ALPINE: Active Link Prediction using Network Embedding

Xi Chen , **Bo Kang** , **Jefrey Lijffijt** and **Tijl De Bie**
Dept. of Electronics and Information Systems, IDLab, Ghent University
{xi.chen, bo.kang, jefrey.lijffijt, tijl.debie}@ugent.be

Abstract

Many real-world problems can be formalized as predicting links in a partially observed network. Examples include Facebook friendship suggestions, consumer-product recommendations, and the identification of hidden interactions between actors in a crime network. Several link prediction algorithms, notably those recently introduced using network embedding, are capable of doing this by just relying on the observed part of the network.

Often, the link status of a node pair can be queried, which can be used as additional information by the link prediction algorithm. Unfortunately, such queries can be expensive or time-consuming, mandating the careful consideration of which node pairs to query. In this paper we estimate the improvement in link prediction accuracy after querying any particular node pair, to use in an active learning setup.

Specifically, we propose ALPINE (Active Link Prediction usIng Network Embedding), the first method to achieve this for link prediction based on network embedding. To this end, we generalized the notion of V-optimality from experimental design to this setting, as well as more basic active learning heuristics originally developed in standard classification settings. Empirical results on real data show that ALPINE is scalable, and boosts link prediction accuracy with far fewer queries.

1 Introduction

Applications of *Link Prediction* (LP) in networks range from predicting social network friendships, consumer-product recommendations, citations in citation networks, to protein-protein interactions. Such networks are usually only partially observed: node pairs are either connected (or *linked*), disconnected (or *unlinked*), or of *unknown* status. Indeed, obtaining network connections is usually resource-intensive (e.g., wet lab experiments or questionnaires), so that many of them remain unknown. Moreover, in many real-world networks new nodes are continuously added, with very limited information on their connectivity to the rest of the network. In such *Partially Observed Networks* (PONs), LP algorithms can be deployed to predict the missing link status information. When no attribute or meta-data is available for the nodes—the situation we focus on in this paper—this must be done relying solely on structural information [Kashima *et al.*, 2009], i.e., the observed part of the PON.

In some cases, a budget is available for querying an oracle for the link status of a limited number of node pairs. For example, wet lab experiments can reveal missing protein-protein interactions, or questionnaires can ask consumers to indicate whether they have seen particular movies before. Unfortunately, such queries can be expensive, while the link status of some node pairs is more informative than those of others—queries must thus be chosen wisely. Given a finite budget, an active learning strategy, identifying and prioritizing the most informative queries, is thus required for optimal LP accuracy of the unobserved part in the PON.

For LP tasks, NE methods have become increasingly popular, owing to their high accuracy as well as versatility for other downstream tasks. Thus, in this paper we develop ALPINE (Active Link Prediction usIng Network Embedding), the first active learning method for NE-based LP in PONs. For concreteness, we derived ALPINE for Conditional Network Embedding [Kang *et al.*, 2019] (CNE), which achieves the state-of-the-art LP accuracy [Kang *et al.*, 2019]. Moreover, as opposed to other popular NE methods (including all those based on random walks), CNE can distinguish *disconnected* node pairs from those with *unknown* status. Additionally, its objective function is easy to express analytically, which allows principled mathematical derivations for our active learning query strategies. Yet, it will be clear that ALPINE can be derived also for a wide range of other NE-based LP methods.

Given a PON, ALPINE must thus quantify the usefulness of querying a node pair with unknown status. We introduce different strategies for doing this. First, generalizing the notion of V-optimality from experimental design and exploiting the notion of Fisher information, we derive a principled measure that quantifies the reduction in variance on the link probabilities for node pairs with unknown status. Second, we propose several other heuristic strategies similar to those used in the standard active learning literature.

Example. We illustrate the idea behind ALPINE (with V-optimality query strategy) on the Harry Potter network [Evans *et al.*, 2014] of 65 Harry Potter characters and 221 ally connections amongst them (see Fig. 1). We take node 'Harry Potter' (pentagon node) and assume its connectivity is not

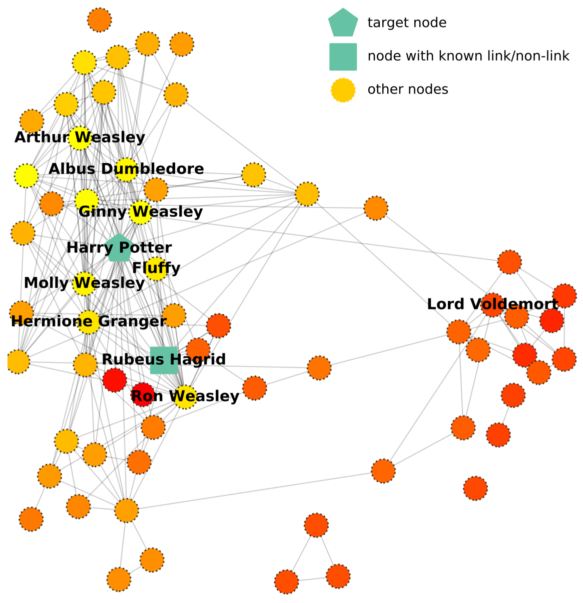

Figure 1: Visualization of the Harry Potter network .

observed except the connection to 'Rubeus Hagrid' (square node). All other connections are assumed observed. ALPINE then scores the informativeness of the link status of 'Harry Potter' to the other characters. The round nodes are colored according to the scores: the yellower, the higher. ALPINE selects the Weasley family (nodes with brightest yellow) as the top ones. Since the Weasley family is well connected with 'Rubeus Hagrid' and it is also connected to many other characters, it makes intuitive sense that the link status between them and 'Harry Potter' is indeed very informative for predicting the other unknown connections for 'Harry Potter'.

