Machine Learning On The Cloud

DAHMRI Khaled[1], KOTZINOS Dimitrios[2]
ETIS - ENSEA / Universit de Cergy-Pontoise / CNRS - France
khaled.dahmri@etu.u-cergy.fr[1], Dimitrios.Kotzinos@u-cergy.fr[2]

***Abstract*—Link prediction task has attracted the attention of many researchers for the past few years, and there have been many works to address this task through different approaches. The goal of link prediction is to predict connections that are most likely to appear or to disappear in the future, based on previous links observation, and network similarity measures that are commonly used in different approaches. However, most of them neglect the evaluation of the network as the time goes by. In this paper, we take into consideration the temporal information (times series) of the network beside a set of network similarity measures and a forecasting model, our method based on a supervised approach for implementation, we use Support Vector Machine (SVM), and logistic regression (LR) which are implemented in Spark-ML [20]. Our experiments show that when temporal information is considered in a network we get a better result, and significantly improve the prediction accuracy comparing to the other approaches.**

I. Introduction

Nowadays social networks are dynamic and constantly in flux with new vertices and edges that are being added or deleted over the time. These networks can be visualized as graphs, where a vertex corresponds to a person in some group and an edge represents some form of association between the corresponding persons, that are usually driven by mutual interests that are intrinsic to a group [1]. Understanding and modeling the dynamics of social networks is considered a complex problem and this is due to a large number of variable parameters. But, what is consider as lass complex problem is understanding the relationship between the nodes.

The problem we address in this paper is to predict the possibility of a future association between two nodes, that there is no association between them in the current state of the graph, based on some similarity measure, and a forecasting model. This problem is commonly known as the Link Prediction task.The link prediction task deals with predicting and forecasting new links which are likely to emerge in network in the future, given the network at the current time. It has a wide range of applications including recommender systems, spam mail classification, identifying domain experts in various research areas...etc. [2].

Link prediction approaches commonly rely on measuring topological similarity between non-connected pair of nodes [3], in the network at a time t in order to predict if a link will occur at a time $t+1$. These metrics provide scores to each pair of nodes which are then used to perform the prediction task either by similarity-based approaches or learning-based approaches. As shown in Figure 1.

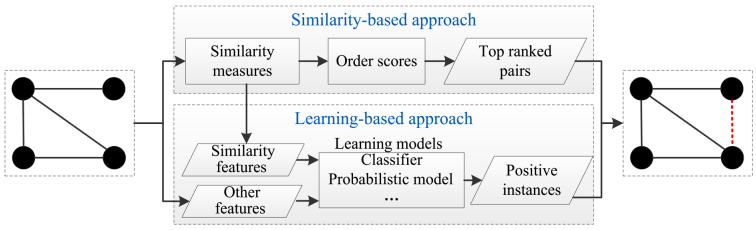

Fig. 1: Link prediction approaches

Similarity-based approaches is to compute the similarities on non-connected pairs of nodes in a social network. Every potential pair of node (x, y) would be assigned a score, where higher score means higher probability that x and y will be linked in the future, and vice versa. Then a ranked list in decreasing order of scores is obtained and links at the top of list are most likely to appear. Learning-based approaches is treating the link prediction problem as a binary classification task, where each pair of nodes corresponds to an instance with features describing nodes, and the class label. If the pair of nodes are connected then they will be labeled as positive, otherwise they will be labeled as negative. For the learning-based approaches, the features consist of two parts: one is the similarity features from the similarity-based approaches, another is the features derived from the social network.[4]

The topological similarity measures are good indicator of future graph connection, but they are less accurate, so

in order to improve the accuracy we constructed a time series for each pair of nodes from previous observations to capture their behavior through the time, and then a forecasting model is applied beside the topological similarity measures to enhance the prediction's accuracy. In this paper we reimplement RPM [3], it is a supervised link prediction method, and for our implementation we use two machine learning methods (Support Vector Machine, and Logistic Regression), applied using Spark MLlib machine learning library [20], and our experiments were conducted with Travian Network Datasets (Trades and Messages dataset) massively multiplayer online games.[5]

The remainder of the paper is organized as follows: section 2 is a survey of related work, section 3 is a description of our approach, section 4 implementation and results of the approach, and finally we end up with a conclusion in section 5.

