Learning to Rank on Network Data

Majid Yazdani
Idiap Research Institute/EPFL
1920 Martigny, Switzerland
majid.yazdani@idiap.ch

Ronan Collobert
Idiap Research Institute
1920 Martigny, Switzerland
ronan.collobert@idiap.ch

Andrei Popescu-Belis
Idiap Research Institute
1920 Martigny, Switzerland
andrei.popescu-belis@idiap.ch

ABSTRACT

This paper proposes a method for learning to rank over network data. The ranking is performed with respect to a query object which can be part of the network or outside it. The ranking method makes use of the features of the nodes as well as the existing links between them. First, a neighbors-aware ranker is trained using a large margin pairwise loss function. Neighbors-aware ranker uses target neighbors scores in addition to objects' content and therefore, the scoring is consistent in every neighborhood. Then, collective inference is performed using an iterative ranking algorithm, which propagates the results of rankers over the network. By formulating link prediction as a ranking problem, the method is tested on several networks, with papers/citations and webpages/hyperlinks . The results show that the proposed algorithm, which uses both the attributes of the nodes and the structure of the links, outperforms several other methods: a content-only ranker, a link-only one, a random walk method, a relational topic model, and a method based on the weighted number of common neighbors. In addition, the propagation algorithm improves results even when the query object is not part of the network, and scales efficiently to large networks.

1. INTRODUCTION

The ranking problem over relational data, such as social networks, is defined as follows: given a query object, all the objects in the network must be ranked by decreasing relevance or relatedness with respect to this object, which may or may not be already part of the network. When the query object is already part of the network, some relations with other objects, represented by directed or undirected edges, are known. In this case, link-based approaches such as random walk models can provide an effective solution for the ranking problem. However, when the query object is not in the network, link-based approaches are not applicable, and supervised learning over node attributes must be considered for learning to rank.

To illustrate our proposal, let us consider a network made of scientific papers (objects) linked by citations between them (relations). Given a "query" paper, the goal is to rank all papers in the network according to a score that reflects the likelihood or appropriateness of being cited by this query paper. If the network is made only of completed papers, then this score should distinguish the papers actually cited by the query paper (high scores) from all those that are not cited (low scores). However, higher scores may also indicate papers that should have been cited but were not, or, conversely, citations that were spuriously inserted. The utility of the ranking task is more obvious for recommending citations to include in a new paper that is being written (an object outside the network if it makes no citations yet) or to extend an existing draft (an object already within the network if it makes some citations). In both cases, the goal is to compute a ranked list of recommendations for additional citations. Similarly, new connections in social networks can be recommended using the same model.

In this paper, we will show that relations between objects in a network can be used in both situations to increase the accuracy of ranking with respect to a given query. We will show how to unify the two above-mentioned approaches to ranking, and make use of the attributes of objects and of the relations between them at the same time. We propose to model the dependencies between the objects of a network, for the ranking problem, by using: (1) a neighbors-aware ranker based on the features of the objects along with neighborhood information; and (2) a collective inference algorithm which propagates the results of the neighbors-aware ranker in the network. We will show that the results obtained by using jointly the neighbors-aware ranker and the collective inference algorithm improve over a variety of state-of-the-art techniques: a content-only ranker, a link-only ranker, a random walk model, a relational topic model, and a common-neighborhood method.

For applications and evaluation, we will study link prediction, which is an instance of the ranking problem. Indeed, in this situation, given a query object, all other objects (i.e. nodes in the network) must be sorted according to the likelihood of creating a link between them and the query object. Link prediction is a binary ranking problem; the loss function used for training rewards a higher ranking of the objects that are actually linked in the network. This formulation has been used in several previous studies [15, 1, 3, 20, 2]. In the rest of this paper, we focus on binary ranking, although the proposed algorithms can be generalized to other ranking problems as well.

The paper is organized as follows. In Section 2 we define the ranking method on network data, first by introducing the neighbors-aware ranker and then by defining the collective inference algorithm. In Section 3 we evaluate the proposed method on several different data sets, and in Section 4 we compare our proposal to previous work.

2. LEARNING TO RANK ON A NETWORK

Homophily is the tendency for two nodes having similar "social characteristics" to have a higher probability of being linked together. Homophily is an important and widely accepted notion in the study of social networks, which applies to network relations of many types, such as marriage, friendship, work, advice, support, information transfer and co-membership [17]. The social characteristics of nodes are derived from various attributes of the nodes, for example age, sex, geographic location, college/university, work place, hobbies/interests. Homophily, though, might not hold in the space of the attributes: for example, marriage usually occurs between different genders (in most countries). We can conclude from the homophily principle that for a given query, the scores of the linked objects should be consistent, because it is likely that these objects have similar social characteristics.

