute collaboration. If there is a link of the first type between two researchers in the network, then the second link type can not exist between them. This model can not represent this negative correlation between link types and is equivalent to the separate model on this network.

The above-mentioned problem is the main drawback of our first model. In the following section we introduce a two-layer similarity model to overcomes this problem.

3.3 Two-layer Similarity

The two-layer similarity model consists of two layers: the first layer embeds objects in the network to a new space, and then the similarity for each link type is learned specifically in this new space. The first layer which embeds the objects to a new space is the same for all link types. The first layer can be viewed as a *shared representation* for objects among all link types. Figure 1 shows schematically the two-layer similarity model.

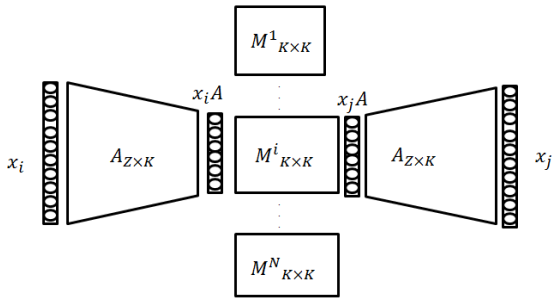

Figure 1: A schematic representation of the two-layer similarity model. Matrix A embeds the objects in the network to a K-dimensional space. Matrix M^i computes the similarity between objects for link type i.

The matrix $A_{Z\times K}$ transforms the objects to a new space with dimension K where Z is the dimension of objects' features and matrix M_i is the matrix representing the similarity for the link type i in the new space. The similarity function can thus be formulated as:

$$similarity_i(x_q, x_t) = \underbrace{(x_q \times A)}_{\text{Shared representation}} \times (M_i) \times \underbrace{(x_t \times A)'}_{\text{Shared representation}}$$

Matrices A and M_i are the parameters of the model which are going to be trained. This two-layer model overcomes the main problem of the previous model, since it can also model negatively correlated link types through the shared

representation. To explain intuitively how this can be done assume again the hypothetical network of researchers and the two types of relations between them. If we assume there are two institutes in our network, the first layer could potentially embed objects into the two clusters in the latent space corresponding to the two institutes. Then the similarity matrices M_i in the latent spaces would learn that first type similarity is higher between the objects in the same cluster, and second type similarity between objects in different clusters. If there exist a first link type between two researchers, then they are embedded in the same cluster and the score of the second link type drops, therefore the shared representation enables joint modeling of even negatively correlated link types.

3.4 Training the Joint Models

We can formulate the learning criterion for a specific link type as a pairwise ranking problem, in which given an object, other objects that are linked in the training data should be ranked higher than non-linked ones. Below, we formalize the learning-to-rank problem and then place the joint rankers introduced in the previous section in this framework.

We make use of a loss function noted $L(\cdot,\cdot)$ to evaluate the prediction result of ranking upon training. The feature vectors are ranked according to their scores, and the resulting ranking is evaluated against the corresponding expected links (known in the training set). If the feature vectors of linked objects are ranked higher, then the loss will be small, otherwise it will be large.

We define here a pairwise hinge loss function L, which attempts to preserve the pairwise order between objects in the training set by maintaining a margin between the scores of each pair [4]. For instance, in a friendship social network, the hinge function (computed only using feature vectors) aims to preserve the order between the scores of the friend pairs and the non-friend pairs, scoring all friends higher than all non-friend. Similarly, in the case of hyperlinked documents, the score of the linked documents pairs should be higher, with a certain margin, than the score of the non-linked documents.

The goal of training is to approximate the parameters of a similarity function for rankers which minimizes the pairwise hinge loss function L over the network $\mathcal{G}$ with the various types of links that are given in the training set. Training is performed over the graph $\mathcal{G} = (V, E_1 \cup E_2 \cup \dots E_N)$, where V is the set of vertices (objects) and E_i is the set of edges (links) of type i.

3.4.1 Training the Shared Similarity Model

The empirical risk minimization by using a pairwise hinge loss function over the training set is as follows. We start from training sets for each link type i:

$$Train_i = \{(a,b,c) | (a,b) \in E_i \wedge (a,c) \notin E_i\}$$

The goal is to minimize the loss function L as follows:

$$Min \quad L = \frac{\lambda}{2}||G||_F^2 + \frac{\lambda}{2}\sum_i ||S_i||_F^2 + \frac{\sum_i \sum_{(a,b,c)\in Train_i} max(0, d - f_i(a,b) + f_i(a,c))}{(\sum_i |Train_i|)}$$

The first two terms, the Frobenius norms of matrices G and S_i, enforce a regularization on the parameters and prevent the arbitrary increase of the parameters in the optimization. The third term of the loss function maintains the order, with a margin d, between the scores of the connected nodes and the unconnected nodes for link type i.