To the best of our knowledge, NE-based active LP has not been studied before. Indeed, while ample previous work on active learning for graphs exists, this mainly focuses on node classification [Yang *et al.*, 2014; Yang *et al.*, 2016; Bilgic *et al.*, 2010]. The **main contributions** of this paper are thus:

- We highlight the importance of distinguishing the *disconnected* versus *unknown* status of node pairs, a rather obvious but often ignored fact (Sec. 3.1).
- We propose ALPINE, an active learning approach for LP algorithms that are based on NE (Sec. 3.2).
- To identify the most informative node pairs to query, we generalize the notion of V-optimality to this setting. Moreover, we also propose a number of simpler heuristic query strategies inspired by active learning for standard learning settings. (Sec. 3.3.)
- Qualitative and quantitative evaluations show that ALPINE with the V-optimality query strategy does indeed perform far better than random querying. Moreover, we show that two easy-to-compute heuristics achieve very similar performance, making them good alternatives on large networks. (Sec. 4).

2 Background

Here we briefly survey active learning, (NE-based) link prediction, and some directly related work.

2.1 Active Learning and Experimental Design

Active learning is a subfield of machine learning, which aims to exploit the situation where learning algorithms are allowed to actively choose the training data from which they learn. It is particularly valuable in domains where training labels are scarce and expensive to acquire [Brinker, 2003; Cai *et al.*, 2017; Settles, 2009]. The success of an active learning strategy depends on how much more effective its choice of training data is, when compared to randomly sampled training data. Of particular interest to the current paper is the pool-based active learning, where a pool of unlabeled data points is provided, and a subset from this pool can be selected for labeling. In the context of the present paper, the unlabeled 'data points' are the node pairs with unknown link status, and an active learning strategy would aim to query the link status of those node pairs from this pool that are most informative for a LP algorithm when used to predict the link status of the node pairs for which it is unknown.

Active learning is closely related to *Optimal Experimental Design* (OED) in statistics [Atkinson *et al.*, 2007], which aims to design optimal 'experiments' (i.e., the acquisition of training labels) with respect to a statistical criterion and within a certain cost budget. The objective of OED is usually to minimize a quantity related to the (co)variance matrix of the estimated model parameters, or of the predictions this model makes on the test data points. In models estimated by the maximum likelihood principle, a crucial quantity in OED is the *Fisher Information*: as the reciprocal of the estimator variance, it allows quantifying the amount of information a training data point carries about the parameter.

While studied since long in statistics, the idea of variance minimization first shows up in the machine learning literature for regression [Cohn *et al.*, 1996], and later the Fisher Information was used to judge the asymptotic values of unlabeled data for classification [Zhang and Oles, 2000]. Yet, despite this related work in active learning, and the rich and mature statistical literature on OED for classification and regression problems, to the best of our knowledge the concept of variance reduction has not yet been applied to LP in networks.

2.2 Link Prediction and Network Embedding

LP algorithms can be used in PONs to predict whether node pairs with unknown status are linked or not. It has been widely applied in friendship recommendations, recommender systems, knowledge graph completion, and more. While there are numerous conventional LP methods based on heuristic statistics [Martínez *et al.*, 2017], recently proposed NE-based methods have been reported to outperform those [Grover and Leskovec, 2016; Kang *et al.*, 2019].

Given a network $\mathcal{G} = (V, E)$ with nodes V and links $E \subseteq \binom{V}{2}$, the goal of NE is to find a mapping $f : V \rightarrow \mathbb{R}^d$ from nodes to d-dimensional real vectors. A NE is denoted as $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n)' \in \mathbb{R}^{n \times d}$, where $\boldsymbol{X}^*$ denotes an *optimal* embedding given the network G with adjacency matrix

$\boldsymbol{A}$. NE's can be used for a variety of downstream tasks, including visualization, node classification, and also LP. When used for LP, a function $g : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ evaluated on the vectors $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ might represent the probability (or other score) for a link to exist between i and j. In practice, g can be found by training a classifier (e.g., logistic regression) on a set of linked and unlinked node pairs, while in other cases it follows directly from the embedding model. The method CNE used in this work is of the latter type, and aims to find an embedding that maximizes the probability of the network given the embedding:

$$P(\mathcal{G}|\boldsymbol{X}) = \prod_{\{i,j\}\in E} P(a_{ij} = 1|\boldsymbol{X}) \cdot \prod_{\{k,l\}\notin E} P(a_{kl} = 0|\boldsymbol{X}),$$

where $P(a_{ij} = 1|\boldsymbol{X}) = g(\boldsymbol{x}_i, \boldsymbol{x}_j)$ for a suitably defined g.

Most NE methods treat node pairs with unknown status as unconnected. For example, in methods based on skip gram with negative sampling (e.g, DeepWalk, node2vec), the random walks used to determine similarities between nodes use only known existing links, ignoring the unknown but potentially existing ones. Those methods may therefore be suboptimal when applied to PONs, and they cannot exploit new knowledge that a node pair with unknown status is not connected. As we will show in Sec. 3, however, CNE can trivially be modified to distinguish these two situations—an important factor contributing to our decision to use CNE in this paper.

2.3 Related work

Our work sits at the intersection of three topics: active learning, link prediction, and network embedding. There exists work on the *pairs* of any two but not three. To the best of our knowledge, ALPINE is the first method for actively learning a NE for the purpose of LP.