In the classification of network data, homophily is exploited by using collective classification algorithms [13], in which a classifier is trained to classify each object by using at the same time the features of the object and the labels of other objects in its neighborhood. A collective inference algorithm propagates the results of the local classifiers in the network. For ranking on a network, we draw here inspiration from this approach to define a collective ranking method which consists of three parts:

1. The *content-only ranker* learns to rank only by using a similarity function on the attributes of nodes, while ignoring relations between them. This ranker embodies the approach which is traditionally used in learning to rank problems.
2. The *neighbors-aware ranker* learns to rank by using the information that is locally available at each node, coming from its attributes and its neighborhood (here, limited to first-degree neighbors).
3. The *collective inference algorithm* propagates the results of local rankers over the network, inspired by an existing collective inference method [16].

2.1 Neighbors-aware Ranker

The neighbors-aware ranker returns a score that is a function of a query, a target node, and the neighborhood scores of the target node. More formally, the neighbors-aware ranker is noted as $\psi(x_q, x_t, \bar{N}(q,t))$, in which x_q and x_t represent respectively the features of the query and target nodes, and $\bar{N}(q,t)$ represents the average score of the target's neighborhood. The domains of the ψ function are $\psi : \mathbb{R}^{2F+2} \rightarrow \mathbb{R}$, where F is the size of the feature space. For the sake of simplicity, and to follow common practice in learning to rank, we define ψ as a linear function of the similarity between query and target, and the neighbors score, although other learners can also be considered [6, 11] :

$$\psi(x_q, x_t, \mathcal{N}(q,t)) = \underbrace{similarity(x_q, x_t)}_{\text{content-only}} + \alpha \underbrace{\bar{N}(q,t)}_{\text{neighbors}} .$$

The first part, $similarity(x_q, x_t)$, is a content-only ranker which computes the similarity between q and t based only on their attributes, as proposed by most previous learning to rank methods, e.g. RankNet [7], polynomial semantic indexing [5] or structure preserving metric learning [20]. Here, we define the similarity function as: $similarity(x_q, x_t) = x_q \times M \times x'_t$. The M matrix is a $F \times F$ matrix, where F is the number of features representing the nodes, e.g. the number of content words for text-based nodes. In practice, to make training and storage possible, some constraints on M are considered, such as using a diagonal matrix or performing low-rank factorization [5, 20].

The second part, i.e. the neighborhood score $\bar{N}$, represents the contribution of the score of the neighbors of t with respect to q. This factor maintains consistency between scores in a neighborhood. For instance, if t's neighbors have high scores, t itself should have high scores as well. This takes into account the dependencies between objects, and allows us to perform collective ranking instead of local ranking only. As in our applications we use directed graphs, we consider two types of neighbors: neighbors from in-links ($\mathcal{N}_{in}(t)$) and neighbors from out-links ($\mathcal{N}_{out}(t)$). We define $\bar{N}$ recursively as follows (where $\psi(x_q, x_n, \bar{N}(q,n))$ is shortened as ψ):

$$\bar{N}(q,t) = \frac{w_1 \sum_{n \in \mathcal{N}_{in}(t)} \psi}{|\mathcal{N}_{in}(t)|} + \frac{w_2 \sum_{n \in \mathcal{N}_{out}(t)} \psi}{|\mathcal{N}_{out}(t)|}.$$

The parameters w_1 and w_2 represent the importance of the in-links neighbors and out-links neighbors, and are learned during the training along with the parameters of the *similarity* function.

According to the resulting formula for the ranker, the score of a target node t is defined based on the score of its neighbors, recursively. The next section explains how we deal with recursivity.

The α parameter (here $\alpha < 1$) is a dampening parameter which decreases the effect of the neighbors' scores on the target node score as their distance from the target node increases. This is a hyper-parameter of the algorithm and will not be learned during training.

2.2 Collective Inference

To compute the score for a given query and a target node, we need to know the scores of the query to the target's neighbors. To compute this recursive function, we start from an initial score and iteratively compute the scores at iteration τ based on the scores at iteration $\tau - 1$. The score of the target node t with respect to the query q at iteration τ, noted simply ψ^τ, is computed using the values of $\psi^{\tau-1}$ in the definition of $\bar{N}(q,t)$ above.