If we consider the definition of f_i from the previous section and the decomposition of M_i into G and S_i, then the subgradients of the loss function with respect to G and to S_i are the following:

$$\bigtriangledown L(G) = \lambda G + \frac{\sum_i \sum_{f_i(a,b)-f_i(a,c)<d} x'_a \times (x_c - x_b)}{\sum_i |Train_i|}$$

$$\bigtriangledown L(S_i) = \lambda S_i + \frac{\sum_{f_i(a,b)-f_i(a,c)<d} x'_a \times (x_c - x_b)}{|Train_i|}$$

If the training graph $\mathcal{G}$ is large, then minimizing the above summation is not tractable. To overcome this problem we make use of a stochastic gradient descent algorithm in which at each iteration we select N samples (a,b,c), one randomly from each $Train_i$ set, and perform gradient descent on each of them. As we are interested in modeling large scale graphs, we will always make use of this stochastic gradient descent algorithm in the rest of the paper. The subgradients of the approximate loss function at iteration t are the following, for an iteration over one of the samples $(a,b,c) \in Train_i$:

$$\bigtriangledown L_t(G) = \lambda G^t + \begin{cases} x'_a(x_c - x_b) & \text{if } f_i(a,b) - f_i(a,c) < d \\ 0 & \text{otherwise} \end{cases}$$

$$\bigtriangledown L_t(S_i) = \lambda {S_i}^t + \begin{cases} x'_a(x_c - x_b) & \text{if } f_i(a,b) - f_i(a,c) < d \\ 0 & \text{otherwise} \end{cases}$$

The update of the parameters is done consequently as follows:

$$G^{t+1} = G^t - \eta_t \bigtriangledown L_t(G)$$
$${S_i}^{t+1} = {S_i}^t - \eta_t \bigtriangledown L_t(S_i)$$

where η_t is the learning rate at the iteration t.

Analysis of the training algorithm. The first observation is that the time complexity of each iteration of the stochastic gradient algorithm is $O(N_Z^2)$ where N_Z is the average of the number of non-zero features of the x_a, x_b and x_c. In the textual domain, the number of non-zero features N_Z is usually a couple of hundreds, which is much smaller than the dimension of a word vector representation of the data, which is usually several hundred thousands (number of all possible words).

We can follow the same approach that is used to prove the convergence of the PEGASOS algorithm in [15] in order to prove the convergence of the training algorithm presented

above. In fact, we can show that the above algorithm is a special case of PEGASOS. Prove of the convergence is in Appendix .1.

Low-Rank Factorization. The dimension of G and S_i is Z^2 where Z is the number of features of the data points. In practice, to make training and storage possible, particularly when we are dealing with high dimension data such as text, a low-rank factorization of G and S_i should be considered. In this case we assume that $G = AA'$ and $S_i = S_{i1}S'_{i2}$ so that A, S_{i1} and S_{i2} are matrices from Z to a lower dimension. Given that M_i is not necessarily symmetric we decompose the specific part S_i to two lower-rank matrices. Also we change the regularization on G and S_i with the regularization on A, S_{i1} and S_{i2} in the objective function. The objective function consisting of the low-rank matrices is not convex any more, but in practice it has been shown that the low-rank factorization performs well [5]. The subgradients are as following for a given sample $(a, b, c) \in \mathit{Train}_i$:

$$\nabla L_t(A) = \lambda A^t + \begin{cases} x'_a(x_c - x_b)A^t + (x_c - x_b)' x_a A^t & \text{if } f_i(a,b) - f_i(a,c) < d \\ 0 & \text{otherwise} \end{cases}$$

$$\nabla L_t(S_{i1}) = \lambda S_{i1}^t + \begin{cases} x'_a(x_c - x_b)S_{i2}^t & \text{if } f_i(a,b) - f_i(a,c) < d \\ 0 & \text{otherwise} \end{cases}$$

$$\nabla L_t(S_{i2}) = \lambda S_{i2}^t + \begin{cases} (x_c - x_b)' x_a S_{i1}^t & \text{if } f_i(a,b) - f_i(a,c) < d \\ 0 & \text{otherwise} \end{cases}$$