NE-based LP. Many graph embedding methods have been proposed in the past years. Based on neighborhood information, first order methods (e.g., CNE [Kang *et al.*, 2019]) and higher order methods (e.g., Deepwalk [Perozzi *et al.*, 2014], node2vec [Grover and Leskovec, 2016]) have been designed to perform multiple tasks, such as LP and node classification. Recently also Graph Convolutional Neural Networks (GCNNs) [Kipf and Welling, 2017] were introduced, allowing nodes to recursively aggregate information from their neighbors. However, GCNNs mainly focus on node classification.

Active learning for NE. There are a few works on learning NE in an active manner recently, but they target node classification [Cai *et al.*, 2017; Chen *et al.*, 2019] instead of LP.

Active learning and link prediction. Work on active learning for graphs has focused on node and graph classification, as well as LP [Settles, 2009; Aggarwal *et al.*, 2014]. The graph classification task considers data samples as graph objects, useful for drug discovery and subgraph mining [Kong *et al.*, 2011], while the node classification task aims to label nodes in graphs [Bilgic *et al.*, 2010; Cesa-Bianchi *et al.*, 2013; Guillory and Bilmes, 2009]. Other active learning research for LP considers different problem settings, e.g., link classification for signed networks [Cesa-Bianchi *et al.*, 2012], learning for graph edge flows [Jia *et al.*, 2019], and training of the neural link predictor [Ostapuk *et al.*, 2019]. Probably the most strongly related method is HALLP [Chen *et al.*, 2014], which uses active learning for LP. However, the method is heuristic, it does not distinguish disconnected from unknown node pairs, and it is based on a simple LP method that classifies node pairs according to a fixed representation of them in terms of engineered features, rather than a learned NE.

3 Method

Section 3.1 formally defines PONs, and discusses how CNE is naturally suitable for embedding PONs. Section 3.2 describes ALPINE, our NE-based active LP framework. Section 3.3 shows how we generalize the notion of V-optimality from experimental design for ALPINE, as well as more heuristic active learning query strategies.

3.1 Link Prediction for PONs

We formally define PONs as follows:

Definition 1. *A Partially Observed Network (PON) is an undirected network $\mathcal{G} = (V, E, U)$ where V is a set of $n = |V|$ nodes, $E \subseteq \binom{V}{2}$ and $U \subseteq \binom{V}{2}$ the sets of node pairs with connected and unknown status respectively, where $E \cap U = \emptyset$. $D \triangleq \binom{V}{2} \setminus (E \cup U)$ is the set of node pairs observed to be disconnected. Therefore, $K = E \cup D$ represents the observed part of the PON.*

To represent three types of node pair status, the adjacency matrix $\boldsymbol{A}$ of a PON has entries $a_{ij} \in \{0, 1, \text{null}\}$.

The task of LP in a PON is to predict the connectivity status of node pairs $(i, j) \in U$, and this is based on the available information in $\mathcal{G}$. Remarkably, most NE methods (and LP methods more generally) do not treat disconnected node pairs differently from node pairs with unknown status. This is inevitably true for methods based on random walks (as a random walk cannot transition from node i to j if not known to be connected, regardless of whether known to be disconnected), and true also for many other methods such as those based on matrix decompositions. CNE, however, can be trivially modified to elegantly do so, by maximizing the probability only for the observed part (i.e., $(i, j) \in E \cup D$):

$$P(\mathcal{G}|\boldsymbol{X}) = \prod_{(i,j)\in E} P(a_{ij} = 1|\boldsymbol{X}) \cdot \prod_{(k,l)\notin E\cup U} P(a_{kl} = 0|\boldsymbol{X}).$$

Furthermore, the link probability in CNE is formed analytically because the embedding is found by solving a Maximum Likelihood Estimation (MLE) problem: $\text{argmax}_{\boldsymbol{X}}\ P(\mathcal{G}|\boldsymbol{X})$. Next, we will show how it allows us to quantify the informativeness of the node pairs with unknown status in ALPINE.

3.2 ALPINE

Here, we introduce ALPINE, a pool-based active learning approach [Settles, 2009] for NE with LP as a downstream task. We develop ALPINE for CNE although we stress that our arguments can be applied in principle to any other NE method of which the objective function can be expressed analytically.

ALPINE works by finding an optimal NE for a given PON $\mathcal{G} = (V, E, U)$, selecting one or a few node pairs from U to query, updating the PON with the new knowledge (i.e. node pairs from U found to be connected are moved to E, those unconnected are removed from U), and re-embedding the updated PON. This process can be iterated until the budget is exhausted or until the model is sufficiently accurate.

We will introduce different strategies for selecting the node pairs to query, relying on different *utility functions* $u_{\boldsymbol{A},\boldsymbol{X}} : V \times V \to \mathbb{R}$ in ALPINE which quantify how useful knowing the connectivity status of a node pair is estimated to be for the purpose of increasing the LP accuracy on the node pairs in U, when based on the updated NE (see Sec. 3.3). Specifically, each query strategy will select the next query as:

$$\operatorname*{argmax}_{(i,j)\in U} \; u_{\boldsymbol{A},\boldsymbol{X}}(i,j),$$

for an appropriate utility function $u_{\boldsymbol{A},\boldsymbol{X}}$. In practice, not just the single best node pair (argmax) is selected in each iteration, but the s best ones (further referred to as the 'step size').