Considering the linear definition of ψ in the previous section, given a query it is possible to find out the final scores of the nodes by solving the inverse of the normalized adjacency matrix, and therefore perform the computation in one shot. There are several reasons for designing an iterative algorithm though: first, ψ can be a non linear function, as for instance similar to [6, 11]. Second, if the network is large, then time complexity makes the computation in one shot intractable. Moreover, we will show that propagating only high scores in each iteration, using our recursive algorithm, improves the

performance significantly.

The initial scores (noted ψ^0) are set as follows. If no prior link is known for the query node, the scores are initialized with the scores of the content-only ranker. If the query node is part of the network, the scores are based on the prior links and are set to 1 if (q, t) is an edge of the graph and to 0 otherwise.

To propagate scores, we propose a collective inference algorithm, which is effective, easy to implement, and scales well to large graphs. We refer to it as the 'Iterative Ranking Algorithm' (IRA) because at each iteration the results of the neighbor-aware rankers are propagated one step further in the graph. The pseudocode for the IRA is given as Algorithm 1 below. At the first step, the initial scores of all nodes are set as above. Then, scores are normalized, and the scores above the threshold propagate to the neighbors at the next step of the algorithm. This propagation of above-threshold scores continues until convergence, or until the maximum number of iterations is reached.

Algorithm 1 Iterative Ranking Algorithm.

IRA for query node q, for a vertex set V and edge set E
if $q \in V$ **then**
  $\forall i \in V, \psi_i^0 = \begin{cases} 1 & \text{if } (q,i) \in E \\ 0 & \text{otherwise} \end{cases}$
else
  $\forall i \in V, \psi_i^0 = \psi_{content}(q, i)$
end if
$\psi_{norm}^0(i) = norm(\psi_i^0)$ (normalize scores to [0, 1])
$\tau = 1$
while NotConverged and $\tau \leq T$ **do**
  for $i \in V$ **do**
    if $\psi_{norm}^{\tau-1}(i) < c$ **then**
      $\psi_i^{\tau-1} = 0$
      (scores below threshold do not propagate)
    end if
  end for
  $HighScoreNodes = \{i | \psi_i^{\tau-1} > 0\}$
  $PossiblyChanged = \{i | i \in \mathcal{N}_{in}(HighScoreNodes) \vee i \in \mathcal{N}_{out}(HighScoreNodes)\}$
  for $i \in PossiblyChanged$ **do**
    Update ψ_i^τ using $\psi^{\tau-1}$
  end for
  $\psi_{norm}^\tau(i) = norm(\psi_i^\tau)$ (normalize scores to [0, 1])
  $\tau = \tau + 1$
end while

Given the linear definition of ψ, the algorithm converges if $|w_1| < 1$ and $|w_2| < 1$, because the effect of long paths decreases exponentially and eventually vanishes. The sketch of the convergence proof is given in the Appendix. To ensure convergence, during the training of the ranker, we constrain the parameters (i.e. the coefficients of the similarity matrix, w_1, and w_2) to be between -1 and 1. Moreover, this constraint acts as a regularizer on the parameters in the minimization of the loss function, avoiding exploding the values of the parameters (see Section 2.3).

The normalization of the scores and their thresholding are used as follows, in order to preserve the linear algebraic nature of the propagation and thus guarantee its convergence. If the normalized score is higher than the threshold c, the *unnormalized* score is passed to the next step, otherwise, the score is set to zero and does not propagate to the next step. Therefore, only some terms are expanded while others are set to zero, which ensures convergence.

The thresholding prevents the propagation of noise in the graph, therefore improving accuracy, as shown in Section 3.2. Moreover, it increases the speed of the algorithm for large graphs, because at each iteration only the scores that may have changed are updated. In fact, only the scores of the neighbors of nodes with scores higher than the threshold can be changed in the next iteration. Many real-world networks are small world networks, in which each node is only connected to a small number of other nodes in its community. In small world networks, at each iteration only few nodes have high scores and the rest have low scores. Therefore, choosing a reasonably high threshold leaves us with only few nodes at each iteration. However, in theory, the thresholding does not change the worst-case time complexity of the algorithm.

2.3 Training the Neighbors-aware Ranker

The goal of training is to approximate a ranker which minimizes a ranking loss function over the graph G for the queries, target nodes, and associated grades that are given in the training set. In this section we discuss the training of a binary ranker for the link prediction problem (the method can be generalized to other ranking problems).