3.4.2 Training two-layer Similarity Model

Similarly to the previous section, for the two-layer model the empirical risk minimization using a pairwise hinge loss function over the training set is as follows. We start from training sets for each link type i:

$$\mathit{Train}_i = \{(a, b, c) | (a, b) \in E_i \wedge (a, c) \notin E_i\}$$

The goal is to minimize the loss function L as follows:

$$Min \quad L = \frac{\lambda}{2}||A||_F^2 + \frac{\lambda}{2}\sum_i ||M_i||_F^2 + \frac{\sum_i \sum_{(a,b,c)\in \mathit{Train}_i} max(0, d - f_i(a,b) + f_i(a,c))}{(\sum_i |\mathit{Train}_i|)}$$

Again, the first two terms enforce a regularization on the parameters and prevent the arbitrary increase of the parameters in the optimization. If we consider the definition of f_i then the subgradients of the loss function with respect to A and to M_i are the following:

$$\nabla L(A) = \lambda A + \frac{\sum_i \sum_{f_i(a,b)-f_i(a,c)<d} (x'_a((x_c - x_b)A) + (x_c - x_b)'(x_a A)) \times M_i}{\sum_i |\mathit{Train}_i|}$$

$$\nabla L(M_i) = \lambda M_i + \frac{\sum_{f_i(a,b)-f_i(a,c)<d} (x_a \times A)'((x_c - x_b) \times A)}{|\mathit{Train}_i|}$$

If the training graph $\mathcal{G}$ is large we make use of a stochastic gradient descent algorithm in which at each iteration we select N samples (a, b, c), one randomly from each Train_i set, and perform gradient descent on each of them. The subgradients of the approximate loss function at iteration t are the following, for an iteration over one of the samples $(a, b, c) \in \mathit{Train}_i$:

$$marg = f_i(a,b) - f_i(a,c)$$

$$\nabla L_t(A) = \lambda A^t + \begin{cases} (x'_a((x_c - x_b)A^t) + (x_c - x_b)'(x_a A^t)) \times {M_i}^t & \text{if } marg < d \\ 0 & \text{otherwise} \end{cases}$$

$$\nabla L_t(M_i) = \lambda {M_i}^t + \begin{cases} (x_a \times A^t)'((x_c - x_b) \times A^t) & \text{if } marg < d \\ 0 & \text{otherwise} \end{cases}$$

The update of the parameters is done consequently as follows:

$$A^{t+1} = A^t - \eta_t \nabla L_t(A)$$
$${M_i}^{t+1} = {M_i}^t - \eta_t \nabla L_t(M_i)$$

where η_t is the learning rate at the iteration t.

Analysis of the training algorithm. The time complexity of each iteration of the stochastic gradient algorithm is $O(K^2 + N_Z \times K)$ where N_Z is the average of the number of non-zero features of the x_a, x_b and x_c and K is the size of latent space. The loss function is not convex in the two-layer similarity model and, therefore, the initialization of the parameter can have a significant effect on the performance.

3.5 Generalization for Related Problems

In this section we show that the above architectures for joint link prediction on multi-networks can be extended for the similar problems.

3.5.1 Joint Classification and Link Prediction

We show that a joint classification and link prediction task over a network can be reformulated as a multi-link prediction task, thus enhancing the generality of our proposal. Indeed, in most real-world networks, objects are labeled with a certain finite label set, and these labels are not independent from the link structure. For example, in a papers/citations network, it is more likely that machine learning papers cite other machine learning papers rather than biology papers. Conversely, a paper which cites many machine learning papers is more likely to be a machine learning paper rather than a biology paper.

We introduce here a new link type based on the similarity of objects' labels: every two objects in the network that have the same label will be connected by such a link. Therefore, the classification and the link prediction tasks can be casted to a multi-type link prediction problem, in which one of the link types is based on the similarity of objects' labels. Consequently, the goal of learning for this task is to find similarity functions for which objects with the same labels are closer than the other objects, and (as above) linked objects are closer than non-linked objects. For the classification task, i.e. to determine the label of a yet unlabeled object, we consider the majority label of the closest objects to that object.