Thus, given a PON $\mathcal{G} = (V, E, U)$, an NE model, a query strategy and associated utility function $u_{\boldsymbol{A},\boldsymbol{X}}$, a step size s, and a budget B (number of node pairs in U that can be queried), each iteration of ALPINE works as follows. We initialize the pool of node pairs with unknown status $U(0) = U$ and that of the known part $K(0)$ at step $it = 0$ according to $\boldsymbol{A}(0)$ of $\mathcal{G}(0) = \mathcal{G}$ given.

1. Compute $\boldsymbol{X}^*(it)$ as an optimal embedding of the $\mathcal{G}(it)$;
2. Find the best query $Q(it) \subseteq U(it)$ as the set of $|Q(it)| = \min(s, B)$ elements from $U(it)$ with largest values for the utility function $u_{\boldsymbol{A}(it),\boldsymbol{X}^*(it)}$;
3. Query the oracle for the connectivity status of the node pairs in $Q(it)$, set $U(it+1) \leftarrow U(it) \setminus Q(it)$, and add each $(i, j) \in Q(it)$ revealed as connected to $E(it + 1)$;
4. Set $B \leftarrow B - |U(it)|$, and break if $B == 0$.

If desired, the LP accuracy based on the embedding can be monitored on a hold-out set during these iterations, and one can stop early as soon as the accuracy meets a threshold.

3.3 Query Strategies for ALPINE

Here we first derive a principled utility function based on the concept of V-optimality from OED. The utility function measures the informativeness of the connectivity of a node pair by identifying to what extent its knowledge is expected to minimize the variance of the predictions for the link status of node pairs in U. After that, we also introduce a range of other heuristic query strategies.

V-optimality and Variance Reduction. V-optimality from OED aims to choose the training data points so as to minimize the variance of the predictions of the learned model on the test data points. As the test set U is finite and given in PONs, this aim naturally fits our problem setting. Thus, with g the link prediction function, and with $P^*_{ij} \triangleq g(\boldsymbol{x}^*_i, \boldsymbol{x}^*_j) = P(a_{ij} = 1|\boldsymbol{X}^*)$ the probability of a link between nodes i and j given the CNE embedding, the utility function used in V-optimality is the reduction of $\sum_{(i,j)\in U} \text{Var}(P^*_{ij})$ achieved by querying a particular node pair from U. The challenge to be addressed is thus the computation of the reduction in the variance terms $\text{Var}(P^*_{ij})$. Omitting details, we outline how this can be done.

In ALPINE, CNE finds the optimal embedding $\boldsymbol{X}^*$ as the Maximum Likelihood Estimator (MLE) given a PON with adjacency matrix $\boldsymbol{A}$, i.e., $\boldsymbol{X}^*$ maximizes $P(\mathcal{G}|\boldsymbol{X})$ w.r.t. $\boldsymbol{X}$. The variance of an MLE can be quantified in terms of the Fisher Information [Lehmann and Casella, 2006]. More precisely, the Cramer-Rao bound [Rao, 1992] provides a lower bound on the variance of a MLE by the inverse of the Fisher Information: $\text{Var}(\boldsymbol{X}^*) \succeq \mathcal{I}(\boldsymbol{X}^*)^{-1}$. Although the Fisher Information can often not be computed exactly (as it requires knowledge of the data distribution), it can be effectively approximated by the *observed* information matrix [Efron and Hinkley, 1978]. For CNE, this observed information matrix *for the MLE* $\boldsymbol{x}^*_i$ is given by (proof omitted for brevity):

$$\mathcal{I}(\boldsymbol{x}^*_i) = \gamma^2 \sum_{(i,j)\notin U} P^*_{ij}(1 - P^*_{ij})(\boldsymbol{x}^*_i - \boldsymbol{x}^*_j)(\boldsymbol{x}^*_i - \boldsymbol{x}^*_j)^T,$$

where γ is a CNE-parameter. Thus, we can bound the covariance matrix $\boldsymbol{C}_i$ of node i's MLE embedding $\boldsymbol{x}^*_i$ as $\boldsymbol{C}_i \triangleq \text{Cov}(\boldsymbol{x}^*_i) \succeq \mathcal{I}(\boldsymbol{x}^*_i)^{-1}$.

Using a first-order analysis (details omitted) to decompose $\text{Var}(P^*_{ij})$ into a contribution from each end point as follows:

$$\text{Var}(P^*_{ij}) = \text{Var}_{\boldsymbol{x}^*_i}(P^*_{ij}) + \text{Var}_{\boldsymbol{x}^*_j}(P^*_{ij}), \tag{1}$$

and using the bound on $\text{Cov}(\boldsymbol{x}^*_i)$ for all i, allows bounding the variance on the estimated probabilities $\text{Var}(P^*_{ij})$ by bounding the two terms in the decomposition as follows:

$$\text{Var}_{\boldsymbol{x}^*_i}(P^*_{ij}) \geq \left[\gamma P^*_{ij}(1 - P^*_{ij})\right]^2 (\boldsymbol{x}^*_i - \boldsymbol{x}^*_j)^T \boldsymbol{C}_i (\boldsymbol{x}^*_i - \boldsymbol{x}^*_j), \tag{2}$$

and similar for $\text{Var}_{\boldsymbol{x}^*_j}(P^*_{ij})$. Querying node pair $(i, j) \in U$ will reduce the covariance matrices $\boldsymbol{C}_i$ and $\boldsymbol{C}_j$, as it creates additional information on their optimal values. For example for $\boldsymbol{x}^*_i$ (and similarly for $\boldsymbol{x}^*_j$ leading to $\boldsymbol{C}^i_j$), the new covariance assuming (i, j) has known status, denoted $\boldsymbol{C}^j_i$, is:

$$\boldsymbol{C}^j_i = \left[\boldsymbol{C}^{-1}_i + \gamma^2 P^*_{ij}(1 - P^*_{ij})(\boldsymbol{x}^*_i - \boldsymbol{x}^*_j)(\boldsymbol{x}^*_i - \boldsymbol{x}^*_j)^T\right]^{-1}. \tag{3}$$

Thus this leads to a reduction of the bounds on $\text{Var}_{\boldsymbol{x}^*_i}(P^*_{ij})$ and $\text{Var}_{\boldsymbol{x}^*_j}(P^*_{ij})$, and thus on $\text{Var}(P^*_{ij})$ due to Eq. (1).