At training time, the graph $G = (V, E)$ is available, where V is the vertex set and E the edge set, as in Algorithm 1. We assume that there are two possible grades, 'connected' or 'not connected', which are given by the existing edges in the graph. We define the training set $\mathcal{T}$ made of triples of nodes, so that the first two are connected, but not the first and the third one: $\mathcal{T} = \{(i, j, z) | (i, j) \in E \text{ and } (i, z) \notin E\}$. The goal of training is to minimize the empirical risk using a pairwise hinge loss function (as in [14]) over the training set $\mathcal{T}$ as follows:

$$Min\ L = \sum_{(i,j,z)\in\mathcal{T}} max(0, d_{marg} - \psi(i, j, \bar{N}) + \psi(i, z, \bar{N}))$$
$$\text{so that } 0 \leq w_i \leq 1 \text{ and } 0 \leq M_{ij} \leq 1$$

The parameters of the neighborhood component of ψ are w_1 and w_2 ($-1 \leq w_1 \leq 1$ and $-1 \leq w_2 \leq 1$), and the parameters of the similarity component are the M_{ij} coefficients ($-1 \leq M_{ij} \leq 1$). As mentioned, the regularization prevents the arbitrary growth of the parameters in the optimization of L, and is equivalent to the optimization of the same loss function plus the infinity norm of the parameters. We define d_{marg} as a constant margin that should separate the examples receiving different grades.

If the training graph G is large, 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 randomly select i, j and z from $\mathcal{T}$ and perform gradient descent on them. To impose the regularization constraint we use gradient projection.

The dimension of M is F^2 where F is the number of fea-

tures of the data points. In practice, to make training and storage possible, particularly when we are dealing with high dimension data such as text, we perform low-rank factorization of M, assuming that $M = AB'$, where A and B are matrices from F to a lower dimension (M is not necessarily symmetric, hence the two lower-rank matrices). The objective function with the low-rank matrices is not convex any more, but in practice it has been shown that low-rank factorization performs well [5].

At each iteration, the time complexity of the stochastic training algorithm is $O(Z^2)$, where Z is the average number of non-zero features of the objects, which can be much smaller than the dimension F of the feature space. For example, for textual data, the feature vectors are very sparse in comparison to the dimension of the feature vector, which is the number of possible words (the vocabulary size).

3. EXPERIMENTAL SETUP & RESULTS

In this section, we apply the proposed method to link prediction on several networks. Given a query node, all other objects are ranked according to their score, with the goal of having the objects that are actually linked to the query node at the top of the list. To build a test set from an initial network, we randomly remove about 10% of the nodes, keeping them for testing, and train the algorithm on the remaining network. To evaluate the algorithm, for each node in the test set, the precision and recall of the first k ranked objects are calculated with respect to actual links.

We use several data sets which have been frequently used for the collective classification, and more details about them can be found in [19]. The networks are by nature directed graphs. Each object (paper or web page) is characterized by a set of binary features or attributes, which are the presence or absence of a given word from the global content-word vocabulary.

1. The *Cora* dataset consists of papers in the field of machine learning. The papers were selected so that in the final corpus every paper cites or is cited by at least one other paper. It contains 5429 edges and 1440 nodes.

2. The *CiteSeer* dataset contains a selection of papers in the field of computer science. Again, the papers were selected with the same constraint as for Cora. It contains 4715 edges and 3709 nodes.

3. The *WebKB* dataset contains web pages and hyperlinks between them, gathered from four different universities. It contains 1608 edges and 1709 nodes.

3.1 Performance of Collective Ranking

We present first the performance of the proposed method in terms of *recall at 10* for the three data sets. The *precision at 10* acts similarly. The evaluation scenario, in which query objects do not have prior links to the network, corresponds for instance to situations in which a new person joins a social network, or a paper is being written and does not cite any other paper yet. Therefore, the ranking with respect to a test query node is performed by using only two types of information sources only: first, the features of the query node and of the objects in the network; and second, the relations between the objects in the network. The second type can be exploited only by using the proposed neighboorhood-aware ranker and propagation algorithm. This task is thus intractable for algorithms based on only the link structure, such as random walk algorithms or algorithms based on common neighborhood.