The empirical risk minimization for joint classification and link prediction by using a pairwise hinge loss function over the training set is as follows. To simplify notations, we assume that there is only one link type i in the network, and we note $\mathcal{C}(o)$ the label or class of an object o. First, the training sets are defined as:

$$\begin{aligned} Train_L &= \{(a,b,c)|(a,b) \in E, (a,c) \notin E\} \\ Train_C &= \{(a,b,c)|\mathcal{C}(b) = \mathcal{C}(a), \mathcal{C}(c) \neq \mathcal{C}(a)\} \end{aligned}$$

Then, training is defined as minimizing the loss function L :

$$Min \quad L = \lambda \times Regularization\ on\ the\ parameters + \frac{\sum_{(a,b,c) \in Train_L \cup Train_C} max(0, d - f_i(a,b) + f_i(a,c))}{(|Train_C| + |Train_L|)}$$

Both models introduced above can be used as the similarity model in the above formulation. In the shared similarity model case we consider a general similarity function which is shared between classification and link prediction tasks, along with specific similarity functions for each task separately. In the two-layer model the first layer is the shared representation between classification and link prediction task, and the second layer shows the specific similarity in the embedded space. The training algorithm is identical to the case of link prediction on the multi-type links networks, which was discussed above.

3.5.2 Networks without attributes

Throughout this paper, we assume that the similarity functions are learned over node attributes. However, in networks where *nodes do not have attributes* or attributes are not predictive , the proposed framework can still be applied. For each node, a feature vector is defined of the size of all nodes in the graph, in which only the element corresponding to the node has value 1 and the others are set to 0.

In this case, the low rank factorization of the similarity matrix in shared similarity model, or the first layer in the two-layer model is similar to the existing methods for network matrix decomposition, such as singular value decomposition or non-negative matrix factorization, except that our method enforces the large margin criterion and is trained jointly. The empirical exploration of this case is the topic of future work.

3.5.3 Inter-Network transfer learning

We show in this paper that we can transfer the learning from one link type to another link type through the shared information part. *Transferring learning from one network to another one* is another interesting and related problem. Considering two different online social networks, one based on friendship (such as Facebook) and the second one based on professional relations (such LinkedIn), the model described here could be applied to share the learning between the two networks as long as it is possible to build identical feature vectors for the nodes in both networks. This problem also deserves further investigations.

4. EXPERIMENTAL RESULTS

Many real-world networks have multiple types of links. For example, in online social networks, people have different types of relationships and interactions. In many such networks, especially those that are collaboratively built, some objects often have fewer links from one link type. In this section, we consider two networks: TED videos and Amazon products. We show that our joint link prediction algorithm improves the link prediction performance, especially when there are fewer links from one link type.

4.1 Network of TED Talks

TED is an online audio-visual broadcasting platform for the TED talks. [1] We consider a network consisting of nearly 1,200 TED talks, with metadata and transcripts, linked by two types of links. The first type relate two talks based on the similarity of contents. We derive such content links from the suggestions of similar talks made by experts from TED. The second type of links connects two videos if they co-exist in many users favorite lists. Since less popular talks do not appear in many people's favorite lists, there are fewer links based on users favorite lists for such talks, and presumably the links for these talks are not very reliable. Note that recognizing automatically that two talks have similar contents is difficult, but inference about the content can be done more easily when two talks co-occur in many people's favorite lists.

More specifically, the second type of links based on users' favorite lists is built as follows. Each talk is represented by a vector in the space of all users, and an element of the vector is 1 if the talk is in the favorite list of the corresponding user, and 0 otherwise. Given one specific talk, the rest of the talks can be sorted according to cosine similarity between their vectors and the initial talk in the space of users. If two talks co-exist in many users' favorites list, the cosine similarity between their vectors in the space of users is high. Therefore, to build the second type of links, we connect each talk with the top k talks based on cosine similarity between their vectors in the users space. We will call this type of links "co-liked".

For example, the TED talk "Lucien Engelen: Crowdsource your health" has two content links (two recommendations of topically-related talks from TED experts): "Jacqueline Novogratz on patient capitalism" and "Nicholas Christakis:

[1]This data set has been collected by Nikolaos Pappas from Idiap, who has kindly shared it for the work presented here. See http://www.ted.com for the original website.

How social networks predict epidemics". The three highest score co-liked talks are: "Marco Tempest: Augmented reality, techno-magic", "Aparna Rao: High-tech art (with a sense of humor)" and "Sheikha Al Mayassa: Globalizing the local, localizing the global". We can observe that the related videos are about the same topic, but co-liked videos are much less predictable from the content.

