Link-Sign Prediction in Dynamic Signed Directed Networks

Quang-Vinh Dang, Claudia-Lavinia Ignat

Link-Sign Prediction in Dynamic Signed Directed Networks

Quang-Vinh Dang† and Claudia-Lavinia Ignat
Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France
Email: claudia.ignat@inria.fr

***Abstract*—Many real-world applications can be modeled as signed directed graphs wherein the links between nodes can have either positive or negative signs. Social networks can be modeled as signed directed graphs where positive/negative links represent trust/distrust relationships between users. In order to predict user behavior in social networks, several studies have addressed the link-sign prediction problem that predicts a link sign as positive or negative. However, the existing approaches do not take into account the time when the links were added which plays an important role in understanding the user relationships. Moreover, most of the existing approaches require the complete network information which is not realistic in modern social networks. Last but not least, these approaches are not adapted for dynamic networks and the link-sign prediction algorithms have to be reapplied each time the network changes.**

In this paper, we study the problem of link-sign prediction by combining random walks for graph sampling, Doc2Vec for node vectorization and Recurrent Neural Networks for prediction. The approach requires only local information and can be trained incrementally. Our experiments on the same datasets as state-of-the-art approaches show an improved prediction.

***Index Terms*—link-sign prediction, dynamic networks, recurrent neural networks, random walks, Doc2Vec**

I. Introduction

Online social network (OSN) services play an important role in our modern life. In literature, an OSN is usually represented as a graph $G =< V, E >$ where vertices are members of the OSN and edges are connections or relations between them [1]. The connections could be trust relations, friendships, memberships or following relations [2].

While most research works in the domain of social networks consider uniquely positive relationships [3], [4], many real-world multi-user scenarios can be represented as signed directed graphs [5] where links can be either positive or negative. In most popular signed directed social networks, the signs of links represent *trust* (positive) or *distrust* (negative) relationships between users [6]. Studies [7], [8] showed that trust relations play an important role in predicting user behaviors in e-commerce.

Several signed directed social networks are available for end-users. Examples include:

- **Epinions**[1] is a product-rating social network [9]. Users of Epinions can write reviews and rate products. They also can indicate that they *trust* (positive) or *distrust* (negative) other users.
- **Slashdot**[2] is a social-based technology news website [10]. Users of Slashdot can tag other users as *friend* (positive) or *foe* (negative).
- **Wikipedia** is a free online encyclopedia, created and edited by volunteers. During Request for Adminship (RfA) process[3], users can vote *for* (positive) or *against* (negative) other users for becoming administrators of certain Wikipedia pages [11].

Link-sign prediction problem is defined as follows. Given a signed directed network where signs of all edges are known, except for an edge, the task is to predict the sign of this edge by using information provided by the rest of the network [3]. This is a binary classification problem.

Link-sign prediction is not limited to signed directed social networks but can be easily applied to any applications which can be represented as a signed directed graph [5], [12]. Some examples of such real-world applications are given below:

- In an election, link-sign prediction is used for predicting for which candidate a particular voter will vote [13]. It can help candidates to better organize their campaigns.
- In social-based recommender systems [14], users tend to follow recommendations of their friends and disregard recommendations of people they do not trust [9]. If we can predict the missing relations between users, we can deliver a better recommendation service. The most important questions users of e-commerce systems consider when buying a new product are "What is the quality of this product? Is it as good as it claimed to be?". Many e-commerce systems allow users to refer to reviews and ratings of other users on the targeted products. However, further questions that arise are "Who are these reviewers? Should I trust them?". If we can predict trust / distrust relationships, we can recommend users reviews from reviewers that they trust.
- In collaborative systems such as Google Docs where multiple people can modify shared documents, users need to grant rights to other users. It is reasonable to argue that users tend to grant access rights to people they trust and deny access rights to people they do not trust. It is difficult for users to set up the access control in large-

† now at TMC Data Science, NL-5656AG Eindhoven, the Netherlands, Email: vinh.dang@tmc.nl

[1]http://epinions.com/

[2]https://slashdot.org/

[3]https://en.wikipedia.org/wiki/Wikipedia:Requests_for_adminship

scale settings. If we can predict the relationship between users, the task can be done automatically.

In this paper, we aim to solve the link-sign prediction problem by means of Recurrent Neural Network (RNN) where we consider a graph as a time-series data. To the best of our knowledge, it is the first time a graph is converted to a time-series data for link-sign prediction. The main idea underlying our approach is that we use Random Walk for sampling local neighborhood of nodes and then Doc2Vec to measure the similarity of nodes in term of their neighborhood. In the final step we use RNN to predict the sign of next link which will be made by nodes. The advantages of our algorithm are the following:

- It does not require full graph information which is very difficult to acquire in modern social networks.
- It can be trained incrementally, i.e. when the graph changes we only need to learn the new information. Modern social networks are known as very dynamic, i.e. their topologies change every second. It is too costly to execute from scratch an algorithm on any change of a social network.

II. Related Work

First of all, we should distinguish two related but different tasks: *link prediction* [15] and *link-sign prediction* [3]. The first task is to predict what link will be established in a graph, while the second task is to predict the sign of a link, given the existence of this link. The difference is visualized in Figure 1. In this study we focus on link-sign prediction problem.

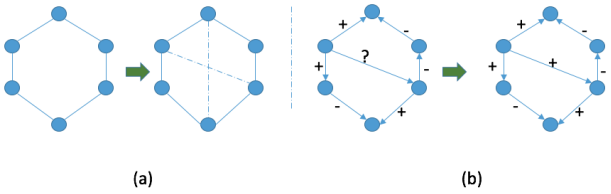


Fig. 1: Different types of link prediction tasks: (a) link prediction in unsigned undirected networks, (b) link sign prediction in signed directed networks

Many studies focused on link sign prediction in signed OSNs [4]. We limit our analysis to graph-based learning studies [1].

Graph-based link-sign prediction solutions take as input a graph with nodes and links between these nodes. The only reliable information is the topology of the graph, i.e. the directions and the signs of links. The solutions can not access other personal information such as historical trading information, gender or income of users.

