Inferring Attitude in Online Social Networks Based on Quadratic Correlation

Cong Wang[1]* and Andrei A. Bulatov[1]

Simon Fraser University
{cwa9,abulatov}@sfu.ca

Abstract. The structure of an online social network in most cases cannot be described just by links between its members. We study online social networks, in which members may have certain attitude, positive or negative, toward each other, and so the network consists of a mixture of both positive and negative relationships. Our goal is to predict the sign of a given relationship based on the evidences provided in the current snapshot of the network. More precisely, using machine learning techniques we develop a model that after being trained on a particular network predicts the sign of an unknown or hidden link. The model uses relationships and influences from peers as evidences for the guess, however, the set of peers used is not predefined but rather learned during the training process. We use quadratic correlation between peer members to train the predictor. The model is tested on popular online datasets such as Epinions, Slashdot, and Wikipedia. In many cases it shows almost perfect prediction accuracy. Moreover, our model can also be efficiently updated as the underlying social network evolves.

Keywords: Signed Networks, machine learning, quadratic optimization

1 Introduction

Online social networks provide a convenient and ready to use model of relationships between individuals. Relationships representing a wide range of social interactions in online communities are useful for understanding individual attitude and behaviour as a part of a larger society.

While the bulk of research in the structure on social networks tries to analyze a network using the topology of links (relationships) in the network [21], relationships between members of a network are much richer, and this additional information can be used in many areas of social networks analysis. In this paper we consider signed social networks, which consist of a mixture of both positive and negative relationships. This type of networks has attracted attention of researchers in different fields [6, 10, 17]. This framework is also quite natural in

* We'd like to thank Dr.Jian Pei for his valuable suggestions and feedbacks on our work.

recommender systems [3]where we can exploit similarities as well as dissimilarities between users and products.

Over the last several years there has been a substantial amount of work done studying signed networks, see, e.g. [14, 7, 8, 5, 12, 15, 23, 24]. Some of the studies focused on a specific online network, such as Epinions [8, 18], where users can express trust or distrust to others, a technology news site Slashdot [12, 13], whose users can declare others 'friends' or 'foes', and voting results for adminship of Wikipedia [5]. Others develop a general model that fits several different networks [7, 14]. We build upon these works and attempt to combine the best in the two approaches by designing a general model that nevertheless can be tuned up for specific networks.

Edge sign prediction. Following Guha et al. [8] and Kleinberg et al. [14], [17] we consider a signed network as a directed (or undirected) graph, every edge of which has a sign, either positive to indicate friendship, support, approval, or negative to indicate enmity, opposition, disagreement. In the edge sign prediction problem, given a snapshot of the signed network, the goal is to predict the sign of a given link using the information provided in the snapshot. Thus, the edge sign problem is similar to the much studied link prediction problem [16, 11], only we need to predict the sign of a link rather than the link itself.

Several different approaches have been taken to tackle this problem. Kunegis et al. [4] studied the friends and foes on Slashdot using network characteristics such as clustering coefficient, centrality and PageRank; Guha et al. [8] used propagation algorithms based on exponentiating the adjacency matrix to study how trust and distrust propagate in Epinion. Later Kleinberg et al. [14] took a machine learning approach to identify features, such as local relationship patterns and degree of nodes, and their relative weight, to build a general model predicting the sign of a given link. They train their predictor on some dataset, to learn the weights of these features by logistic regression.

Our contribution. In this paper we also take the machine learning approach, only instead of focusing on a particular network or building a general model across different networks, we build a model that is unique to each individual network, yet can be trained on different networks. We suggest several new features into both the modeling of signed networks and the method of processing the model.

The basic assumption of our model is that users' attitude can be determined by the opinions of their peers in the network (compare to the balance and status theories [6] from social psychology, see [14]). Intuitively, peer opinions are guesses from peers on the sign of the link from a source to target node. We assume that peer opinions are only partially known, some of them are hidden. We introduce two new components into the model: set of trusted peers and influence.