We build a feature vector for each talk from three sources: (1) speaker name, (2) title of the talk, and (3) talk description as provided by TED (a short text). Each feature vector has a dimension of 4,771 (based on a filtered vocabulary of 4,771 words), but vectors are very sparse, on average with only 35 non-zero coefficients.

4.2 Network of Amazon Products

We build a network from a subset of Amazon products (described in [12]) with two types of links between them. The first type comes directly from the Amazon website: for each product, some of the co-purchased products are shown by Amazon. Each product in the network is connected to the products which are claimed to be mostly co-purchased by the Amazon website. We call this type of links "co-purchased".

Each product is assigned to at least one category, often to more. Categories form a hierarchy, from more general ones to the most detailed ones. We represent each product as a vector in the space of all categories: if the product is assigned to a category, the corresponding element is 1, otherwise it is 0. Cosine similarity between category vectors of two products is high if they are mostly in the same categories from the hierarchy. To build this type of the links, we connect each product to the k most similar products using cosine similarity between the category vectors of the products. We call this type of links "category links".

For example, consider the book "Writers in the Schools: A Guide to Teaching Creative Writing in the Classroom". This book is co-purchased with the following other books: "The Magic Pencil: Teaching Children Creative Writing [...]" and "Journal Jumpstarts: Quick Topics and Tips for Journal Writing". Two category links are "Reading, Writing, and Learning in ESL (2nd Edition)" and "Time to Know Them: A Longitudinal Study of Writing and Learning at the College Level".

Features of each product in the network come from the title and description of the product. We use a dictionary of 3,765 words and sampled 10,000 products to build our network of products. So, our second network is about eight times larger than the first one.

4.3 Prediction Performance

To study the link prediction ability of our models experimentally, we split the objects of each network into disjoint training and test sets. The training is performed on the objects in the training set and the links between them, excluding links from these objects to the test set. To evaluate prediction performance, for each object in the test set, all the objects in the entire data set (training plus test sets) are ranked by the ranker, and the Mean Average Precision (MAP) is calculated for the resulting ranked list according to the links in the test set. The final scores are the average of 5-fold cross validation, with 80% of the data used for training and 20% for testing, in each fold.

We are interested in observing the prediction performance of the joint model when there are few available links from one link type. To perform such an analysis, we make available to the joint learner a full training set for one link type (80% of the objects in the network), and a variable-sized training set for the other link type, varying from a smaller subset (30% of the nodes) to the full training set (80%) by fixed increments (10%). The test set is always fixed (20%). We observe the evolution of prediction performance with the size of the training set for the second type of links.

Table 1 shows the average MAP for the objects in the test set for the co-purchased links, in the Amazon products network, for several models. As explained, the size of the co-purchase training set varies from 30% to 80% of the entire network, while the training set for category links is constant (80%). Both 'separate' and shared similarity model are using low rank factorization of M due to the dimension size of the features, with a latent space of dimension 100. The size of latent space in two-layer network is set to 100 as well. All matrices are initialized by random values.

The 'separate' model shown in Table 1 uses only the link type that must be predicted and is identical to the classical distance learning model that has been used in previous works [4, 16, 5]. This model was previously compared with state-of-the-art link prediction methods and was shown to be effective for link prediction on networks. The separate model is ignorant about the other link type. The joint models make use of both link types according to the models described in this article. Table 1 clearly shows that the joint models improve the results significantly comparing to the separate model, SvmRank and Cosine similarity. The separate model is itself superior to the other models tested in [4, 16, 5]). SVMrank [11, 10] has shown high performance in ranking; to make it applicable to large graphs (here, the Amazon products) the optimization was performed on the primal form by using stochastic gradient descent. Not surprisingly, the two-layer Similarity model has better performance in comparison to the Shared Similarity model.

Table 2 shows similar results, but reversing the link types. In this experiment, we varied the proportion of category links made available for training, from 30% to 80% of the total number of objects, and kept the co-purchase links at a constant value (all links over 80% of the nodes used for training).