Studies [16] claimed that the prediction task could be much easier if we can access the personal information. However, due to the raising concerns about privacy, it is suitable to avoid using this kind of information. Moreover, a graph-based prediction algorithm is general and it can be applied in an arbitrary graph.

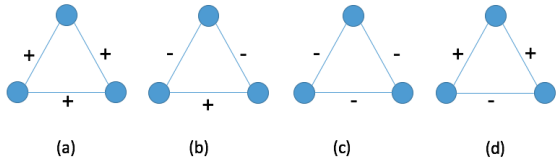


Fig. 2: Visualization of structural balance theory [3]. According to *structural balance theory*, triads *(a)* and *(b)* are balanced, while *(c)* and *(d)* are not. According to *weak balance theory* [17], triads *(a), (b)* and *(c)* are balanced, while *(d)* is not. Structural balance theory does not take the direction of edges into consideration.

For instance, Burke and Kraut [11] designed a solution to predict voting in Wikipedia RfA. However, the solution is limited to Wikipedia RfA as it uses election candidates and voters personal information.

One of the first graph-based link-sign prediction studies is the work of Guha et al. [9]. The authors developed a trust and distrust propagation framework by defining four atomic propagating operators which can be described in natural language as "if A trusts B and B trusts C so A trusts C", "if A trusts C and D and B trusts C so B trusts D", "if A trusts B and C trusts B so C trusts A" and "if A and B trust D and C trusts A so C trusts B". The prediction is calculated by propagating these atomic operators on user relations matrix. Theoretically the propagation could be recursively applied until all missing links are predicted. However, longer propagation distances lead to lower confidence of the prediction results.

Several algorithms rely on two social psychology rules: *structural balance theory* and *social status theory*. In short, *structural balance theory* states that, a triad which represents relations between three users tends to be balanced, i.e. it has an odd number of positive signs regardless the direction, as visualized in Figure 2. *Social status theory* claims that, if there is a positive edge from A to B, then A considers to have a *lower* social status than B, and if there is a negative edge from A to B, then A considers to have a *higher* social status than B. Using informal notions, we could express *social status theory* as, if $A \xrightarrow{+} B$, then $A < B$, and if $A \xrightarrow{-} B$, then $A > B$. If everyone agreed on a common social status, we could make a prediction as, if $A > B$ then $A \xrightarrow{-} B$ and $B \xrightarrow{+} A$[4]. Social status theory is visualized in Figure 3.

Based on these two theories, Leskovec et al. [3] trained logistic regression on a set of seven degree features calculated from triads of OSN graphs. The work of Leskovec et al. is extended by Chiang et al. [18] by using longer cycles such as quadrilaterals or pentagons. Hsieh et al. [17] presented low-rank matrix approximation with *weak balance theory*, which

[4]In this paper, we use the notation $A \xrightarrow{+} B$ to represent a positive edge from the vertex A to B, and $A \rightarrow B$ to represent an edge from A to B regardless the sign.

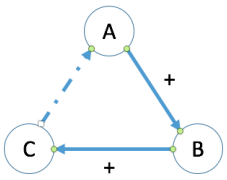


Fig. 3: Visualization of social status theory [3]. The sign of the dash line from C to A is inferred by their social status. Because $A \xrightarrow{+} B$ and $B \xrightarrow{+} C$, therefore we have $B > A$ and $C > B$, so $C > A$, hence social status theory predicts that the sign of line from C to A is negative.

extended the *structural balance theory* by considering a triad with all three negative edges as a balanced triad. A recent algorithm based on the two social theories is presented in [19] where the authors combine the two theories with users trustworthiness and predict how likely a user will trust other users. Zhou et al. [20] presented a parallel technique called PLSP to improve the training speed of classifier based on social psychology theories. PLSP achieves good performance, but requires a global network view and additional information such as user reviews.

Dubois et al. [6] combined path-probability trust inference with spring-embedding technique for trust / distrust prediction. The proposed algorithm performs well in dense networks, i.e. where network vertices form triangles, but its performance decreases dramatically in sparse networks. Additionally, the proposed algorithm requires a global view of the entire network.

Song et al. [4] argued that, (i) even structural balance theory and social status theory played an important role in existing research studies on link sign prediction, these theories are not very suitable in large-scale and extreme sparse networks, and (ii) a fully observed network is not always available in practice. Authors derived Bayesian node features based on partially observed networks and used a logistic regression classifier for link sign prediction. However, the obtained performance is weak compared to other recent studies.

Several studies used deep learning for feature selection of graphs before applying other "shallow" machine learning algorithms. Liu et al. [21] used Deep Belief Network (DBN) on node degree feature sets, Deng et al. [22] applied deep auto encoders for feature selection in social recommendations, and Li et al. [23] used Restricted Boltzmann Machine (RBM) for feature selection.

Inspired by recommender systems, You et al. [19] considered trust / distrust user relationships as recommendations, and applied matrix factorization for link sign prediction problem.

Wang et al. [12] presented an adaptation of matrix completion techniques designed to work with continuous numeric values to a problem of binary (positive, negative) values.

Khodadali and Jalili [24] focused on micro-structure, i.e. a group of three users with bidirectional links and their similarities for link-sign prediction.

The above described works have the following weaknesses that we aim to solve in our proposed approach:

- They do not take into account the time when the network links are established. In fact, the existing studies take a snapshot of a social network at a particular point of time then analyze the graph topology at this time. There is no difference between a link that has been established ten years ago and a link that has been added one second before the analysis. Studies in link prediction [16], [25] claimed that time factor plays an important role in predicting links that will be established in the networks. Time factor should be taken into consideration as well in link-sign prediction task.
- They usually require a fully observed network to operate [4].
  - Computation of the complete snapshot of modern social networks would be significant time and resource consuming. Moreover, modern social networks such as Facebook are very dynamic and the underlying topology will change before the information collection process finishes.
  - As many studies in social networks [26], [27] pointed out, users are mostly influenced by their friends rather than people they do not know. It is reasonable to argue that the activities of a user in Nepal should not make a lot of influence on another user in Mexico. We base our study on the core idea that, we need only local information of a link to predict its sign.
- They rely on a static snapshot of a graph, therefore they need to be re-executed if there is any change in the graph, such as a new added link. As graph change events occur very often in modern social networks, algorithms should perform the training on new data only.