II. Related work

There are many works focused on solving special link prediction problems, which can be divided into six categories: temporal link prediction, active/unactive link prediction, link prediction in bipartite networks, link prediction in heterogeneous networks, unfollow or disappearing link prediction, and link prediction scalability [4], in this paper we are interested in temporal link prediction category.

Traditional approaches for link prediction use a chosen topological similarity metric on non-connected pairs of nodes of the network at a present time to obtain a score that is going to be used by an unsupervised or a supervised method for link prediction. These approaches rely only on analyzing the network at the current moment without considering when the links were created in the network [6], and recently some works have appeared and they considered the evaluation of topological metrics in the network over time, and these works presented better results [6], [3], [7], [8], [9], [10].

In [3] presented a new supervised link prediction method RPM (Rate Prediction Model), that uses time series to predict the rate of link formation by identifying the most active individuals in the social network, by measuring time-sliced network structure and finds individuals whose influence is rapidly rising. RPM achieves statistically significant improvements over related link prediction methods. In [6] Created time series for each non-connected pair of nodes by calculating their similarity scores at different times in the past, then they designed a forecasting model based on time series to obtain final link prediction scores of the pairs. Their work presents satisfying results, but shows limitations regarding the number of networks used in the experiments and their domains. In [10] presented a graph-based link prediction method that incorporate the temporal information contained in evolving networks. The results show that time-stamps of past interactions significantly improve the prediction accuracy of new and recurrent links over the proposed methods, and They found that the performance can be improved by either time-based weighting of edges or weighting of edges according to the connecting strength. In [8] They presented a new time-aware feature, known as time score, that captures the important aspects of time stamps of interactions and the temporality of the link strengths. And they analyzed the effectiveness of time score with different parameter settings for different network data sets. Their results showed a significant improvement in predicting future. In [9] They used moving averages and percentage of change of baseline metrics, and they also used the Dynamic Bayesian Network (DBN), to mine the actual relationships between the metrics and forming links at each time step. The DBN was able to predict relationships between the metrics and forming links at each time step, as it evolved over time, and they achieved high prediction accuracies. In [7] proposed a time-series link prediction approach that takes into consideration temporal evolutions of link occurrences to predict link occurrence probabilities at a particular time, and they have demonstrated that time-series models of single-link occurrences achieve comparable link prediction performance with commonly used static graph link prediction algorithms, and they showed that a combination of static graph link prediction algorithms and time-series models produced better results comparing with static graph link prediction methods. In [1] They studied link prediction as a supervised learning task, and they identified a set of features that are key to the performance under the supervised learning setup. They have shown that the link prediction problem can be handled effectively by modeling it as a classification problem, and they have shown that most of the popular classification models can solve the problem with an acceptable accuracy, but the SVM beats all of them in many performance metrics. In [11] They proposed a deep learning framework called conditional temporal restricted Boltzmann machine (ctRBM), which predicts links based on individual transition variance as well as influence introduced by local neighbors, the model is robust to noise and have the exponential capability to capture nonlinear variance. In [12] They proposed a nonparametric link prediction algorithm for a sequence of graph snapshots over time. The model predicts links based on the features of its

endpoints, as well as those of the local neighborhood around the endpoints. In [13] They explored several matrix and tensor-based approaches to solving the link prediction task. For the matrix methods, the results show that using a temporal model to combine multiple time slices into a single training matrix is superior to simple summation of all temporal data, also show that the tensor-based methods are competitive with the matrix-based methods in terms of link prediction performance.

III. Proposed approach

In this section we will give an overview about the implemented approach, that we will see with more details in implementation and results section, and we will see also in this section the key elements that are used in this approach like : Time series, similarity topological metrics...etc.