Putting things together allows defining the V-optimality utility function, and proves a theorem for computing it:

Definition 2. *The V-optimality utility function $u_{\boldsymbol{A},\boldsymbol{X}^*}$ evaluated at (i, j) quantifies the reduction in the bound on the sum of the variances Var(P^*_{kl}) (see Eqs. (1) and (2)) of all P^*_{kl} with $(k, l) \in U$, achieved by querying node pair $(i, j) \in U$.*

Theorem 1. *The V-optimality utility function is given by:*

$$u_{\boldsymbol{A},\boldsymbol{X}^*}(i,j) = \sum_{k:(i,k)\in U} u^{ik}(i,j) + \sum_{l:(j,l)\in U} u^{jl}(i,j),$$

where

$$u^{ik}(i,j) = (\gamma P^*_{ik}(1 - P^*_{ik}))^2 (\boldsymbol{x}^*_i - \boldsymbol{x}^*_k)^T (\boldsymbol{C}_i - \boldsymbol{C}^j_i)(\boldsymbol{x}^*_i - \boldsymbol{x}^*_k),$$
$$u^{jl}(i,j) = (\gamma P^*_{jl}(1 - P^*_{jl}))^2 (\boldsymbol{x}^*_j - \boldsymbol{x}^*_l)^T (\boldsymbol{C}_j - \boldsymbol{C}^i_j)(\boldsymbol{x}^*_j - \boldsymbol{x}^*_l).$$

Applying the Sherman-Morrison formula to Eq. (3), allows rewriting $u^{ik}(i, j)$ as:

$$\frac{\gamma^4 P^*_{ij}(1 - P^*_{ij})}{1 + \gamma^2 P^*_{ij}(1 - P^*_{ij}) d_{jj}(\boldsymbol{x}^*_i)} \left[P^*_{ik}(1 - P^*_{ik})\right]^2 d_{kj}(\boldsymbol{x}^*_i)^2.$$

where $d_{jj}(\boldsymbol{x}^*_i) = (\boldsymbol{x}^*_i - \boldsymbol{x}^*_j)^T \boldsymbol{C}_i (\boldsymbol{x}^*_i - \boldsymbol{x}^*_j)$ and $d_{kj}(\boldsymbol{x}^*_i) = (\boldsymbol{x}^*_i - \boldsymbol{x}^*_k)^T \boldsymbol{C}_i (\boldsymbol{x}^*_i - \boldsymbol{x}^*_j)$. Thus, unsurprisingly, the variance reduction is always positive.

Table 1: Percentage increase of AUC with a budget of 10% of $|U|$.

Query Strategy	Polbooks			C.elegans		
	s=10	s=50	s=100	s=10	s=50	s=1k
rand.	0.87	0.81	0.88	1.05	1.03	1.04
max-deg.	0.57	0.70	0.70	1.80	1.54	1.32
page-rank.	1.01	0.88	0.84	1.71	1.45	1.35
min-dis.	1.97	1.91	2.09	2.06	1.65	1.36
max-prob.	1.94	2.13	2.17	2.14	1.57	1.29
max-ent.	2.17	2.28	2.29	2.25	1.73	**1.37**
v-opt.	**2.35**	**2.34**	**2.33**	**2.44**	**1.88**	1.36
	USAir			MP_cc		
	s=100	s=500	s=1k	s=300	s=1.5k	s=3k
rand.	0.21	0.22	0.21	0.37	0.37	0.38
max-deg.	0.60	0.58	0.57	0.31	0.32	0.29
page-rank.	0.56	0.57	0.56	0.55	0.56	0.52
min-dis.	0.75	0.75	0.77	0.28	0.29	0.30
max-prob.	0.97	0.93	0.91	0.06	0.06	0.08
max-ent.	**0.99**	**0.95**	**0.93**	0.82	0.82	**0.81**
v-opt.	0.94	0.90	0.87	**0.88**	**0.85**	0.80
	PPI			Blog		
	s=10k	s=30k	s=50k	s=100k	s=300k	s=500k
rand.	0.48	0.50	0.51	0.14	0.13	0.13
max-deg.	0.82	0.85	0.82	0.33	0.32	0.33
page-rank.	0.87	0.88	0.88	0.33	0.33	0.34
min-dis.	0.97	0.97	0.96	0.24	0.23	0.25
max-prob.	1.08	1.07	1.09	0.30	0.28	0.30
max-ent.	1.10	1.09	1.10	0.33	0.31	0.33
v-opt.	**1.14**	**1.16**	**1.16**	**0.37**	**0.35**	**0.37**

Table 2: Runtime evaluation in seconds.