The important observation is that when there are few links available for training (columns at the left of the tables), the improvement of using both link types by the joint ranker is higher, percentage of the improvement is given in the parenthesis. However, even when the training set is large, the joint modeling is useful. Moreover, we observe in the extreme case when there is no training data available (marked "0 (transfer)" in the tables) and the ranker is only trained on the other link type, still the performance reached through *transferring* the learning from the other link type is significantly higher than the ad-hoc similarity metric in baselines (cosine-similarity). In the two-layer Similarity model

Model	Proportion of training set					
	0 (transfer)	**0.3**	**0.4**	**0.5**	**0.6**	**0.8**
Cosine Similarity	8.03	8.03	8.03	8.03	8.03	8.03
SVMrank	–	8.94	9.06	8.99	8.92	9.11
Separate Model	–	8.39	8.53	8.70	8.92	9.13
Shared Similarity	*8.43*	**9.16 (+9.1%)**	**9.21 (+7.9%)**	**9.31 (+7.0%)**	**9.56 (+7.1%)**	9.59 (+ 5.0%)
two-layer	*9.00*	**9.33 (+11.2%)**	**9.43 (+10.5%)**	**9.48 (+8.9%)**	**9.60 (+8.18%)**	**9.70 (+7.22%)**

Table 1: Average Mean Average Precision (MAP) for predicting *co-purchase links* between Amazon products, when the size of co-purchase training set varies from 30% to 80% of the entire network. The joint model uses both co-purchase and category links (training set for the latter is always 80% of the network). Bold results are significantly better (t-test, $p < 0.001$).

Model	Proportion of training set					
	0 (transfer)	**0.3**	**0.4**	**0.5**	**0.6**	**0.8**
Cosine Similarity	7.50	7.50	7.50	7.50	7.50	7.50
SVMrank	–	7.73	7.82	7.89	7.92	7.92
Separate Model	–	7.86	7.99	8.07	8.38	8.61
Shared Similarity	*7.97*	**8.31 (+5.7%)**	**8.42 (+5.3%)**	**8.53 (+5.7%)**	**8.73 (+4.1%)**	**8.92 (+ 3.6%)**
two-layer	*8.48*	**8.79 (+11.8%)**	**8.8(+10.51%)**	**8.8(+9.66%)**	**9.12(+8.8%)**	**9.25(+7.4%)**

Table 2: Average Mean Average Precision (MAP) for predicting *category links* between Amazon products, when the size of category training set varies from 30% to 80% of the entire network. The joint model uses both co-purchase and category links (training set for the former is always 80% of the network). Bold results are significantly better (t-test, $p < 0.001$).

we transfer the first layer and use an orthogonal matrix with all elements one for the second layer. The results confirm that there is correlation between categorical similarity of the products in Amazon and being co-purchased by customers, and each of them can help in modeling and predicting of the other one.

Table 3 similarly shows the average MAP of the TED dataset for the content links and Table 4 shows the average MAP for the co-liked links. The Joint models, specially two-layer model, significantly improve the link prediction performance for both link types. This shows that there is a big correlation between relatedness of talks in TED platform and being in the favorite list of many users.

The joint models have a higher performance than the separate models on both networks that we experimented with. The difference in the performance of the joint model and separate model is higher when there are fewer number of links from one link type. In that case, the denser link type can help more strongly the sparser link type through the shared information part. But even when there are equal links from different link types, joint modeling through shared information helps with the generality of the model and improves the link prediction performance.

4.4 Effect of Low-rank Factorization Dimension

We finally study the effect of low-rank factorization on the performance of ranking when the full training set is available. When the graph is larger, it is possible to train larger dimensions, but when the network is not large, larger dimensions make over-fitting more likely.

Table 5 shows the effect of dimension on link prediction performance for Amazon products. We observe that the growth of increase in the performance decreases when we increase the dimension. The time complexity of the algorithm linearly depends with the dimension. Therefore, for large scale graphs, one should choose a dimension by considering the trade-off between the time complexity and the performance.

Table 6 similarly shows the effect of dimension on the link prediction performance over the TED talks. Considering the size of network, increasing the dimension decreases the performance, as it over-fits the training set.

5. CONCLUSION

In this papar, we proposed two joint similarity learning models over nodes' attributes for link prediction in networks with multiple link types. The first model learns a similarity metric that consists of two parts: the general part, which is shared between all link types, and the specific part, which is trained specifically for each type of link. The second model consists of two layers: the first layer, which is shared between all link types, embeds the objects of the network into a new space, and then a similarity is learned specifically for each link type in this new space. Both models are applicable to large networks with high-dimensional feature spaces. The experiments show that the proposed joint modeling and training frameworks improve link prediction performance significantly for each link type in comparison to multiple baselines. This improvement is higher when there are fewer links available from one link type in the network. The two-layer similarity model outperforms the first one which is expected due to its capability of modeling negative correlations among different link types. Moreover, we illustrated that even if the models are trained completely on one link type and tested on the other, our models significantly improve the performance in comparison to ad-hoc