Not all peer opinions are equally reliable, and we therefore choose a set of trusted peers whose opinions are important in determining the user's action. The set of trusted peers is one of the features our algorithm learns during the training phase. The algorithm forms a set of trusted peers for each individual node. The optimal composition of such a set is not quite trivial, because even trusted peers

may disagree, and sometimes it is beneficial to have trusted peers who disagree. Thus, the set of trusted peers of a node has to form a wide knowledge base on other nodes in the network.

While peer opinions provide important information, this knowledge is sometimes incomplete. Relying solely on peer opinions implies that the attitude of a user would always agree with the attitude of a peer. However, it also matters is how this opinion correlates with the opinion of the user we are evaluating. To take this correlation into account we introduce another feature into the model, influence. Suppose the goal is to learn the sign of the link between user A and user B, and C is a peer of A. Then if A tends to disagree with C, then positive attitude of C towards B should be taken as indication that A's attitude towards B is less likely to be positive. The opinion of C is then considered to be the product of his attitude towards B and his influence on A. Usually, influence is not given in the snapshot of the network and has to be learned together with other unknown parameters. We experiment with different ways of defining peer opinion, and found that using relationships and influences together is more effective than using relationships alone.

To learn the weights of features providing the best accuracy we use the standard quadratic correlation technique from machine learning [9]. This method involves finding an optimum of a quadratic polynomial, and while being relatively computationally costly, tends to provide very good accuracy. To mitigate the cost of computation we use two approaches. Firstly, we apply several techniques to split the problem and avoid solving large quadratic problems. Secondly, we attempt to make the main algorithm independent on a specific tool of quadratic optimization so that this step consuming the bulk of processing time can be easily improved as better solvers appear.

2 Approach

In our method, we start with the underlying model of a network, then proceed to the machine learning formulation of the edge sign prediction problem, and finally describe the method to solve the resulting quadratic optimization problem.

2.1 Underlying Model

We are given a snapshot of the current state of a network. Such a snapshot is represented by a directed graph $G = (V, E)$, where nodes represent the members of the network and edges represent the links (relationships). Some of the links are signed to indicate positive or negative relationships. Let $s_{x,y}$ denote the sign of the relationship from x to y in the network. It may take two different values, $\{-1, 1\}$, indicating negative and positive relationships respectively.

To estimate the sign $s_{x,y}$ of a relationship from x to y, we collect peer opinions. In different versions of the model a peer can be any node of the network, or any node linked to x. Let $p^z_{x,y} \in \{-1, 0, 1\}$ denote the peer opinion of peer z on the sign $s_{x,y}$. When $p^z_{x,y} = 1$ or $p^z_{x,y} = -1$, it indicates that the z believes

that $s_{x,y} = 1$ or $s_{x,y} = -1$ respectively. When $p^z_{x,y} = 0$ that means z does not have enough knowledge to make a valid estimation.

Another assumption made in our model is that not every peer can make a reliable estimation. Therefore we divide all peers of a node into two categories, and count the opinions only of the peers from the first category, trusted peers. The problem of how to select a set of trusted peers and use their opinions for the estimation will be addressed later. Let P_x denote the set of trusted peers of a. We estimate the sign $s_{x,y}$ of a relationship from x to y by collecting the opinions of peers $z \in P_x$. If the sum of the opinions is nonnegative, then we say $s_{x,y}$ should be 1, otherwise, it should be -1. This can be expressed as,

$$s_{x,y} = sign\left(\sum_{z \in P_x} p^z_{x,y}\right).$$

2.2 Machine learning approach

Our approach to selecting an optimal set of trusted peers is to consider the quadratic correlations between each pair of peers. The overall performance of a set of peers is determined by the sum of the individual performances of each of them together with the sum of their performance in pairs. The individual performance measures the accuracy of individual estimations, while the pairwise performance measures the degree of difference between the estimations of the pair of peers. We want to maximize the accuracy of each individual and the diversity of each pair at the same time.