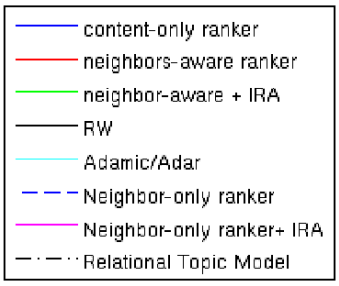

Figure 1: Color coding of the compared ranking methods (RW stands for Personalized PageRank random walk).

However, another frequent real-world situation is when some prior relations of the query object are known, e.g. a paper with some known citations or a person which already has some relations in a social network. It is possible to explore jointly this context and the above one by studying the performance of the proposed algorithms when the *proportion of the known prior links of the query objects in the test set* varies from zero to a non-zero value below 100%, so that some unknown links are left for testing. In what follows, we vary the proportion of known prior links from 0 to 0.7, thus exploring both scenarios above.

The threshold of the IRA algorithm is set to $c = 0.8$, which makes the computation fast on the studied networks. The effect of this threshold is discussed in the following section through additional experiments. The similarity matrix M is constrained to be diagonal. The reported results are the *average of 10 different runs* for which the test set was chosen randomly at the start of each run. We consider average recall at 10 for eight conditions (see color-coding in Figure 1). One is the full proposal of this paper (neighbors-aware ranker with IRA), another one is its first part only (without propagation), plus a content-only ranker, a neighbors-only ranker with IRA, and the same ranker without IRA. The remaining three are state-of-the-art ones, shown to reach high scores in previous studies: Personalized Page Rank random walk [12], Adamic and Adar similarity [15], and a Relational Topic Model.

The values of recall in Figure 2 show that for all three data sets, using the neighbors-aware ranker with the IRA propagation algorithm leads to ranking scores that are always higher than all the other algorithms, including the neighbors-aware ranker alone. Therefore, the method proposed in this paper appears to be effective and competitive.

When the number of known relations of the query objects is increased (moving towards the right side of the graphs), the performances of the neighbors-only ranker and of the link-based methods get closer to the rest of the rankers (such

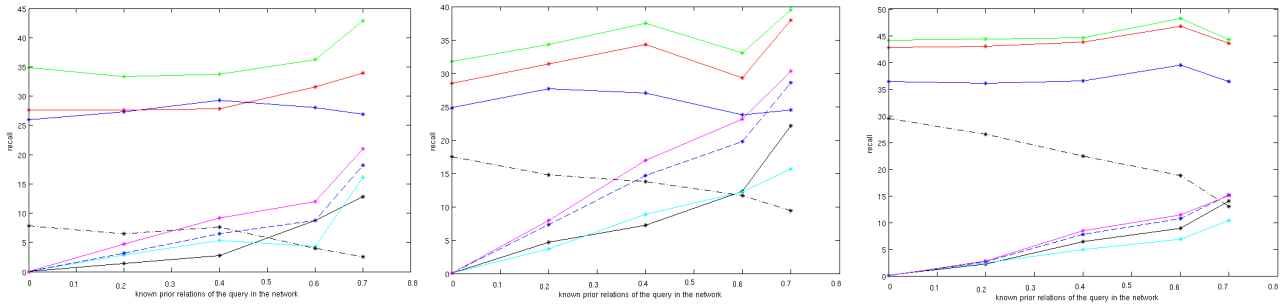

Figure 2: Recall at 10, as a percentage, for the eight rankers listed in the text and shown in Figure 1. The proportion of known prior relations of the query objects in the network is increased from 0 to 0.7. From left to right, the data sets are WebKB, Cora, and CiteSeer.

as the random walk one), because the neighborhood and link structure information around the query nodes become available. But when there are few known relations of the query objects in the network (left side of the graphs), the performance of the neighbors-aware ranker plus IRA, the neighbors-aware ranker alone, and even the content-only ranker are much higher than link-based approaches such as the random walk model.

Propagation by IRA is shown to be useful even when queries have no known relations at all in the network, in the leftmost data points in Figure 2. In this case, the IRA propagates the result of the content-only ranker in the network by exploiting the relations between objects in the network. For instance, this can increase the score of a node that does not get a high score from feature-based similarity, but is connected to many nodes with high feature-based scores. For instance, in a paper/citation network, a low-score paper according to its content might see its score increased after propagation because it is cited by many high-score papers. Therefore, modeling the relations between objects in the network even when there are no known links from the query object is shown to be helpful.