	rand.	max-deg.	page-rank.	min-dis.	max-prob.	max-ent.	v-opt.
Polbooks	2e−4	2e−3	0.04	7e−3	1e−3	9e−4	0.09
C.elegans	2e−5	0.02	0.10	0.05	0.01	8e−3	0.75
USAir	3e−5	0.02	0.14	0.07	0.01	0.01	0.93
MP_cc	2e−5	0.06	1.59	0.19	0.04	0.03	2.86
PPI	3e−5	2.85	3.86	9.62	2.53	2.23	255.61
Blog	0.02	25.57	36.36	74.04	24.05	7.71	3e3

Heuristic Query Strategy

Besides V-optimality, we also propose 5 heuristic utility functions inspired by common active learning query strategies (omitting subscripts from the utility function u for brevity):

- **max-ent.**: $u(i,j) = -P^*_{ij}\log P^*_{ij} - (1-P^*_{ij})\log(1-P^*_{ij})$.
- **max-prob.**: $u(i,j) = P^*_{ij}$.
- **min-dis.**: $u(i,j) = -||\boldsymbol{x}^*_i - \boldsymbol{x}^*_j||_2$.
- **page-rank.**: $u(i,j) = \mathrm{PR}_i + \mathrm{PR}_j$.
- **max-deg.**: $u(i,j) = \sum_{k:(i,k)\notin U} a_{ik} + \sum_{k:(j,k)\notin U} a_{jk}$.

The **max-ent.** query strategy is a specific variant of the popular uncertainty sampling strategy in active learning, with the entropy as the uncertainty measure. The second and third strategies both tend to query node pairs that are linked with high probability. Indeed, this is true by definition for **max-prob.**, and approximately true for **min-dis.** as nearby nodes in the embedding are connected with higher probability. The intuition behind these strategies is that links are often sparse in a network, so that queries that result in the discovery of new links are more informative. The last two are degree-related, and PR_i in **page-rank.** is the PageRank score of i evaluated by treating node pairs with unknown status as disconnected.

4 Experiments

We investigated the following questions: **Q1** Does the behaviour of ALPINE make sense qualitatively? **Q2** Do the different query strategies perform well in predicting the node pairs with unknown status? **Q3** Do the query strategies scale to large networks?

Data. ALPINE is evaluated on 6 real datasets of varying sizes. **Polbooks** is a network consisting of 105 books about the US politics among which 441 connections indicate the co-purchasing relations between the book pairs [Adamic and Glance, 2005] . **C.elegans** is a neural network of C.elegans with 297 neurons and 2,148 synapses as their links [Watts and Strogatz, 1998]. **USAir** is a network of 332 airports connected through 2,126 US Airlines [Handcock *et al.*, 2003]. **MP_cc** network is a twitter network we gathered in April 2019 for the Members of Parliament (MP) in the UK, which originally contains 650 nodes, and we only use its largest connected component of 567 nodes and 35531 friendship (i.e., mutual follow) connections. **PPI** is a protein-protein interaction network with 3,890 proteins and 76,584 interactions [Breitkreutz *et al.*, 2007]. **Blog** is a friendship network of 10312 bloggers from BlogCatalog, containing 333983 connections [Zafarani and Liu, 2009].

4.1 Qualitative evaluation

As illustrated in Fig. 1, we apply ALPINE to the case when a new node is added to a network, and examine the behaviour of the V-optimality query strategy. We take node 39 ('Harry Potter', pentagon node) as newly arrived who has only one initial known connection to node 22 ('Rubeus Hagrid', square node). Thus, the node pairs involving Harry, except the one with Rubeus, all have unknown status.

In the first iteration, the V-optimality query strategy used in ALPINE scores all the node pairs $(i,j) \in U$ and suggests to query ('Harry Potter', 'Arthur Weasley') where 'Arthur Weasley' (node with brightest yellow) is the father of the Weasley family. The other members of Weasley family are also scored high. This indicates that predicting the relationships of 'Harry Potter' with other characters can be improved by first querying possible connections with members of the Weasley family – close allies of 'Harry Potter' and with many connections to other characters.

From the second to the fifth iterations, the top ranked nodes are 'Ginny Weasley', 'Fred Weasley', 'George Weasley', 'Albus Dumbledore'. These early suggestions sketch the relationships of 'Harry Potter' to the entire network, thus allowing it to be well-embedded with just a few queries.

4.2 Quantitative evaluation

We quantitatively evaluate ALPINE with different query strategies for LP on node pairs with unknown status in a PON in an iterative manner. All query strategies proposed in Sec. 3.3 are used. Additionally, a baseline query strategy which samples node pairs uniformly at random from U, is included for comparison. In order to construct PON based on the benchmarks, information on 20% of the node pairs (both links and non-links) is removed, which forms U. Then we apply ALPINE for varying budget B and and a range of step sizes s.