Our goal is to use the information in G to build a predictor $S(x, y)$ that predicts the sign $s_{x,y}$ of an unknown relationship from x to y with high accuracy. Function $S(x, y)$ is defined as the sign of the sum of peer opinions as follows. Let

$$F_x(y) = \sum_{z \in P_x} p^z_{x,y} \tag{1}$$

be the sum of individual peer opinions. Then set

$$S(x, y) = \begin{cases} 1, & \text{if } F_x(y) \geq 0\ , \\ -1, & \text{if } F_x(y) < 0. \end{cases}$$

Since P_x is unknown, we introduce a new variable $w_{z,x} \in \{0, 1\}$ which indicates if a node $z \in V$ should be included into set P_x. Hence, we rewrite (1) using the characteristic function $w_{z,x}$ as,

$$F_x(y) = \sum_{z \in V} w_{z,x} p^z_{x,y}. \tag{2}$$

Quadratic optimization problem. We are now ready to set the machine learning problem. A training dataset (a subset of G) is given. Every entry of the training dataset is a known edge along with its sign. Let a training dataset be $D =$

$\{(x_i, y_i, s_{x_i,y_i})|i = 1, ..., M\}$. The goal is to minimize the objective function, finding the optimal weight vector $w = \{w_x|x \in V\}$, where $w_x = \{w_{z,x}|z \in V\}$. We use machine learning methods [9] to train the predictor $S(x, y)$ and learn an optimal weight vector such that the objective function below is minimized.

$$w^{opt} = argmin_w \left(\sum_{(x,y,s_{x,y}) \in D} \left(\frac{1}{N} \sum_{z \in V} (w_{z,x} p^z_{x,y} - s_{x,y})^2 + \lambda |w| \right) \right).$$

Note that there will be more details on peer opinion terms $p^z_{x,y}$.

2.3 Peer opinion variants

As mentioned earlier, we are going to test our model using different peer opinion formulations. First, let $s'_{x,y}$ be extension of $s_{x,y}$ to edges with unknown sign and also to pairs of nodes that are not edges defined by

$$s'_{x,y} = \begin{cases} s_{x,y}, \text{ if } s_{x,y} \text{ exists,} \\ 0, \quad \text{otherwise.} \end{cases}$$

Simple-adjacent. The simplest option, later referred to as *Simple-adjacent*, is based on the given information, to formulate peer opinions using existing relationships from peers to the target node, that is, $p^z_{x,y} = s'_{z,y}$.

Standard-pq. In the *Standard-pq* option the influences $r_{z,x} \in \{-1, 0, 1\}$ associated with each pair of vertices z, x is an unknown parameter. A positive influence, $r_{z,x} = 1$, indicates that the attitude of z affects x positively, while a negative influence, $r_{z,x} = -1$ indicates that the attitude of z affects x negatively, and our expectation of $p^z_{x,y}$ based on z's opinion $s'_{z,y}$ has to be reversed, that is, $p^z_{x,y} = s'_{z,y} r_{z,x}$. Also, in the Standard-pq mode we consider nonadjacent nodes as potential peers to accommodate the problem of possible missing edges. Details of the Standard-pq mode is explained in the experiment section. Since the standard formulation gives us the best result in experiments, we use it throughout our discussion. Using the standard formulation, we rewrite Equation (2) as

$$F_x(y) = \sum_{z \in V} w_{z,x} s'_{z,y} r_{z,x}. \tag{3}$$

Standard-adjacent. Finally, in the *Standard-adjacent* option the peers of x are restricted to the neighbours of x. $F_x(y) = \sum_{z \in N(x)} w_{z,x} s'_{z,y} r_{z,x}$. The rest is defined in the same way as for the *Standard-pq* option.