III. Motivation

Our solution is based on two main observations.

A. Local influence

In order to predict the sign of a link we might only need the local information of this link. Many existing solutions in link-sign prediction rely on global information, i.e. these solutions need to know the information of the entire graph, or in other words they require a fully-observed network. As we discussed above, it is almost impossible today to retrieve the full topology of a network. Moreover, the task of link-sign prediction is defined as "predict sign of a single link" [3]. To predict the sign of a link from node A to node B, we do not need the information of the whole network but only local information, i.e. information about neighborhood of A and B as suggested by studies [4], [6].

On the other hand, many research works in both sociology and computer science [28]–[31] claimed a general principle in friendship or positive link prediction: people tend to make friends with other similar people. The issue is how should we

measure the similarity between people. In this study, we used Random Walk and Doc2Vec for this task.

B. Time matters

Secondly, we notice that the graph analysis problem can be treated as a time-series problem, because a graph is built by adding nodes and links one by one. To the best of our knowledge, it is the first time a time-series analysis technique is used in link-sign prediction.

To illustrate our observation, let us consider an example in voting prediction with two candidates A and B as visualized in Figure 4. Given a user u, we want to predict for which candidate u will vote. Suppose that u has the same number of positive links to supporters of both candidates as in Figure 4. In this situation, all approaches which are based on the two above social theories consider that u equally prefers the two candidates. However, if we know that the links to supporters of candidate A were established a long time ago while all the links to supporters of candidate B were recently established, we can safely assume that u supported A in the past but recently supports candidate B. Therefore it could make sense to predict that u will vote for B.

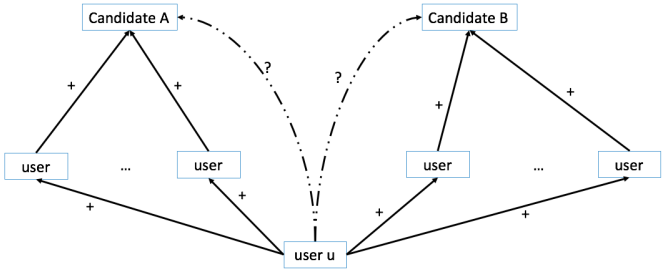


Fig. 4: An example of time factor in link-sign prediction. Without time information it will be difficult to predict the sign of links from u to A and B.

In many natural language processing (NLP) tasks, a document is represented by a graph between words. Therefore, NLP techniques can also be applied back into graph analysis problem, as proved by various recent studies [32]–[34].

IV. Background

Our algorithm relies on three techniques: random walk, Doc2Vec and Recurrent Neural Network. While the idea of random walk is quite simple and popular, in this section we will give a brief introduction about the other two techniques.

A. Doc2Vec

Doc2Vec [35] is an upgraded technique from Word2Vec [36]. As the name suggests, the main idea of Doc2Vec is to convert a document into a numerical vector in order to apply downstream machine learning algorithms on the output vector, such as to measure the similarity of different documents [37].

In recent years, different natural language processing (NLP) techniques, particularly Word2Vec algorithm, have been applied successfully in studying graph data [32], [34]. The difference between Doc2Vec and Word2Vec is that Doc2Vec also learns data *tags*, i.e. the link signs in our study. Therefore, we used Doc2Vec for the vectorization task in order to be able to measure the similarity between nodes.

The main idea is that we perform random walks from different nodes, then feed the output vectors of random walks into Doc2Vec in order to collect numerical vectors which represent the neighborhood of the nodes. Based on these vectors we can calculate the similarity between nodes by using simple metrics such as cosine similarity function.

B. Recurrent Neural Network

Recurrent Neural Network (RNN) [38, Chapter 10] has been introduced as a tool for time-series analysis for a long time [39]. Due to the recent rapid development in deep learning research, RNN is being considered as one of the best tools in analyzing time-series data [40].

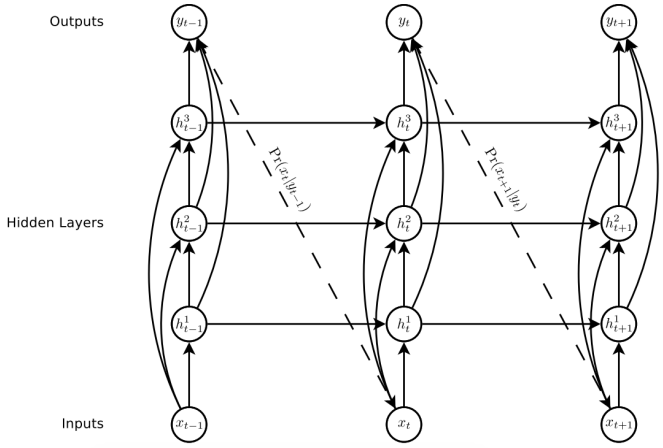


Fig. 5: Deep recurrent neural network [41].

The input of an RNN consists of a vector sequence $x = (x_1, x_2, ..x_T)$ that is processed through a stack of N recurrently connected hidden layers in order to compute the hidden vector sequences $h^n = (h_1^n, ..., h_T^n)$ and the output vector $y = (y_1, y_2, ..y_T)$. The output is used to predict the next input token, i.e. we use the output y_t to predict the distribution of the input x_{t+1}. While other neural network models learn only from the current data [38], RNN learns also from previous data to predict the future. This feature makes RNN be suitable to learn time-series data.

A detailed description of RNN can be found in several textbooks, such as [42] or [38].

1) Long-Short Term Memory: As of this writing, Long-Short Term Memory (LSTM) [43] is considered as one of the best RNN cells [38], [44]. Using LSTM means that we replace the activation function in RNN by a LSTM cell. A LSTM cell is visualized in Figure 6.