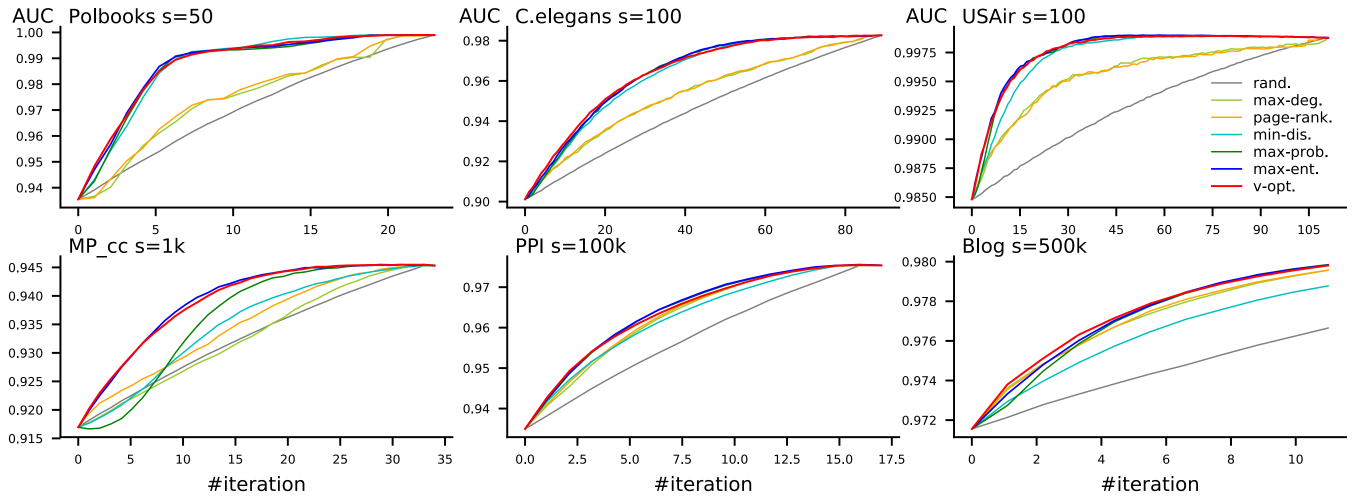

Figure 2: Performance of the AL query strategies in PONs.

Fig. 2 shows the LP accuracy over subsequent iterations for all datasets, where the first 5 are queried with no budget limitation (i.e., $B = |U|$) and **Blog** with $B = 5M$. Hereby, the LP accuracy is quantified as the AUC of all node pairs in $U(0)$ for $it = 0$ with respect to the ground truth. Results are averaged over several $U(0)$s, and the PON with each $U(0)$ is initialized with different random embeddings (10×10 for the first four, 5×5 for the fifth and 4×2 for the last). Scores of the random strategy are further averaged over 5 runs.

All non-random strategies perform consistently and substantially better than the random query strategy, apparently clustering into two groups within which the accuracy is similar. The 4 best strategies are all NE-based: **v-opt.**, **max-ent.**, **max-prob.**, and **min-dis**. With these strategies, ALPINE boosts link prediction accuracy with far fewer queries. One interesting finding is that **max-prob.**, while different in spirit to uncertainty maximization, still performs similar to **max-ent.**. The reason could be that positive links are considered as more informative because real-world networks are usually sparse such that linked node pairs are more informative. The second group consists of **page-rank**, and **max-deg.**, both strategies that do not require the NE. Thus, the link status of high-degree nodes is more informative than that of the random ones, but as a strategy it is inferior to the NE-based ones.

In practice, however, active learning is particularly useful when the budget is small. Thus, we investigated in greater detail the relative performance of the various query strategies for a small budget B, equal to 10% of $|U|$. Table 1 shows the increase in percentage points of the LP AUC compared to the AUC before active learning, and this for three different step sizes. The V-optimality query strategy outperforms the others in most cases, and is close second or third in a few other cases, although **max-prob.** and **max-ent.** are never much worse. As a side result, the Table shows that the LP AUC is relatively insensitive to the step size.

We also evaluated ALPINE for predicting the connectivity to a newly added node, using as few queries as possible, with similar conclusions (details will be in an extended version).

4.3 Scalability

The runtime analysis (on a server with Intel Core i5 CPU 2.30GHz and 8GB RAM) of ALPINE with different query strategies is shown in Table 2. The embedding dimension is set to 8, and the removed information from the original network is 20% of the node pairs. The results are averaged over 10 random runs. It shows that the computation time per iteration of the V-optimality strategy increases dramatically as the network size grows. Given this, and their competitive performance in terms of LP accuracy, **max-prob.** and **max-ent.** are probably the query strategies of choice on larger problems.

5 Conclusion

Link prediction is an important task in network analysis, tackled increasingly using network embeddings. It is particularly important in partially observed networks, where finding out whether a pair of nodes is linked is time-consuming or costly, such that for a large number of node pairs it is not known if they are connected or not.

We propose to make use of active learning in this setting, and introduce ALPINE, a specific active learning approach for link prediction in such partially observed networks using network embedding. We first derived a principled query strategy that generalizes the notion of V-optimality from optimal experimental design to the current setting, identifying those node pairs which, if queried, will maximally reduce the variance on the link scores for the node pairs with unknown connectivity status. Additionally, several heuristic active learning strategies are also proposed as computationally efficient alternatives. Empirical evaluations show that ALPINE with the V-optimality query strategy performs best overall, albeit at a relatively high computational cost, while two intuitive heuristics achieve similar accuracies and scale to larger networks. All query strategies outperform by a large margin the random query strategy.

As future work, we plan to further improve the scalability of ALPINE by e.g., using incremental embeddings at each iteration.

Acknowledgments

The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC Grant Agreement no. 615517, from the Flemish Government under the "Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen" programme, from the FWO (project no. G091017N, G0F9816N, 3G042220), and from the European Union's Horizon 2020 research and innovation programme and the FWO under the Marie Sklodowska-Curie Grant Agreement no. 665501. We thank Ahmad Mel for helping collecting the MP network.

References

[Adamic and Glance, 2005] Lada A. Adamic and Natalie Glance. The political blogosphere and the 2004 u.s. election: Divided they blog. In *Proc. of the 3rd International Workshop on Link Discovery*. ACM, 2005.