2.4 Simplifying the model

In our model, we are given a directed complete graph $G = (V, E)$. In (3), both $w_{z,x}$ and $r_{z,x}$ are unknown parameters. Since $r_{z,x} \in \{-1, 0, 1\}$, we can reduce the

number of unknown parameters by considering all possible values of $r_{z,x}$, and rewriting $F_x(y)$ as $F_x(y) = \sum_{z \in V} w^+_{z,x} s'_{z,y} - w^-_{z,x} s'_{z,y}$,where $w_{z,x} = w^+_{z,x} + w^-_{z,x}$ for $w^+_{z,x}, w^-_{z,x} \in \{0, 1\}$. If $w^+_{z,x} = 1$, then $z \in P_x$ and $r_{z,x} = 1$. Similarly, $w^-_{z,x} = 1$ indicates that $z \in P_x$ and $r_{z,x} = -1$. When both $w^-_{z,x} = 0$ and $w^+_{z,x} = 0$, then $z \notin P_x$. When $w^-_{z,x} = 1$ and $w^+_{z,x} = 1$, the two term cancel out each other. Moreover, the regularization term ensures such case will not happen as $w^-_{z,x} = w^+_{z,x} = 0$ is always better than $w^-_{z,x} = w^+_{z,x} = 1$. Although $r_{z,x}$ can take three possible values, there are only two terms in Equation (3) since when $r_{z,x} = 0$, the term is also zero regardless of the value of $s'_{z,y}$.

Now to minimize the objective function, we need to determine the optimal weight vector $w = \{w_x | x \in V\}$ where $w_x = \{w^+_{z,x}, w^-_{z,x} | z \in V\}$ such that

$$w^{opt} = argmin_w \left(\sum_{(x,y,s_{x,y}) \in D} \left(\frac{1}{N} \sum_{z \in V} (w^+_{z,x} - w^-_{z,x}) s'_{z,y} - s_{x,y} \right)^2 + \lambda |w| \right). \tag{4}$$

From the definition, we know that w_x and w_y are independent for different nodes x and y. Instead of solving for w^{opt} directly, we can solve w^{opt}_x for each $x \in V$ separately, and then combine their values to get $w^{opt} = \{w^{opt}_x | x \in V\}$

$$w^{opt}_x = argmin_{w_x} \left(\sum_{(x,y,s_{x,y}) \in D} \left(\frac{1}{N} \sum_{z \in V} (w^+_{z,x} - w^-_{z,x}) s'_{z,y} - s_{x,y} \right)^2 + \lambda |w_x| \right). \tag{5}$$

Now, instead of solving a QUBO of size $2n^2$, we could solve n QUBOs of size $2n$ separately which can be solved approximately by a heuristic solver. Another approach (similar to [20]) is to further simplify the problem, as it is still challenging to solve each of these size $2n$ QUBOs exactly.

Breaking down the problem. In order to find a good approximation of the optimal solution to the QUBO defined by (4), we could break it down to much smaller QUBOs. Given a subset $U \subset V$, let $w_{x,U} = \{w^+_{z,x}, w^-_{z,x} | z \in U\}$, and define a restricted optimization problem as follows

$$w^{opt}_{x,U} = argmin_{w_{x,U}} \left(\sum_{(x,y,s_{x,y}) \in D} \left(\frac{1}{N} \sum_{z \in U} (w^+_{z,x} - w^-_{z,x}) s'_{z,y} - s_{x,y} \right)^2 + \lambda |w_{x,u}| \right). \tag{6}$$

The optimal solution w^{opt}_x can be approximated by combining w^{opt}_{x,U_i} for $V = \bigcup_{i=1}^m U_i$. Next, we describe a method to decompose V and to combine w^{opt}_{x,U_i}.

2.5 Method

The optimization problem defined by (4) is a quadratic unconstrained binary optimization problem which is NP-hard in general. To solve the problem, we use

two approaches. First, we solve the problem for each individual node separately, as given by (5), using METSLib Tabu search heuristics. The clear setback of this method is that for large problems the tabu search does not find the best solution. Second, we apply a similar method as described in [20] to reduce the size of the problem dramatically. In order to do that the variables are first ordered according to their individual prediction error: For each data point $(a, u, s_{a,u})$ in the training dataset, we count e_{v_-}, the number of instances $p^v_{a,u} \neq s_{a,u}$ when $r_{v,a} = -1$, and e_{v_+}, the number of instances $p^v_{a,u} \neq s_{a,u}$ when $r_{v,a} = 1$ separately. Note that since $p^v_{a,u} = r_{v,a}s'_{v,u}$, we can compute this number; and that if u is not a neighbour of v then it contributes to both e_{v_-} and e_{v_+}. Then, we replace v by v_+ and v_- with individual prediction error, e_{v_+} and e_{v_-} respectively. The subset U is iteratively selected by picking the first d nodes in the list that are not yet considered. The value of d is an important parameter of the algorithm and is selected manually at the beginning of the algorithm. In the experiment section, we show how the prediction accuracy changes as the size of d changes. The sorting and selecting processes not only reduce the amount of computation, but also allow us to consider the relevant nodes first. The small subproblems are now solved by the brute-force method or Cplex. Algorithm 1 describes the method we use to solve each subproblem. Algorithm 2 uses Algorithm 1 as a subroutine and explains how the problem is broken down into subproblems and also how to combine the solutions of subproblems to obtain an approximate solution.