In this paper we handle the link prediction problem as a supervised classification task, where each instance of data corresponds to a set of features of a pair of nodes in the social network with a label as a class. The idea is to build time series for each pair of nodes of the network using similarity topological metrics and a forecasting model, and then we assign a class for each pair of nodes, class 1 for pair of nodes that are linked, class 0 otherwise, and then we implement our classifier for training and testing, and in this paper we used two classifiers: Support Vector Machine (SVM), and Logistic Regression (LR) using Spark MLlib machine learning library [20].

A. Time series

In order to build the time series, the network G observed at time t must be split into several time-sliced snapshots, that is, states of the network at different times in the past. Afterwards, a window of prediction is defined, representing how further in the future we want to make the prediction. Then, consecutive snapshots are grouped in small sets called frames. Frames contain as many snapshots as the length of the window of prediction. These frames compose what is called Framed Time-Sliced Network Structure (S) [6].

Let Gt be the graph representation of a network at time t . Let $[G_1, G_2, ..., G_T]$ be the frame formed by the union of the graphs from time 1 to T. Let n be the number of periods (frames) in the series. And let w be the window of prediction. Formally, S can be defined as:

$$S = [G_1, ..., G_w], [G_{w+1}, ..., G_{2w}], ...[G_{(n-1)w+1}, ..., G_{nw}]$$

For instance, suppose that we observed a network from day 1 to day 9, and our aim is to predict links that will appear at day 10. In this example, the forecast horizon (window of prediction) is one day. Our aim here is to model how the networks evolve every day in order to predict what will happen in the forecast horizon [3]. In this paper we observe our network for one month, splitted into 30 snapshots, and each snapshot corresponds to one day.

B. Features set

1) Network Similarity Metrics:

In this paper we use the following network similarity metrics, that are commonly used, in link prediction problem:

a) Common Neighbors (CN) [14]:

Computes the number of nodes with direct relationships with both members of the node pair. It is defined as:

$$CN(x, y) = |\Gamma(x) \cap \Gamma(y)|$$

b) Jaccard Coefficient (JC) [15]:

Computes the probability of two nodes be linked by calculating the ratio between the number of neighbors they share and the total number of distinct neighbors they have. It is defined as:

$$JC(x, y) = \frac{\|\Gamma(x) \cap \Gamma(y)\|}{\|\Gamma(x) \cup \Gamma(y)\|}$$

c) Adamic Adar (AA) [3], [16] :

Similar to JC, assigns a higher importance to the common neighbors that have fewer total neighbors. Hence, it measures exclusivity between a common neighbor and the evaluated pair of nodes. It is defined as:

$$AA(x, y) = \sum_{z \in \|\Gamma(x) \cap \Gamma(y)\|} \frac{1}{\log(\Gamma(z))}$$

d) Preferential Attachment (PA) [17], [14] :

The Preferential Attachment measure assumes that the probability of a future link between two nodes is proportional to their degrees $|\Gamma(x)|$. It is defined as:

$$PA(x, y) = |\Gamma(x) \times \Gamma(y)|$$

e) Shortest Path (SP):

In this paper we use shortest paths dijkstra, that calculates shortest path lengths for given vertices in a graph.

$$SP(x, y) = Spaths_{dijkstra}(x, y)$$

f) Betweenness Centrality (CB) [18] :

Computes the number of times a node lies on a shortest path between any two nodes of the network other than itself:

$$C_B(v_i) = \sum_{i \neq j \neq k} \frac{|spaths(v_j, v_k | v_i)|}{|spaths(v_j, v_k)|}$$

2) Forecasting Model:
There exist many forecasting models that we can use, like Average (Av) [19], Moving Average (MA) [19], Exponential Smoothing [19]...etc., but in this paper we use Weighted Moving Average [3].