The calculation of LSTM is as follows [42]:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \quad (1)$$

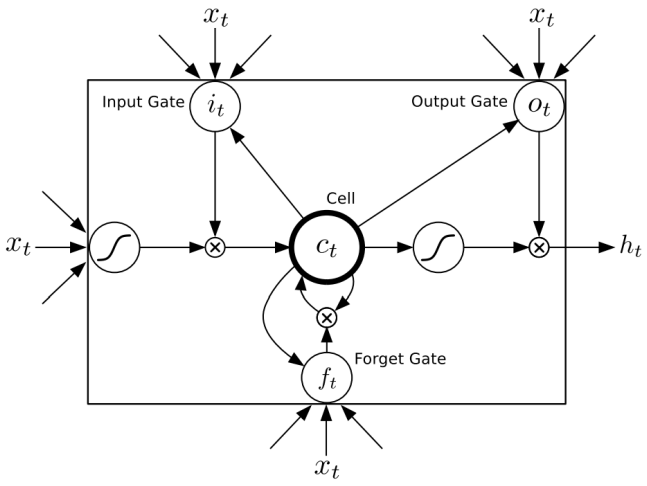


Fig. 6: A LSTM cell [41]

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \quad (2)$$

$$c_t = f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \quad (3)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_{t-1} + b_o) \quad (4)$$

$$h_t = o_t \tanh(c_t) \quad (5)$$

wherein:

- σ is the logistic sigmoid function.
- i, f, o, c are *input gate, forget gate, output gate* and *cell input* activation vectors respectively. All these vectors have the same size as the hidden vector h.
- W is the weight matrix. Its subscripts have the obvious meaning such as W_{hi} is the hidden-input gate matrix and W_{xo} is the input-output gate matrix.

A very important and special feature of LSTM is that it is equipped with a forget gate, which allows the model to *forget* outdated information.

In its original design, RNN-LSTM takes as input a series of numerical values and it can predict the next value in the series.

2) Stateful LSTM: Another important advantage of LSTM is that it can be trained incrementally. It means that if a RNN-LSTM model existed already and new data is received, we only need to train the model on the new data without training everything from scratch.

In fact, the core idea of stateful LSTM is very simple. After training a batch of examples, we keep the state of the model to continue training on the next example. However, this is only possible in RNN but not in other popular deep learning models due to the sequential training nature of RNN : RNN learns from the input which is a series of training examples one by one.

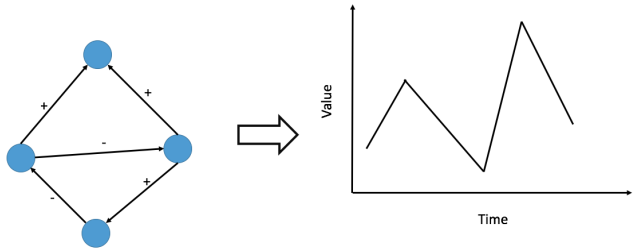


Fig. 7: Conversion from a graph to time-series data

V. Our Proposed Algorithm

In this section we present our proposed approach for trust/distrust prediction problem.

RNN-LSTM is a very powerful tool in analyzing time-series data. However, RNN-LSTM cannot analyze a graph directly. In its original design, the input of RNN-LSTM is a series of numeric values and it can predict the next value. A key problem is how can we transform a graph into a time-series data as visualized in Figure 7.

Our approach can be divided in two steps. In the first step, we perform a sampling process then measure the distance between nodes. In the second step, we feed the action list of a node as a time-series data into RNN-LSTM for prediction.

Intuitively, in the first step we measure the similarity of neighborhoods of two nodes whose connected link has an unknown sign. In the second step, based on a source node links created in the past, we predict the sign of the next link which will be established by using RNN.

A. Node distance by random walk & Doc2Vec

As discussed above, the first step in the sign prediction of the link from node A to node B is the distance measurement between A and B.

There are two main approaches to measure the node distance in a graph: *global-based* measurement and *local-based* measurement [45]. Global-based measurement means that the full observation of the graph is available and local-based measurement means that only local topology around current interest nodes is available.

We chose local-based measurement because of two reasons:

- As we discussed above, the full observation of real-world networks is not available.
- According to the studies [4], [6], [28], the decision of a user is mainly influenced by their direct friends, and the influence quickly diminishes as the distance to other users increases.

The distance measurement task can be further divided in two smaller tasks: graph sampling and vector mapping.

1) Sampling by Random Walk: Random Walk is used widely in graph sampling [32], [34], [46]–[50].

In order to measure the distance between nodes, we perform random walk in each node. The random walk we used is similar to [34], meaning that we follow the edges regardless

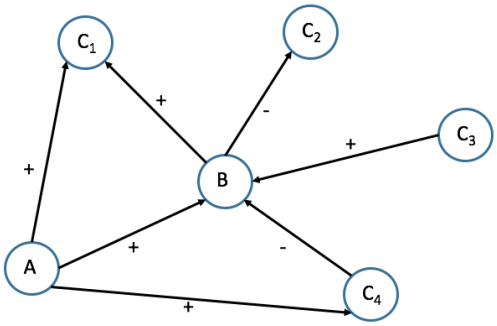


Fig. 8: Graph sampling by random walk.

the direction with the transitional probability. Let us consider an example of a walk as visualized in Figure 8. Suppose that the walk has just moved from node A to B. The unnormalized transitional probability is the following:

$$\alpha(A) = \frac{1}{p} \tag{6}$$

$$\alpha(x|\text{there is a link between } A \text{ and } x) = 1 \tag{7}$$

$$\alpha(x|\text{there is no link between } A \text{ and } x) = \frac{1}{q} \tag{8}$$

It means, there is an unnormalized probability of $\frac{1}{p}$ for the walk to immediately come back the previous node (node A). Similarly, the probability of $\frac{1}{q}$ is the probability that the walk further explores the network not known before. Different from Node2Vec [34], we keep the sign of visited links through the walk.

2) Vector mapping: Now for each node, we have a list of nodes as the result of random walk. Recent studies in graph embedding use natural language processing techniques [32], [34], [51].

Existing studies usually rely on Word2Vec [36] for mapping a series of nodes to a vector. However, because the links in our case are signed links, we used Doc2Vec [35] instead of Word2Vec for the task. The core idea is that we feed the sign of the links also with the input data. The task of graph vectorizing is described in Algorithm 1.