Algorithm 1 Learn the parameters for a subset

Require: training dataset: TD, validation dataset: VD, a subset of nodes: U
Ensure: values of w and $Z \subset U$ where Z is the set of trusted peers
  $Z = \emptyset$, $e_{min} = |TD|$
  for $\lambda = \lambda_{min}$ to λ_{max} **do**
    $Z_{current} = \emptyset$
    solve the optimization $w^{opt} = argmin_w(\sum_{i=1}^{M}(\frac{1}{N}\sum_{z\in U}(w_z^+ - w_z^-)s_{z,y} - s_{x,y})^2 + \lambda|w|)$
    if $w_z == 1$ **then**
      $Z_{current} = Z_{current} \cup z$
    end if
    Measure the validation error e_{val} on VD using $Z_{current}$
    if $e_{val} < e_{min}$ **then**
      $Z = Z_{current}$, $e_{min} = e_{val}$
    end if
  end for

2.6 Running Time

Although the QUBO problem is NP-hard in general, the proposed algorithm can be very efficient when using the right solver and right parameter d. Let $T(d)$

Algorithm 2 solve the optimization problem

Require: training dataset: TD, validation dataset: VD, The size of the subset: d
Ensure: values of w_z for $z \in V$, and the set of trusted peers Z
$e_{old} = |TD|$, $e_{new} = |TD|$-1, $Z, Z_{current} = \emptyset$
sort nodes of V by their individual prediction errors in increasing order
U = the first d nodes in V
while $e_{old} > e_{new}$ **do**
$Z = Z_{current} \cup Z$
w_z, $Z_{current}$ = Algorithm 1(TD, VD, U) e_{old}=e_{new}
Measure the validation error e_{new} on vd using Z
update U with the next d nodes in V
end while

denote the time of solving a size d optimization problem defined by (6), and k_λ be the number of λ values tested. The running time of Algorithm 1 is $O(k_\lambda T(d))$. Let n_v denote the number of neighbours of a node v. By our definition, n_v is at most $\|V\|$. In Algorithm 2, Algorithm 1 is repeated at most $\frac{n_v}{d}$ times. Therefore, the running time of Algorithm 2 is $O(\frac{n_v k_\lambda}{d} T(d))$. Traditionally, the best λ for a model is determined before training stage through cross validations. In our model, we are building a personalized predictor for each node. Hence, we need to pick a λ for each node separately. During the training stage, we test $k_\lambda = 25$ different values in the range [$\lambda_{min} = .001$, $\lambda_{max} = 0.25$], and use the λ which gives the lowest validation error. Since k_λ is a constant, the running time crucially depends on the efficiency of the QUBO solver. For example, in the experiments, when $d = 10$ and $T(d)$ is limited to 1 sec for METSLib-solver [2], the predictor for a node can be determined instantly. Yet, when $d = 10$, using Cplex-solver [1] would take a few seconds to minutes to determine the predictor for a node.

Since the main focus of our paper is on the prediction accuracy of the model, we do not measure and compare the running times for different solvers, and we keep our experiments on a standard set of solvers rather than some exclusive ones. Although the model is currently limited by the power of the software solvers, it has shown a good potential. Its performance will improve as better solutions are found by more efficient solvers. One of such solvers could be the quantum system which is rapidly developing at D-wave System. A recent work [19] which compares the performances of different software solvers with D-wave hardware on different combinatorial optimization problems shows promise.