[Aggarwal *et al.*, 2014] Charu C Aggarwal, Xiangnan Kong, Quanquan Gu, Jiawei Han, and S Yu Philip. Active learning: A survey. In *Data Classification*, pages 599–634. Chapman and Hall/CRC, 2014.

[Atkinson *et al.*, 2007] Anthony Atkinson, Alexander Donev, et al. *Optimum experimental designs, with SAS*, volume 34. Oxford University Press, 2007.

[Bilgic *et al.*, 2010] Mustafa Bilgic, Lilyana Mihalkova, and Lise Getoor. Active learning for networked data. In *Proc. of ICML*, pages 79–86, 2010.

[Breitkreutz *et al.*, 2007] Bobby-Joe Breitkreutz, Chris Stark, et al. The biogrid interaction database: 2008 update. *Nucleic acids research*, 36:D637–D640, 2007.

[Brinker, 2003] Klaus Brinker. Incorporating diversity in active learning with support vector machines. In *Proc. of ICML*, pages 59–66, 2003.

[Cai *et al.*, 2017] Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. Active learning for graph embedding. *arXiv preprint arXiv:1705.05085*, 2017.

[Cesa-Bianchi *et al.*, 2012] Nicolo Cesa-Bianchi, Claudio Gentile, Fabio Vitale, and Giovanni Zappella. A fast active learning algorithm for link classification. In *Proc. of ICTCS*, 2012.

[Cesa-Bianchi *et al.*, 2013] Nicolo Cesa-Bianchi, Claudio Gentile, et al. Active learning on trees and graphs. *arXiv preprint arXiv:1301.5112*, 2013.

[Chen *et al.*, 2014] Ke-Jia Chen, Jingyu Han, and Yun Li. Hallp: A hybrid active learning approach to link prediction task. *JCP*, 9(3):551–556, 2014.

[Chen *et al.*, 2019] Xia Chen, Guoxian Yu, et al. Activehne: Active heterogeneous network embedding. In *Proc. of IJCAI*, 2019.

[Cohn *et al.*, 1996] David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. Active learning with statistical models. *J. Artif. Int. Res.*, 4(1):129?145, 1996.

[Efron and Hinkley, 1978] Bradley Efron and David V Hinkley. Assessing the accuracy of the maximum likelihood estimator: Observed versus expected fisher information. *Biometrika*, 65(3):457–483, 1978.

[Evans *et al.*, 2014] Craig Evans, Josh Friedman, Efe Karakus, and Jatin Pandey. Potterverse. https://github.com/efekarakus/potter-network, 2014.

[Grover and Leskovec, 2016] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proc. of KDD*, pages 855–864, 2016.

[Guillory and Bilmes, 2009] Andrew Guillory and Jeff A Bilmes. Label selection on graphs. In *Proc. of NeurIPS*, pages 691–699, 2009.

[Handcock *et al.*, 2003] Mark S Handcock, David Hunter, et al. Statnet: An R package for the statistical modeling of social networks. *Web page http://www.csde.washington.edu/statnet*, 2003.

[Jia *et al.*, 2019] Junteng Jia, Michael T Schaub, Santiago Segarra, and Austin R Benson. Graph-based semi-supervised & active learning for edge flows. In *Proc. of KDD*, pages 761–771, 2019.

[Kang *et al.*, 2019] Bo Kang, Jefrey Lijffijt, and Tijl De Bie. Conditional network embeddings. In *Proc. of ICLR*, 2019.

[Kashima *et al.*, 2009] Hisashi Kashima, Tsuyoshi Kato, et al. Link propagation: A fast semi-supervised learning algorithm for link prediction. In *Proc. of SDM*, pages 1100–1111, 2009.

[Kipf and Welling, 2017] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *Proc. of ICLR*, 2017.

[Kong *et al.*, 2011] Xiangnan Kong, Wei Fan, and Philip S Yu. Dual active feature and sample selection for graph classification. In *Proc. of KDD*, pages 654–662, 2011.

[Lehmann and Casella, 2006] Erich L Lehmann and George Casella. *Theory of point estimation*. Springer Science & Business Media, 2006.

[Martínez *et al.*, 2017] Víctor Martínez, Fernando Berzal, and Juan-Carlos Cubero. A survey of link prediction in complex networks. *ACM Comput. Surv.*, 49(4):69, 2017.

[Ostapuk *et al.*, 2019] Natalia Ostapuk, Jie Yang, and Philippe Cudré-Mauroux. Activelink: deep active learning for link prediction in knowledge graphs. In *Proc. of WWW*, pages 1398–1408, 2019.

[Perozzi *et al.*, 2014] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proc. of KDD*, pages 701–710, 2014.

[Rao, 1992] C Radhakrishna Rao. Information and the accuracy attainable in the estimation of statistical parameters. In *Breakthroughs in statistics*, pages 235–247. 1992.

[Settles, 2009] Burr Settles. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009.

[Watts and Strogatz, 1998] Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world' networks. *nature*, 393(6684):440, 1998.

[Yang *et al.*, 2014] Zhilin Yang, Jie Tang, and Yutao Zhang. Active learning for streaming networked data. In *Proc. of CIKM*, pages 1129–1138, 2014.

[Yang *et al.*, 2016] Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In *Proc. of ICML*, 2016.

[Zafarani and Liu, 2009] R. Zafarani and H. Liu. Social computing data repository at ASU, 2009.

[Zhang and Oles, 2000] Tong Zhang and Frank J. Oles. A probability analysis on the value of unlabeled data for classification problems. In *Proc. of ICML*, 2000.