Algorithm 1: Graph Vectorizing

Data: a signed directed graph $G =< V, E >$
Result: a list of vectors, each vector representing a node in the graph.

```
// initialization
walks := an empty vector;
N = |V|;
// random walk
for i in 1:N do
    w := RandomWalk (V[i]);
    walks.append (w);
// vectorize
output := Doc2Vec (walks);
return output;
```

Algorithm 2: Sign Prediction

Data: output of the Graph Vectorizing task $graph\_vectors$.
Data: the graph $G =< V, E >$
Data: two nodes A and B whose link has unknown sign.
Result: predicting sign of the link $A \rightarrow B$.

```
// initialization
distance_vectors := an empty vector;
// distance calculation
for nb in neighbors(A) do
    if nb != B then
        v_A = graph_vectors(A);
        v_nb = graph_vectors(nb);
        d := cosine_distance(v_A, v_nb) * sign(A → nb);
        distance_vectors.append(d);
sort(distance_vectors, key = established_time);
// sign prediction
rnn := RNN_LSTM(distance_vectors);
raw_predict := rnn.predict(step = 1);
sign := ifelse (raw_predict > 0,1,-1);
return sign;
```

B. Recurrent Neural Networks for Link-Sign Prediction

After using Doc2Vec for transforming a node series to a vector, we have a list of vectors, each vector representing a node in the graph. The algorithm to predict the sign of the link $A \rightarrow B$ is described in Algorithm 2. Our link sign prediction algorithm does not use the full graph topology, but only local information. For predicting the link $A \rightarrow B$, Algorithm 2 needs only the vector representations ($graph\_vectors$) for A and its neighbor nodes. The distance vectors between A and its neighbor nodes are sorted based on edge creation time before they are fed into RNN_LSTM processing. The prediction step is specified as argument to the $predict$ function.

If the network topology changes, we just need to update the LSTM model as described in IV-B2 that makes our algorithm suitable for dynamic large-scale networks. The Algorithm 1 can be performed in off-line mode periodically such as every three months and update the LSTM model when the network changes.

VI. Experimental Results

A. Datasets

We performed link sign prediction on three popular signed directed OSNs datasets: Epinions, Slashdot and Wikipedia[5]. The datasets are provided by the authors of [52]. Several basic statistics of the three datasets are displayed in Table I.

The first and second row of the table display the number of nodes (vertices) and the number of edges in each dataset. The third row shows the fraction of existing edges over the

[5]The datasets are available at http://snap.stanford.edu

	Epinions	Slashdot	Wikipedia
# of nodes	119 217	82 140	7 118
# of edges	841 200	549 202	103 747
fraction of edges	$6e^{-5}$	$8e^{-5}$	$2e^{-3}$
+ edges (%)	85.0	77.4	78.8
− edges (%)	15.0	22.6	21.2
largest WCC (%)	99.1	100	100
average # of directed connection	590	327	418
# of triads	13 375 407	1 508 105	790 532
fraction of triads	$1.35e^{-10}$	$5.46e^{-11}$	$4.25e^{-9}$

TABLE I: Basic statistics of datasets. WCC stands for *weakly connected component*.

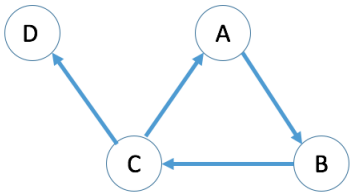


Fig. 9: Visualization of connected components. A, B and C form a strongly connected component, while A, B, C and D form a weakly connected component (WCC).

Size	Epinions	Slashdot	Wikipedia
100	16.15%	26.95%	6.92%
200	27.46%	46.24%	25.44%
300	37.82%	64.37%	44.02%
400	47.53%	75.06%	60.93%
500	55.54%	82.20%	70.87%
1000	81.25%	95.01%	93.44%

TABLE II: Cumulative distribution of primary neighborhood set size. For instance, on Epinions dataset, 16.15% of edges have the primary neighborhood set size smaller or equal to 100.

number of edges in a fully connected network with the same number of nodes. We can see that all OSN graphs are sparse.

The fourth and fifth row display the percentage of positive and negative edges per dataset. We can observe that a large portion of edges are positive, therefore a predicting model needs to provide a prediction with accuracy higher than the percentage of positive edges. For instance, the prediction model accuracy for Epinions dataset should be at least 85% which corresponds to the naive approach accuracy that predicts every output as positive.

The information "largest WCC" presented in the sixth row of Table I shows how much percentage of total edges belong to the largest weakly connected component in each dataset. WCC is visualized in Figure 9. The graphs are weakly connected, similarly to other OSN platforms. For instance, 99.91% of Facebook users are connected [53].

The seventh row of Table I presents the average size of primary neighborhood sets of all edges in each dataset. We define the *primary neighborhood set* of a link $A \rightarrow B$ as the set of all links with one vertex being either A or B and the other vertex being neither A nor B. The distribution of primary neighborhood set size is displayed in Figure 10. The histograms show that the distributions of primary neighborhood set size are similar between datasets.

The primary neighborhood set of a link shows the *richness* of the local information of this link. Given a particular link, if this set is too small, we simply do not have enough local information to make a prediction. In this case, we have to extend the training set by taking into account further neighbors rather than direct ones. The cumulative distribution of primary neighborhood set size is displayed in Table II. On the other hand, using a larger training set will increase the training time dramatically.

The eighth row "fraction of triads" of Table I presents the fraction of number of existing triads over total number of possible triads in each dataset. These fractions are extremely small, meaning that triads are not popular in the three datasets. Therefore, the algorithms relying on sociology rules [3], [17] might not perform well on these datasets.

B. Link-Sign Prediction on Static Graphs

In this section, we report on the results we obtained on link-sign prediction in static graphs, i.e. the graphs where all nodes and links are available and there is no removal or addition of nodes or links. We followed the leave-one-out validation setting of [3], i.e. we alternatively remove the sign of one link and try to predict this sign. The predicted result is then compared with the correct link sign.