3 Experiment

3.1 Datasets

We use three datasets borrowed from [14]. In order to make comparison possible the datasets are unchanged rather than updated to their current status. The dataset statistics is therefore also from [14] (see Table 1).

Table 1. Basic statistics on the datasets

Dataset	Epinions	Slashdot	Wikipedia
Nodes	119217	82144	7118
Edges	841200	549202	103747
+1 edges	85.0%	77.4%	78.7%
-1 edges	15.0%	22.6%	21.2%

Table 2. Number of edges passing the threshold

Dataset	Epinions	Slashdot	Wikipedia
(p,q)=(15,20)	247725	25436	51372
embeddedness 25 ([14])	205796	21780	28287

3.2 Parameters of datasets

In our experiment, we split each dataset into two parts. We randomly pick one tenth of the dataset for testing. The remaining dataset is used for training. The training dataset is split into two equal parts, half for training and half for validating during the training process.

When the dataset is sparse, it is hard to build good classifiers due to the lack of training and testing data. In order to get a better understanding of the performance of the model, edge embeddedness of an edge uv is introduced in [14, 7] as the number of common neighbours (in the undirected sense) of u and v. Instead of testing the model over the entire dataset, they only consider the performance restricted to subsets of edges of different levels of minimum embeddedness. For example, Kleinberg et al. [14] restrict the analysis to edges with minimum embeddedness 25.

Similarly, we also introduce two parameters that restrict the analysis of our model. Instead of considering the edge embeddedness of a link, we consider the node embeddedness of the source and target nodes. The first parameter, p, controls which nodes are considered peers of a given node v. A node u is considered a peer of v if it has at least p common neighbours with v. When $p = 0$, we consider every node in the network as a peer of v. The second parameter, q, restricts the set of links whose sign we attempt to predict. We try to predict the sign $s_{u,v}$, only if u is connected to at least q peers of v.

In Table 3, we show the dependence of the prediction accuracy of our model on different values of p and q. The data in Table 3 is obtained using option *Standard-pq* with $d = 10$ solved by Cplex. As p and q grows, the performance of the model clearly improves. However, increasing the values of p and q severely restricts the set of nodes that can be processed. We choose somewhat optimal values of these parameters, $q = 20$ and $p = 15$ and use them in the rest of our experiments. It is also worth to notice that values $q = 20$ and $p = 15$ are less restrictive than edge embeddedness 25. As shown in Table 2, more edges pass the $q = 20$ and $p = 15$ threshold than the edge embeddedness 25 threshold.

3.3 Experimental results

As explained before, our model depends on several parameters: internal parameters, such as, the peer opinion variant and the method of solving the QUBO, and

Table 3. Prediction Accuracy for Different Values for p, q

Dataset	Epinions	Slashdot	Wiki
(p,q)=(10,0)	91.7%	84.2%	85.0%
(p,q)=(10,10)	92.8%	91.6%	86.5%
(p,q)=(10,20)	93.7%	93.9%	86.6%
(p,q)=(10,30)	95.6%	95.1%	87.6%
(p,q)=(15,0)	93.7%	87.7%	85.2%
(p,q)=(15,10)	95.8%	96.1%	86.3%
(p,q)=(15,20)	96.2%	97.9%	86.9%
(p,q)=(15,30)	96.3%	96.0%	88.5%
(p,q)=(20,0)	93.5%	87.4%	85.0%
(p,q)=(20,10)	96.2%	98.1%	86.8%
(p,q)=(20,20)	96.3%	98.6%	86.9%
(p,q)=(20,30)	96.5%	99.2%	89.0%

Table 4. Prediction Accuracy of Balanced Approach

Dataset	Epinions	Slashdot	Wiki
Standard-pd (average)	85.14 %	82.82%	62.58%
false negative	0.84%	0.50%	2.10%
false positive	28.89%	33.8%	72.6%
Standard-pq (balanced)	89.36%	85.37%	76.81%
false negative	6.20%	14.37%	22.5%
false positive	22.93%	15.35%	23.90%
All123 ([14])	93.42%	93.51%	80.21%
EIG ([8])	85.30%	N/A	N/A

external parameter, such as, balancing the dataset. We first make the comparison for different internal parameters using the original unbalanced datasets.