Model	Proportion of training set					
	0 (transfer)	**0.3**	**0.4**	**0.5**	**0.6**	**0.8**
Cosine Similarity	4.55	4.55	4.55	4.55	4.55	4.55
SVMrank	–	5.10	6.44	6.87	7.44	8.02
Separate Model	–	4.94	5.78	6.49	8.18	9.28
Shared Similarity	*6.30*	**5.96 (+20.6%)**	**6.98 (+20.7%)**	7.52 (+ 15.8%)	9.27 (+ 13.3%)	10.38 (+ 11.8%)
two-layer	*7.61*	**8.72(+76.5%)**	**8.95(+54.8%)**	**9.50(+46.3%)**	**9.91(+21.1%)**	11.13(+19.9%)

Table 3: Average Mean Average Precision (MAP) of *content links* in TED when the size of content train set is changed. The joint model uses both content and co-liked links, the size of co-liked links is 0.8. Bold results are significantly better (t-test, $p < 0.001$).

Model	Proportion of training set					
	0 (transfer)	**0.3**	**0.4**	**0.5**	**0.6**	**0.8**
Cosine Similarity	2.84	2.84	2.84	2.84	2.84	2.84
SVMrank	–	2.40	2.72	2.73	2.33	3.77
Separate Model	–	3.42	3.75	4.02	4.82	5.70
Shared Similarity	*4.77*	4.14 (+21.0%)	**4.54 (+21%)**	**4.83 (+20.1%)**	**6.10 (+26.5 %)**	6.44 (+ 12.9%)
two-layer	*5.50*	**6.11(+78.6%)**	**6.22(+65.8%)**	**6.49(+61.4%)**	**6.97(+44.6%)**	**7.60(+33.3%)**

Table 4: Average Mean Average Precision (MAP) of *co-liked* links in TED when the size of co-liked train set is changed. The joint model uses both content and co-liked links, the size of content links is 0.8. Bold results are significantly better (t-test, $p < 0.001$).

Model	Dimension			
	20	**50**	**100**	**150**
Separate Model (co-purchase)	6.60	8.52 (+29.9%)	9.13 (+7.16%)	9.31(+1.97%)
Shared Similarity (co-purchase)	7.29	9.24 (+26.75%)	9.59 (+3.79%)	9.91(+3.34%)
two-layer (co-purchase)	7.29	8.94(+22.63%)	9.70 (+8.50%)	10.11(+4.22%)
Separate Model (category)	7.59	8.41 (+10.08%)	8.61 (+2.38%)	8.67(+0.70%)
Shared Similarity (category)	7.96	8.76 (+10.05%)	8.92 (+1.83%)	8.98(+0.67%)
two-layer (category)	7.96	8.84(+11.05%)	9.25 (+4.63%)	9.40(+1.62%)

Table 5: Average Mean Average Precision (MAP) for predicting links between Amazon products.

Model	Dimension				
	10	**20**	**50**	**100**	**150**
Separate Model (content)	9.33	1**0.63**	9.52	9.47	9.28
Shared Similarity(content)	9.53	**11.39**	11.05	11.02	10.38
two-layer (content)	9.90	**11.72**	11.67	11.13	11.06
Separate Model (co-liked)	5.95	**6.48**	6.21	5.89	5.70
Shared Similarity (co-liked)	6.30	**6.96**	7.09	6.39	6.44
two-layer (co-liked)	7.63	**7.99**	7.97	7.60	7.67

Table 6: Average Mean Average Precision (MAP) for predicting links between TED videos.

similarity metrics.

APPENDIX

.1 Convergence of the Training Shared Similarity Model

To prove the convergence, first we need to establish an upper bound for the norm of the subgradients at each iteration $||\bigtriangledown_t(.)||_F^2 < U$ which can be easily found if the data points are bounded. For example, if all data points vectors are normalized, then they are bounded, which is the case in our study.