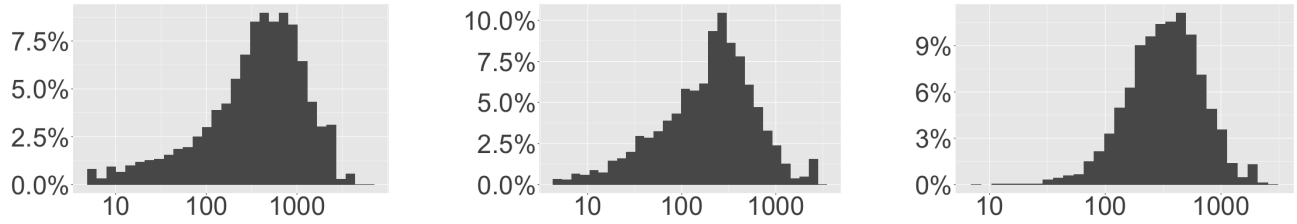
Due to the fact that most existing studies reported the performance of their algorithms using *accuracy* score, we keep using this metric for comparison purpose, even if the datasets are highly imbalanced. We also report F1-score but only for further references.

We used a RNN with one LSTM layer with 512 neurons. The number of roll back steps is five. We used the function $tanh$ as the activation function with the dropout ratio of 0.5 in both linear and recurrent connections [54]. We used the *mean squared error* as our loss function[6].

We present the accuracy scores on the three datasets in comparison with state-of-the-art solutions in Table III.

We present the F1-score of our RNN-LSTM approach and two other baseline algorithms ([9] and [3]) in Table IV. The F1-score for the baseline algorithms was computed based on our own implementation of these algorithms.

[6]The implementation and dataset are available at https://github.com/vinhqdang/link_sign_prediction



(a) Epinions (b) Slashdot (c) Wikipedia

Fig. 10: Distribution of size of primary neighborhood sets in three datasets (log scale)

	Epinions	Slashdot	Wikipedia
Degree features (2010) [3]	90.39	83.76	83.58
Triad features (2010) [52]	90.42	80.42	82.46
Degree + triad features (2010) [3], [52]	92.25	84.91	84.87
Longer cycles features (2011) [18]	90.64	83.83	84.04
Spring-based inference (2011) [6]	89	82	81
Low-rank modeling (2012) [17]	92.48	84.57	84.93
Weighted MF-LiSP (2013) [55]	89.0	80.2	80.0
LR (2013) [55]	91	82	81
PLSP (2014) [20]	96.2	89.6	89.1
BNTC + BNPC + Triad (2015) [4]	93.61	85.24	87.28
BNTK + BNPC + Triad (2015) [4]	93.13	85.65	87.37
ESS (2015) [56]	95.0	88.08	-
RNN-LSM	**96.31**	**91.66**	**89.76**

TABLE III: Link Sign Prediction Accuracies (%). The best accuracies are highlighted in bold. The values are extracted from corresponding papers. The ESS Wikipedia prediction metric was not reported in [56].

	Epinions	Slashdot	Wikipedia
Trust propagation [9]	0.892	0.885	0.882
Degree features [3]	0.889	0.893	0.887
RNN-LSTM	0.911	0.905	0.896

TABLE IV: F1-score of different algorithms on static graphs. The F1-scores for the two baseline algorithms [3], [9] are based on our own implementation.

C. Link-Sign Prediction on Dynamic Graphs

In this section, we compare link-sign prediction in dynamic graphs, i.e. graphs where nodes are fixed but links are added over time. Because there is no existing study on this kind of graphs, we re-implemented the two algorithms described in [9] and [3] as baseline algorithms.

In this experiment, we first established the network by adding links one by one. When the number of links reaches $1,000$, the prediction is started. We fed the next link into each algorithm, i.e. our algorithm and those presented in [9] and [3]. After all three algorithms made the prediction, we added this new link into the training set and fed the next link.

Running times of the first 200 predictions of the three algorithms are displayed in Figure 11. The running time of the trust propagation algorithm [9] is constant regardless of the size of the dataset, while the running time of the logistic regression based on sociology rules [3] increases almost linear with the graph size. These observations can be explained by the fact that the trust propagation is a simple rule-based approach while the sociology-based algorithm relies on logistic regression which needs more time for training new data. On the other hand, the RNN-LSTM takes a long period of time for the first prediction, but the running time is reduced dramatically for next predictions as we do not need to train again the data.

Similarly, we displayed the accuracy score of the three algorithms on dynamic graphs in Figure 12. The accuracy is calculated after the first 100 predictions and after each prediction on a new link. Again, we could see that the performance of the trust propagation [9] does not depend much on the size of dataset, while the logistic regression based approach [3] performs better when more data are available. The RNN-LSTM also achieves higher score with more data but the influence of new data is less than in the algorithm described in [3].

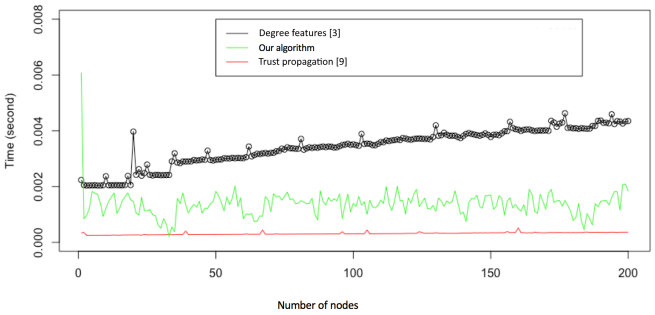


Fig. 11: Running time of different algorithms on dynamic graphs.

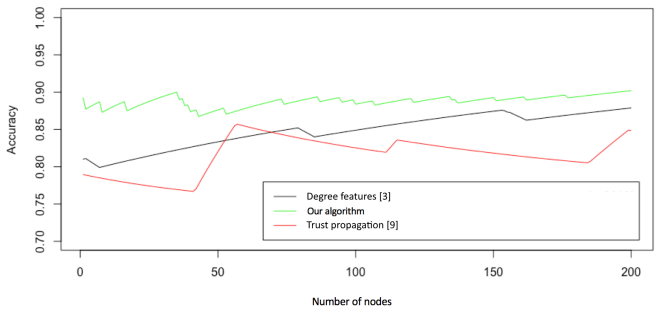


Fig. 12: Accuracies on dynamic graphs.