a) Weighted Moving Average (WMV):
Let $F_t(t = 1, ..., T)$ be a time series with T observations with A_t defined as the observation at time t and F_{t+1} the time series forecast at time $t + 1$.
This model is similar to moving average [19] method, but allows one period to be emphasized over others. The sum of weights must add to 100% or 1.00, it is defined as:

$$F_{t+1} = \sum C_t A_t$$

This model was tested in [3] with other models on Travain datasets, and it shows that Weighted Moving Average presents the best performance model as it is shown in Figure 2, with $n = 3$ that represents the most recent observed values, and parameters $C1$,$C2$ and $C3$ set to 0.2,0.3, and 0.5 respectively

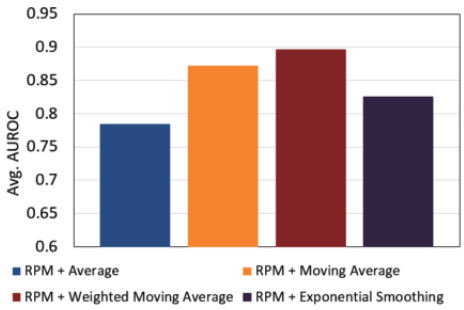

Fig. 2: Performance of RPM using different forecasting models on Travian DataSet

C. DataSets

In this paper we use two datasets (Trade and Messages) from Travian Network Datasets [5]. Travian is a popular browser-based real-time strategy game with more than 5 million players. Players form alliances of up to 60 members under a leader or a leadership team. Each game cycle lasts a fixed period (a few months), and the data used in this paper is one-month period splitted into 30 days (snapshots).Figure 3 indicates that Travian is a highly dynamic dataset, with over 90% of the edges changing between snapshots[3].

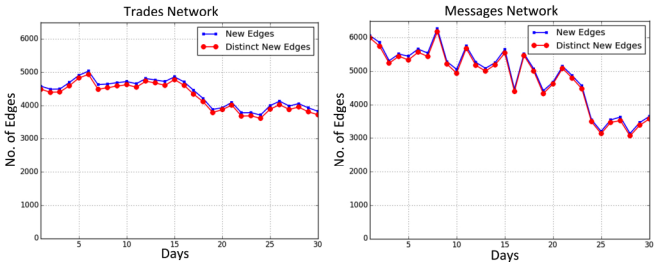

Fig. 3: Dynamics of the Travian network (trades: left and messages: right). The blue line shows the new edges added, and the red line shows edges that did not exist in the previous snapshot.

The data is publicly available : http://ial.eecs.ucf.edu/travian.php . Table 1 gives dataset summary for each dataset:

Data	Travian (Messages)	Travian (Trades)
No. of nodes	2,809	2,466
Link (Class 1)	44,956	87,418
No Link (Class 0)	7,845,525	5,993,738
No. of snapshots	30	30

TABLE I: Dataset Summary

IV. Implementation and Results

Our implementation is conducted as follows:

A. Building Time series for the network

In this paper, the dataset that we use is a network for one month period, and it is already splitted into 30 snapshots, where each snapshot represents one day.

For each snapshot we apply steps B, C and D:

B. Creating similarity metrics matrix

In this step we create a set of Matrix, and each matrix corresponds to a Similarity Metric between the all possible pair of nodes (vertices) that we described earlier, the seize of matrix is equal to the seize of vertices of the network, and it varies from snapshot to another as our datasets are dynamics, so some vertices might be deleted or added as the time goes by.

C. Creating forecasting vector

In this step we create a vector of a forecasting model for each node in the network.

D. Creating Features Vector

In this step we build a vector for each pair of nodes with a set of features and label class as described previously, class 1 for pair of nodes which are liked to each other , and class 0 otherwise, and these vectors will be used to construct our training and testing datasets. Figure 4 shows an instance of a vector of features between two nodes x and y.