The neighbors-aware ranker, which uses neighborhood information that is only one transition away, improves the results significantly for Cora and CiteSeer networks, but does not always improve the results on the WebKB network. On the other hand, the propagation algorithm, IRA, improves the performance more on WebKB and Cora network and is less effective on the CiteSeer network. The combination of both the neighbors-aware ranker and the IRA has the capability to model varying-length dependencies if needed, and results in a robust ranking method which is effective on different networks.

Precision and recall score are not necessarily increasing with the increase of the known links of the query objects (going from the left to the right side of the each graph). This is because for each query object in the dataset only a fixed number of connected objects (e.g. cited papers) is available, and by revealing more of them the number of potential correct answers decreases, i.e. the upper bound of precision at k (here $k = 10$).

3.2 Effect of Threshold

In this section, we analyze experimentally the effect of the threshold c below which the normalized scores are not propagated in the IRA algorithm. We report *recall at 10* for the three data sets while varying the threshold c, for three different proportions of known links (0.2, 0.4 and 0.8), in Figures 3. The first observation is that the performance increases when increasing the threshold. The main reason is that with a lower threshold, many wrong predictions are propagated in the graph. Still, when the threshold is close to 1, the performance drops again as the propagation (which was shown above to be clearly useful) decreases and eventually stops.

The results demonstrate the advantage of the proposed method, because the performance increases when using a high threshold, and moreover in this case only a few nodes are updated at each iteration, thus accelerating the IRA. The running time, in practice, is nearly constant with respect to the graph size, although the theoretical worst case time complexity remains linear. We report in Figure 4 the required inference time on the data sets while varying the threshold c. Increasing the threshold makes the required time decrease rapidly (for low thresholds), as there are many nodes with very low scores (long tail). Then, by increasing further the threshold, the required time decreases only slightly. This confirms our assumption that these networks are small world graphs and form connected communities, which makes our propagation algorithm very efficient.

We performed a similar experiment varying α when the threshold c is fixed. The results show that for very small values of α the neighborhood does not have any effect on the score and the performance unsurprisingly decreases. If α is close to 1, then although the w_1 and w_2 are trained on the network, the training becomes more difficult as the loss function gives a high weight to the neighborhood part in comparison to the similarity part, so again performance decreases. In addition, when *alpha* is close to 1, the convergence is harder and loops can have negative effects on the performance.

4. RELATED WORK

Related work falls into several categories: ranking based on link structure, learning to rank on objects' attributes, and

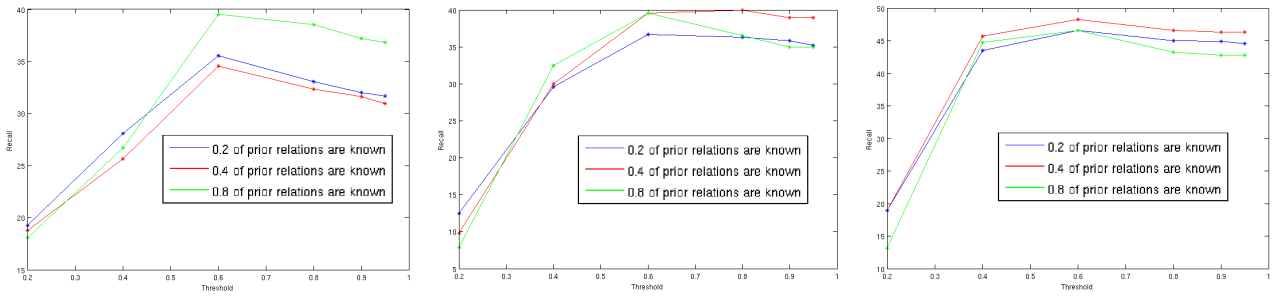

Figure 3: Recall at 10 of the neighbors-aware ranker with IRA, as a percentage, when increasing the threshold c of the IRA from 0.20 to 0.95 on the three data sets. In each graph, the curves correspond to 0.2, 0.4 and 0.8 of prior relations known.