THEOREM 1. *Let* $\bar{G} = \frac{1}{T}\sum_{t=1}^{T} G^t$ *and* $\bar{S}_i = \frac{1}{T}\sum_{t=1}^{T} {S_i}^t$ *be the averages of* G *and* S_i *so far. Let the update rate be* $\eta_t = \frac{1}{\lambda t}$ *with* $\lambda \leq 1/4$. *Let also* $G^* = \arg\min_G L(G)$ *and* $S_i^* = \arg\min_{S_i} L(S_i)$. *If the data points are bounded, i.e.* $||x_a' x_b||_F^2 < R$ *with* $R \geq 1$, *then with a probability of at least* $1-\delta$, *we have (with* $r = 4R^2$*):*

$$L(\bar{G}) \leq L(G^*) + \frac{21r\ln(T/\delta)}{\lambda T}$$

$$L(\bar{S}_i) \leq L(S_i^*) + \frac{21r\ln(T/\delta)}{\lambda T}$$

PROOF. Each of the $L_t(G)$ and $L_t(S_i)$ is $\lambda-$strongly convex as they consist of a part with a shape $\frac{\lambda}{2}||M||_F^2$ and a

convex function (the average hinge function). According to the Theorem 1 in [15] the upper bound of the subgradients norms $|| \bigtriangledown L_t(S_i)||_F^2$, $|| \bigtriangledown L_t(G)||_F^2$ is $r = 4R^2$.

$L(S_i)$ and $L(G)$ can be considered as the loss function of a one-class PEGASOS algorithm for which the input data pints are $x'_a(x_c - x_b)$ for $(a, b, c) \in Train_i$. In Lemma 2 in [15] the inequalities stated in the theorem are proven for the PEGASOS loss function and are therefore valid for the $L(G)$ and $L(S_i)$. □

As a consequence of the above theorem we can write :

$$\bar{L} = L(\bar{G}) + \sum_i L(\bar{S}_i) \leq L(G^*) + L(S_i^*) + \frac{21(N+1)r\ln(T/\delta)}{\lambda T}$$

Therefore we can see that to obtain an error inferior or equal to ϵ with the confidence $1 - \delta$ we need $\tilde{O}(\frac{N+1}{\epsilon\delta\lambda})$ iterations.

A. REFERENCES

[1] L. A. Adamic and E. Adar. Friends and neighbors on the web. *SOCIAL NETWORKS*, 25:211–230, 2001.

[2] A. Agarwal, S. Chakrabarti, and S. Aggarwal. Learning to rank networked entities. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, pages 14–23, New York, NY, USA, 2006. ACM.

[3] L. Backstrom and J. Leskovec. Supervised random walks: Predicting and recommending links in social networks. In *Proc. Web Search and Data Mining (WSDM)*, 2011.

[4] B. Bai, J. Weston, D. Grangier, R. Collobert, K. Sadamasa, Y. Qi, O. Chapelle, and K. Weinberger. Learning to rank with (a lot of) word features. *Inf. Retr.*, 13(3):291–314, June 2010.

[5] B. Bai, J. Weston, D. Grangier, R. Collobert, K. Sadamasa, Y. Qi, C. Cortes, and M. Mohri. Polynomial semantic indexing.

[6] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, ICML '05, pages 89–96, New York, NY, USA, 2005. ACM.

[7] Y. Cao, J. Xu, T.-Y. Liu, H. Li, Y. Huang, and H.-W. Hon. Adapting ranking svm to document retrieval. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '06, pages 186–193, New York, NY, USA, 2006. ACM.

[8] J. Chang. Relational topic models for document networks. In *In Proceedings of the Conference on AI and Statistics (AISTATS*, 2009.

[9] K. Crammer and Y. Singer. Pranking with ranking. In *Advances in Neural Information Processing Systems 14*, pages 641–647. MIT Press, 2001.

[10] R. Herbrich, T. Graepel, and K. Obermayer. Large margin rank boundaries for ordinal regression. In A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 115–132, Cambridge, MA, 2000. MIT Press.

[11] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '02, pages 133–142, New York, NY, USA, 2002. ACM.

[12] J. Leskovec, L. A. Adamic, and B. A. Huberman. The dynamics of viral marketing. *ACM Trans. Web*, 1(1), May 2007.

[13] D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *Proc. ACM International Conference on Information and Knowledge Management (CKIM)*, 2003.

[14] K. Miller, T. Griffiths, and M. Jordan. Nonparametric latent feature models for link prediction. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1276–1284, 2009.

[15] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter. Pegasos: primal estimated sub-gradient solver for svm. *Math. Program.*, 127(1):3–30, 2011.

[16] B. Shaw, B. Huang, and T. Jebara. Learning a distance metric from a network. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 1899–1907. 2011.