VII. Discussion

Several research studies addressed the link sign prediction in the last decade. However, many proposed algorithms share the same shortcoming that they fail when applied on sparse networks. For instance, in order to predict the sign of $A \rightarrow B$, structural balance theory [3] requires that A and B need to form a triangle with another vertex C, which is not always available in real-world social networks [4]. Table I showed that, the fraction of existing triads over the total number of possible triads that can be formed in OSNs is very small, therefore structural balance theory and social status theory are not very suitable.

Our algorithm can be seen as predicting the *step* a user makes on the graph. A step is a link established from a user to another user. The core idea is that, if we have the log of steps made by a user in the past, we can predict the next step of this user.

As we consider a graph as a time-series data, the algorithm has the limitation that the prediction can be made only for the next established link from a node. In other words, if we have a log of 10 links made by a user, we can only predict the sign of the 11^{th} link made by this user and we cannot skip to predict the 12^{th} link for instance. In order to do that, we have to make the prediction of the 11^{th} link, consider it as a correct information, add this prediction into the training set and make the prediction of the 12^{th} link. However, this limit is shared by other algorithms [9].

The study can be extended in several directions. Firstly, trust relations between users can be used to improve the quality of recommendation systems [7]. Secondly, we might integrate other domain-specific studies to enhance link prediction quality in a particular domain. For instance, research works in quality assessment of Wikipedia [37], [57]–[59] can enhance the link prediction algorithms relying uniquely on network topology: users tend to trust other users who contribute to high quality works. Moreover, binary-level trust can be replaced by numerical value trust by considering user past behavior [60].

VIII. Conclusion

In this paper, we combine several state-of-the-art techniques in natural language processing (Doc2Vec) and deep learning (RNN-LSTM) with the traditional random walk graph sampling for link-sign prediction. Experiments showed that our algorithm achieves better performance metrics in both static and dynamic graphs in comparison with state-of-the-art link-sign prediction algorithms. Furthermore, the running time of our algorithm is better in large-scale graphs.

References

[1] W. Jiang, G. Wang, M. Z. A. Bhuiyan, and J. Wu, "Understanding graph-based trust evaluation in online social networks: Methodologies and challenges," *ACM Comput. Surv.*, vol. 49, no. 1, p. 10, 2016.

[2] I. King, M. R. Lyu, and H. Ma, "Introduction to social recommendation," in *WWW tutorials*. ACM, 2010, pp. 1355–1356.

[3] J. Leskovec, D. P. Huttenlocher, and J. M. Kleinberg, "Predicting positive and negative links in online social networks," in *WWW*. ACM, 2010, pp. 641–650.

[4] D. Song and D. A. Meyer, "Link sign prediction and ranking in signed directed social networks," *Social Netw. Analys. Mining*, vol. 5, no. 1, pp. 52:1–52:14, 2015.

[5] J. Tang, Y. Chang, C. Aggarwal, and H. Liu, "A survey of signed network mining in social media," *ACM Comput. Surv.*, vol. 49, no. 3, pp. 42:1–42:37, 2016.

[6] T. DuBois, J. Golbeck, and A. Srinivasan, "Predicting trust and distrust in social networks," in *SocialCom/PASSAT*. IEEE, 2011, pp. 418–424.

[7] Q.-V. Dang and C.-L. Ignat, "dTrust: A simple deep learning approach for social recommendation," in *CIC*. IEEE, 2017, pp. 209–218.

[8] Q.-V. Dang, "Trust assessment in large-scale collaborative systems. (évaluation de la confiance dans la collaboration à large échelle)," Ph.D. dissertation, University of Lorraine, Nancy, France, 2018.

[9] R. V. Guha, R. Kumar, P. Raghavan, and A. Tomkins, "Propagation of trust and distrust," in *WWW*. ACM, 2004, pp. 403–412.

[10] J. Kunegis, A. Lommatzsch, and C. Bauckhage, "The slashdot zoo: mining a social network with negative edges," in *WWW*. ACM, 2009, pp. 741–750.

[11] M. Burke and R. E. Kraut, "Mopping up: modeling wikipedia promotion decisions," in *CSCW*. ACM, 2008, pp. 27–36.

[12] J. Wang, J. Shen, P. Li, and H. Xu, "Online matrix completion for signed link prediction," in *WSDM*. ACM, 2017, pp. 475–484.

[13] P. Rozenshtein, N. Tatti, and A. Gionis, "Finding dynamic dense subgraphs," *ACM Trans. Knowl. Discov. Data*, vol. 11, no. 3, pp. 27:1–27:30, Mar. 2017. [Online]. Available: http://doi.acm.org/10.1145/3046791

[14] J. He and W. W. Chu, "A social network-based recommender system (SNRS)," in *Data Mining for Social Network Data*, ser. Annals of Information Systems. Springer, 2010, vol. 12, pp. 47–74.

[15] D. Liben-Nowell and J. M. Kleinberg, "The link-prediction problem for social networks," *JASIST*, 2007.

[16] J. Tang, H. Gao, H. Liu, and A. D. Sarma, "eTrust: understanding trust evolution in an online world," in *KDD*. ACM, 2012, pp. 253–261.

[17] C. Hsieh, K. Chiang, and I. S. Dhillon, "Low rank modeling of signed networks," in *KDD*. ACM, 2012, pp. 507–515.

[18] K. Chiang, N. Natarajan, A. Tewari, and I. S. Dhillon, "Exploiting longer cycles for link prediction in signed networks," in *CIKM*. ACM, 2011, pp. 1157–1162.

[19] Q. You, O. Wu, G. Luo, and W. Hu, "A probabilistic matrix factorization method for link sign prediction in social networks," in *MLDM*, ser. Lecture Notes in Computer Science, vol. 9729. Springer, 2016, pp. 415–420.

[20] J. Zhou, L. Han, Y. Yao, X. Zeng, and F. Xu, "A parallel approach to link sign prediction in large-scale online social networks," *Comput. J.*, 2014.