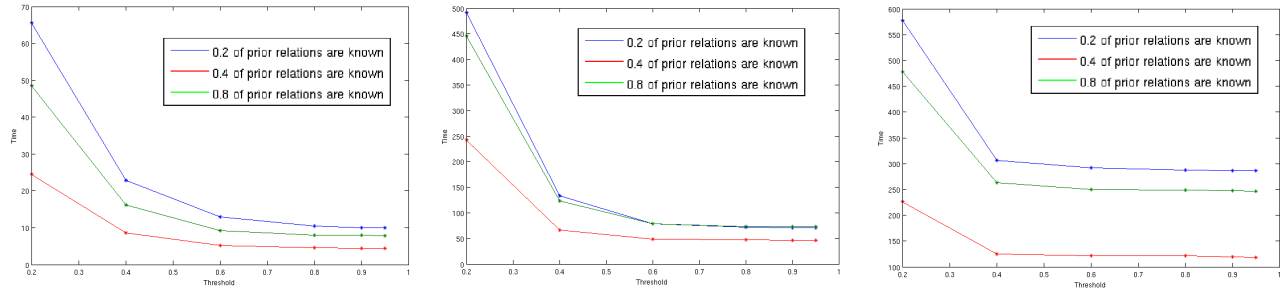

Figure 4: Inference time of the IRA, with the neighbors-aware ranker, when increasing the threshold c from 0.20 to 0.95 on the three data sets. In each graph, the curves correspond to 0.2, 0.4 and 0.8 of prior relations known.

learning to predict links.

Ranking based on link structure. The link prediction task can be formulated as a ranking task of pairs of nodes, based on similarity metrics that use link structure, for example random walk metrics or metrics using the common neighborhood. The Adamic and Adar [1] similarity measure, which is based on common neighborhood, yields relatively high performance in link prediction [15]. Methods based on random walks, such as Personalized Page Rank [12] or Visiting Probability [21] also showed good performance on relatedness, recommendation, and link prediction tasks. However, these approaches are not trained on the attributes of nodes (and those of edges, when available). Therefore, if a query object is weakly (or not) included in the link structure, these methods can not perform well (e.g., in the case of paper citation, if a paper contains only few or no citations). Our proposal, instead, can take advantage of the attributes of nodes when learning the similarity function.

Learning to rank on objects' attributes. Many methods for learning to rank have been proposed, with various learning abilities. Perception Ranking [10] is an online algorithm for ordinal classification, which can be employed for ranking as a pointwise method. The algorithm learns several parallel perceptron models which perform classification between the neighboring grades.

IR SVM [8] is a pairwise method which formulates the ranking problem as an SVM classification, for document retrieval. 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 and seems especially problematic for link prediction, due to the numerous non-linked examples. Similarly, SvmRank [14] is a pairwise ranking algorithm which transforms the pairwise ranking to SVM binary classification. However, batch optimization on large graphs is not possible with SvmRank due to the large number of non-linked pairs.

RankNet [7] is a feed forward neural network which is trained by back propagation to learn the scores of each training example. Authors in [4] perform a supervised training of a nonlinear model between vectors of words to preserve a pairwise ranking between documents, which scales well to large data sets with many words. Similarly, authors in [20] trained a distance metric by stochastic gradient descent over a hinge loss function that preserves the network structure.

In comparison to the approaches above, which assume that data points are independent, we consider dependency between the objects in the network by using neighborhood information together with features of objects. Moreover, we use collective inference to propagate the results of each ranker. In other words, we fill the gap between the link-based ranking and learning to rank on features, and we show that our algorithm is more effective than either approach.

Learning to predict links. Authors in [3] describe the problem of link prediction as a supervised learning task and use supervised random walks to solve it. Although it uses both link structure and object features, this method requires the iterative computation of the gradient, which makes training inefficient. Similarly, another supervised method for learning to rank [2] uses a random walk model and learns the transition probabilities from the ordered pairs of objects in the training data. As a transition probability is learned for each link, overfitting is likely and training is not effective for large-scale graphs due to number of parameters. Moreover, the two previous methods do not apply when the query object is not part of the network.

Authors in [18] adopt a generative Bayesian nonparametric approach to simultaneously infer the number of latent features and to learn which entities have each feature. This method is difficult to train, and inference for large scale graphs is time-consuming. Relational Topic Models (RTM) [9] consider both the documents and the links between them. For each pair of documents, their link is a binary random variable that is conditioned on their content. The model can be used to predict links between documents and is used as a comparison point below (Section 3). Our proposal outperforms RTM on the studied networks, presumably because the neighborhood information is not modeled explicitly in RTMs.

5. CONCLUSION AND FUTURE WORK

We proposed a learning to rank algorithm on network data, which uses both the attributes of the nodes and the structure of the links, for learning and inference. The proposed collective inference algorithm, IRA, propagates the predicted scores through the graph on condition that they are above a given threshold. Thresholding improves performance and makes a time-efficient implementation possible.