In Table 6, we compare the performance of our model using different peer opinion formulations. As shown in the table, standard formulations (*Standard-adjacent*, *Standard-pq*) have better prediction accuracy then the simple formulation (*Simple-adjacent*), so it is useful to introduce influences. For Slashdot and Wikipedia, restricting peers to neighbours (*Standard-adjacent*) is not as effective as using the set of nodes with at least $p = 15$ common neighbours as peers (*Standard-pq*), although the difference is neglegible. Surprisingly, for Epinions, it is slightly better to only consider neighbours as peers. We compare the results with those of [7] (HOC-5) and [14] (All123). Unfortunately, Kleinberg et al. [14] provide only a (somewhat wide) range of the results their model produces on such datasets. Yet,even such partial results allow us to conclude that collecting opinions from trusted peers is an effective method to infer people's attitude.

Table 5. Prediction Accuracy of Different Values of d

Dataset	Epinions	Slashdot	Wiki
d=10 (exact)	96.36%	98.00%	87.69%
d=10 (cplex)	96.17%	97.31%	86.98%
d=25 (cplex)	96.20%	97.52%	85.60%
METSLib	93.03%	96.79%	86.22%

Table 6. Prediction Accuracy of Different Algorithms

Dataset	Epinions	Slashdot	Wiki
Standard-adj	96.59%	97.93%	87.29%
Standard-pq	96.36%	98.00%	87.69%
Simple-adj	95.93%	97.68%	87.03%
HOC-5 ([7])	90.80%	84.69%	86.05%
All123 ([14])	90-95%	90-95%	N/A
EIG ([8])	93.60%	N/A	N/A

In Table 5, we compare the performance of our model with different values of d. We expected the prediction accuracy to increase as the value of d increases. However, experimental result shows that it is not the case. Limited

by the strength of each solver, accuracy of the algorithm is very sensitive to the quality of approximation. But still, we think the prediction accuracy should increase if we can find a better solution for the problem for larger value of d.

In [8] and [14] the authors use certain techniques to test their approaches on unbiased datasets. They use, however, different ways to balance the dataset and/or results. For instance, [8] does not change the dataset (Epinions), but, since the dataset is biased toward positive links, they find the error ratio separately for positive and negative links, and then average the results. More precisely, they test the method on a set of randomly sampled edges that naturally contains more positive edges. Then they record the error rate on all negative edges, sample randomly the same number of positive edges (from the test set), find the error rate on them, and report the mean of the two numbers.

The approach of [14] is different. Instead of balancing the results they balance the dataset itself. In order to do that they keep all the negative edges in the datasets, and then sample the same number of positive edges removing the rest of them. All the training and testing is done on the modified datasets.

Although we have reservations about both approaches, we tested our model in both settings. The results are shown in Table 4. Observe that since our approach is to train the predictor for a particular dataset rather than finding and tuning up general features as it is done in [8] and [14], and the test datasets are biased toward positive edges, it is natural to expect that predictions are biased toward positive edges as well. This is clearly seen from Table 4. We therefore think that average error rate does not properly reflect the performance of our algorithm.

In the case of balanced datasets our predictor does not produce biased results, again as expected. This, however, is the only case when its performance is worse than some of the previous results. One way to explain this is to note that density of the dataset is crucial for accurate predictions made by the quadratic correlation approach. Therefore we had to lower the embeddedness threshold used in this part of the experiment to $p = 5$, $q = 5$, while [14] still tests only edges of embeddedness at least 25.

4 Conclusion

We have investigated the link sign prediction problem in online social networks with a mixture of both positive and negative relationships. We have shown that a better prediction accuracy can be achieved using personalized features such as peer opinions. Although the improvement upon previous results is not significant, a nearly perfect prediction accuracy is hard to achieve. Another advantage of the model is that it accommodates the dynamic nature of online social networks by building a predictor for each individual nodes independently. This enables fast updates as the underlying network evolves over time.