[21] F. Liu, B. Liu, C. Sun, M. Liu, and X. Wang, "Deep learning approaches for link prediction in social network services," in *ICONIP*, 2013, pp. 425–432.

[22] S. Deng, L. Huang, G. Xu, X. Wu, and Z. Wu, "On deep learning for trust-aware recommendations in social networks," *IEEE TNNLS*, vol. PP, no. 99, pp. 1–14, Feb 2016.

[23] X. Li, N. Du, H. Li, K. Li, J. Gao, and A. Zhang, "A deep learning approach to link prediction in dynamic networks," in *SDM*. SIAM, 2014, pp. 289–297.

[24] A. Khodadadi and M. Jalili, "Sign prediction in social networks based on tendency rate of equivalent micro-structures," *Neurocomputing*, p. 10, 2017.

[25] X. Liu, A. Datta, and E.-P. Lim, *Computational Trust Models and Machine Learning*. Chapman and Hall/CRC, 2014.

[26] X. Chen, C. A. Wang, and X. M. Zhang, "All online friends are not created equal: Discovering influence structure in online social networks," in *PACIS*, 2013, p. 56.

[27] H. Guo, P. Pathak, and H. K. Cheng, "Estimating social influences from social networking sites - articulated friendships versus communication interactions," *Decision Sciences*, vol. 46, no. 1, pp. 135–163, 2015.

[28] M. Cygan, M. Pilipczuk, M. Pilipczuk, and J. O. Wojtaszczyk, "Sitting closer to friends than enemies, revisited," *Theory Comput. Syst.*, vol. 56, no. 2, pp. 394–405, 2015.

[29] G. A. Klein, *Sources of power: How people make decisions*. MIT press, 1999.

[30] L. Lu and T. Zhou, "Link prediction in complex networks: A survey," *Physica A: Statistical Mechanics and its Applications*, 2011.

[31] T. Yuan, J. Cheng, X. Zhang, Q. Liu, and H. Lu, "How friends affect user behaviors? an exploration of social relation analysis for recommendation," *Knowl.-Based Syst.*, vol. 88, pp. 70–84, 2015.

[32] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: online learning of social representations," in *KDD*. ACM, 2014, pp. 701–710.

[33] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: large-scale information network embedding," in *WWW*. ACM, 2015, pp. 1067–1077.

[34] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *KDD*. ACM, 2016, pp. 855–864.

[35] Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents," in *ICML*, ser. JMLR Workshop and Conference Proceedings, vol. 32. JMLR.org, 2014, pp. 1188–1196.

[36] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013, pp. 3111–3119.

[37] Q.-V. Dang and C.-L. Ignat, "Quality assessment of wikipedia articles without feature engineering," in *JCDL*. ACM, 2016, pp. 27–30.

[38] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT Press, 2016.

[39] J. T. Connor, D. R. Martin, and L. E. Atlas, "Recurrent neural networks and robust time series prediction," *IEEE Trans. Neural Networks*, vol. 5, no. 2, pp. 240–254, 1994.

[40] J. C. B. Gamboa, "Deep learning for time-series analysis," *arXiv preprint arXiv:1701.01887*, p. 13, 2017.

[54] Y. Gal and Z. Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," in *NIPS*, 2016, pp. 1019–1027.

[41] A. Graves, "Generating sequences with recurrent neural networks," *CoRR*, vol. abs/1308.0850, 2013.

[42] ——, *Supervised Sequence Labelling with Recurrent Neural Networks*, ser. Studies in Computational Intelligence. Springer, 2012, vol. 385.

[43] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[44] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *CoRR*, vol. abs/1412.3555, 2014.

[45] S. Even, *Graph algorithms*, 2nd ed. Cambridge University Press, 2011.

[46] J. Leskovec and C. Faloutsos, "Sampling from large graphs," in *KDD*. ACM, 2006, pp. 631–636.

[47] S. V. N. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgwardt, "Graph kernels," *Journal of Machine Learning Research*, vol. 11, pp. 1201–1242, 2010.

[48] B. F. Ribeiro and D. F. Towsley, "Estimating and sampling graphs with multidimensional random walks," in *Internet Measurement Conference*. ACM, 2010, pp. 390–403.

[49] R. Li, J. X. Yu, L. Qin, R. Mao, and T. Jin, "On random walk based graph sampling," in *ICDE*. IEEE Computer Society, 2015, pp. 927–938.

[50] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017, p. 14.

[51] P. Ristoski and H. Paulheim, "Rdf2vec: RDF graph embeddings for data mining," in *International Semantic Web Conference (1)*, ser. Lecture Notes in Computer Science, vol. 9981, 2016, pp. 498–514.

[52] J. Leskovec, D. P. Huttenlocher, and J. M. Kleinberg, "Signed networks in social media," in *CHI*. ACM, 2010, pp. 1361–1370.

[53] J. Ugander, B. Karrer, L. Backstrom, and C. Marlow, "The anatomy of the facebook social graph," *CoRR*, vol. abs/1111.4503, 2011.

[55] P. Agrawal, V. K. Garg, and R. Narayanam, "Link label prediction in signed social networks," in *IJCAI*. IJCAI/AAAI, 2013, pp. 2591–2597.

[56] G.-N. Wang, H. Gao, L. Chen, D. N. Mensah, and Y. Fu, "Predicting positive and negative relationships in large social networks," *PloS one*, vol. 10, no. 6, 2015.

[57] Q.-V. Dang and C.-L. Ignat, "Quality assessment of wikipedia articles: a deep learning approach," *SIGWEB Newsletter*, vol. 2016, no. Autumn, pp. 5:1–5:6, 2016.

[58] ——, "Measuring quality of collaboratively edited documents: The case of wikipedia," in *CIC*. IEEE Computer Society, 2016, pp. 266–275.

[59] ——, "An end-to-end learning solution for assessing the quality of wikipedia articles," in *OpenSym*. ACM, 2017, pp. 4:1–4:10.

[60] ——, "Computational trust model for repeated trust games," in *Trustcom/BigDataSE/ISPA*. IEEE, 2016, pp. 34–41.