The proposed algorithm improved the link prediction performance on three different networks, for binary graded scores (connected vs. not connected). The results showed more specifically that the neighbors-aware ranker, which uses content features and scores of the neighbors, has a higher performance than the content-only ranker, the neighbors-only ranker, a Relational Topic Model and two linked-based similarity measures. Moreover, using the IRA in addition to the neighbors-aware ranker improves the performance even more.

In the current model, the influence of the neighbors is not considered: all neighbors are assumed to be equal. However, neighbors might have different influences, including negative ones. Our model can be generalized to learn a node-specific influence of neighbors as a function of their features and of the features of the edges between them. This model has more parameters and needs therefore bigger training sets to perform reliable experiments. Moreover, a recursive procedure can be considered for training the ranker, which does not assume that the training network is complete. The ranker could be trained first by using the scores from the current links, and then by replacing them with those provided by the ranker learned in the previous iteration. These perspectives deserve further experimental investigations.

APPENDIX

The convergence of the IRA algorithm can be proven as follows. We first make explicit the value of ψ_i^τ, the score of the node i after τ iterations, and we omit q from the equations for simplicity.

$$\psi_i^\tau = Sim_i + \alpha \frac{w_1}{|\mathcal{N}_{in}(t)|} \sum_{n_1 \in \mathcal{N}_{in}(t)} (Sim_{n_1} +$$

$$+\alpha \frac{w_1}{|\mathcal{N}_{in}(n_1)|} \sum_{n_2 \in \mathcal{N}_{in}(n_1)} \psi_{n_2}^{\tau-2} + \alpha \frac{w_2}{|\mathcal{N}_{out}(n_1)|} \sum_{n_2 \in \mathcal{N}_{out}(n_1)} \psi_{n_2}^{\tau-2})$$

$$+\alpha \frac{w_2}{|\mathcal{N}_{out}(t)|} \sum_{n_1 \in \mathcal{N}_{out}(t)} (Sim_{n_1} +$$

$$+\alpha \frac{w_1}{|\mathcal{N}_{in}(n_1)|} \sum_{n_2 \in \mathcal{N}_{in}(n_1)} \psi_{n_2}^{\tau-2} + \alpha \frac{w_2}{|\mathcal{N}_{out}(n_1)|} \sum_{n_2 \in \mathcal{N}_{out}(n_1)} \psi_{n_2}^{\tau-2})$$

If we continue expanding the above equation, the coefficient for the similarity of a node that is n transitions away is $\alpha^n \times w_{a_1} \times \ldots w_{a_n}$ with $a_i \in \{1, 2\}$. When increasing n, this coefficient tends to zero, given that $\alpha < 1$, $w_1 < 1$, and $w_2 < 1$, which means that the effect of long paths (including loops) vanishes, and the algorithm converges.

Acknowledgments

This work has been supported by the Swiss National Science Foundation through the National Center of Competence in Research (NCCR) on Interactive Multimodal Information Management (IM2), http://www.im2.ch.

6. 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. *Information Retrieval*, 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. In *Advances in Neural Information Processing Systems*, volume 22, pages 64–72, 2009.

[6] A. Bordes, X. Glorot, J. Weston, and Y. Bengio. Joint learning of words and meaning representations for open-text semantic parsing. *Journal of Machine Learning Research - Proceedings Track*, pages 127–135, 2012.

[7] 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.

[8] 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.

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

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

[11] X. Glorot, A. Bordes, J. Weston, and Y. Bengio. A semantic matching energy function for learning with multi-relational data. *CoRR*, abs/1301.3485, 2013.

[12] T. H. Haveliwala. Topic-sensitive PageRank: A context-sensitive ranking algorithm for web search. *IEEE Transactions on Knowledge and Data Engineering*, 15:784–796, 2003.

[13] D. Jensen, J. Neville, and B. Gallagher. Why collective inference improves relational classification. In *Proceedings of the 10th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '04, pages 593–598, New York, NY, USA, 2004. ACM.

[14] 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.

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

[16] L. McDowell, K. Gupta, and D. Aha. Cautious collective classification. *Journal of Machine Learning Research (JMLR)*, 2009.

[17] M. McPherson, L. Smith-Lovin, and J. M. Cook. Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 27:415–444, 2001.

[18] 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.

[19] P. Sen, G. M. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–106, 2008.

[20] 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.

[21] M. Yazdani and A. Popescu-Belis. Computing text semantic relatedness using the contents and links of a hypertext encyclopedia. *Artificial Intelligence*, 194:176–202, 2013.