References

1. *IBM ILOG CPLEX Optimizer*, http://goo.gl/uKIx6K, 2010.

2. *METSLib*, https://projects.coin-or.org/metslib, 2011.
3. P.Avesani, P.Massa, and R.Tiella, *A trust-enhanced recommender system application: Moleskiing*, SAC, 2005, pp. 1589–1593.
4. M.Brzozowski, T.Hogg, and G.Szabó, *Friends and foes: ideological social networking*, CHI, 2008, pp. 817–820.
5. M.Burke and R.Kraut, *Mopping up: modeling Wikipedia promotion decisions*, CSCW, 2008, pp. 27–36.
6. D.Cartwright and F.Harary, *Structure balance: A generalization of Heider's theory*, Psychological Review **63** (1956), no. 5, 277–293.
7. K.-Y.Chiang, N.Natarajan, A.Tewari, and I.Dhillon, *Exploiting longer cycles for link prediction in signed networks*, CIKM, 2011, pp. 1157–1162.
8. Ramanathan V. Guha, Ravi Kumar, Prabhakar Raghavan, and Andrew Tomkins, *Propagation of trust and distrust*, WWW, 2004, pp. 403–412.
9. I.Guyon, S.Gunn, M.Nikravesh, and L.A.Zadeh, *Feature extraction: foundations and applications*, Series Studies in Fuzziness and Soft Computing, vol. 207, Springer, 2006.
10. F.Harary, *On the notion of balance of a signed graph*, Michigan Math. J. **2** (1953), no. 2, 143–146.
11. M.Al Hasan, V.Chaoji, S.Salem, and M.Zaki, *Link prediction using supervised learning*, SDM'06 workshop on Link Analysis, Counterterrorism and Security, 2006.
12. J.Kunegis, A.Lommatzsch, and C.Bauckhage, *The Slashdot Zoo: mining a social network with negative edges*, WWW 2009, pp. 741–750.
13. C.Lampe, E.Johnston, and P.Resnick, *Follow the reader: filtering comments on Slashdot*, CHI, 2007, pp. 1253–1262.
14. Jure Leskovec, Daniel P. Huttenlocher, and Jon M. Kleinberg, *Predicting positive and negative links in online social networks*, WWW, 2010, pp. 641–650.
15. Y. Li, W.Chen, Y.Wang, and Z.-L. Zhang, *Influence diffusion dynamics and influence maximization in social networks with friend and foe relationships*, WSDM 2013, pp. 657–666.
16. D.Liben-Nowell and J.Kleinberg, *The link prediction problem for social networks*, CIKM 2003, pp. 556–559.
17. D.LibenNowell and J.Kleinberg, *The link-prediction problem for social networks*, J. Am. Soc. Inf. Sci. Technol. **58** (2007), no. 7, 1019–1031.
18. P.Massa and P.Avesani, *Controversial users demand local trust metrics: An experimental study on epinions.com community*, AAAI, 2005, pp. 121–126.
19. C.McGeoch and C.Wang, *Experimental evaluation of an adiabiatic quantum system for combinatorial optimization*, CF 2013, pp. 23:1–23:11.
20. H.Neven, V.Denchev, G.Rose, and W.Macready, *Training a large scale classifier with the quantum adiabatic algorithm*, CoRR **abs/0912.0779** (2009).
21. M.E.J. Newman, *The structure and function of complex networks*, SIAM Review **45** (2003), no. 2, 167–256.
22. B.Sarwar, G.Karypis, J.Konstan, and J.Riedl, *Analysis of recommendation algorithms for e-commerce*, ACM Conf. on Electronic Commerce, 2000, pp. 158–167.
23. P.Symeonidis, E.Tiakas, and Y.Manolopoulos, *Transitive node similarity for link prediction in networks with positive and negative links*, RecSys 2010, pp. 183–190.
24. C.Wang, V.Satuluri, and S.Parthasarathy, *Local probabilistic models for link prediction*, ICDM 2007, 2007, pp. 322–331.