Features	F0	F1	F2	F3	F4	F5	F6	class
Values	CN(x, y)	JC(x, y)	AA(x, y)	P A(x, y)	SP (x, y)	CB(x)	WMV(x)	0/1

Fig. 4: Instance of a vector of features with a label

E. Preparing training and testing datasets

Once the vectors of features are created of our data, the current data now is transformed from 30 snapshots of a network where each snapshot corresponds to the state of the network at a time t, to 30 files of features where each file corresponds to a set of vectors of features of all possible pair of nodes of the corresponding snapshot. In this step we proceed with splitting our data that contains the features into training and testing dataset, where the training dataset represents 80% of our data which represents the first 24 days (24 snapshots), and the testing dataset represents 20% of our data which represents the last 6 days (6 snapshots). An important challenge in this step is handling extreme class skewness. The number of possible links is quadratic in the number of existing links in a network, resulting in large class skewness. The most commonly used technique for this problem is to balance the training dataset by using a small subset of the over-represented class [3], what we did in this paper is reducing the class 0 in our training set in a way that the size of class 0 (random selection) is twice the size of class 1.

F. Implementing the classifiers

After preparing the training and testing datasets, we implement our classifiers: SVM and LR with Apache Spark [20], running the classifiers with Spark enables us to train and to test the classifiers with the full data distribution.

We choose SVM for our implementation based on previous studies and works [3], [6], [1] that have proven the consistency and effectiveness of the SVM in link prediction comparing to of the well-known classification algorithms (decision tree, k-NN, multilayer perceptron, RBF network), and as Support Vector Machine (SVM) algorithm is a linear method of Binary classification, we wanted to impliment another linear algorithm of Binary classification like Logistic Regression (LR) besides the SVM and compare the two of them.

In this paper our results are measured with *under the ROC curve (AUROC)*. The results are summarized in Table 2 for all methods reported in [3] with the Travian datasets (Trades and Messages), and as it can be seen we managed to get better results with both methods (SVM and LR) which are approximately identical. We get better results comparing to the original paper RPM [3] that we reimplemented, which is a supervised link prediction method, and the only difference that we made for our implementation is balancing the dataset as described earlier to bias our classifiers against the strong presentation of the class 0, in contrast to their training set which is highly imbalanced and that might causes the misleading classification, even though, the results they get are better than the existing methods, sa it can be seen it the results table.

Comparing to Supervised-MA [6] Supervised learner with Moving Average predictor, implemented SVM from the WEKA environment [21], similar to the approach that we are implementing, except that they used Moving Average as a forecasting model, and Preferential Attachment, Common Neighbors, Adamic Adar and Jaccards Coefficient as topological similarity metrics.

Comparing to Supervised which is a baseline supervised classifier that uses the same topological similarity metrics as features without any forecasting model[3], we also compare our results to performance of individual topological similarity metrics like: Preferential Attachment, Common Neighbors, Adamic Adar and Jaccards Coefficient that were reported in [3]. From our results and the results of different approaches presented in Table 2, we find out that when we take into consideration the temporal information of the network, and when use a combination of topological similarity metrics and an appropriate forecasting model we get better results, and significantly improve the prediction accuracy.

Approches/NetworkS	Travian (Messages)	Travian (Trades)
Our approch with SVM	**0.9423**	**0.8417**
Our approch with LR	**0.9419**	**0.8463**
RPM	0.8970	0.7859
Supervised-MA	0.8002	0.6143
Supervised	0.7568	0.7603
Common Neighbors	0.4968	0.5002
Jaccard Coefficie	0.6482	0.4703
Preferential Attachment	0.5896	0.5441
Adamic/Adar	0.5233	0.4962

TABLE II: Results

V. Conclusion

In this paper we reimplemented a supervised link prediction method, which is based on a time series, a set of topological similarity metrics and a forecasting model to predict the link formation. Our experiments were performed on two datasets from Travian Network Datasets (Trades and Messages dataset) which is a massively multiplayer online games, and the results obtained shows significant improvements over link other prediction approaches. in the future we plan to consider other datasets from different domains to better understand the link prediction task, and defining a degree of confidence for link prediction instead of providing a hard binary classification.

References

[1] M. Al Hasan, V. Chaoji, S. Salem, and M. Zaki, "Link prediction using supervised learning," in *SDM06: workshop on link analysis, counter-terrorism and security*, 2006.

[2] V. Srinivas and P. Mitra, *Link Prediction in Social Networks: Role of Power Law Distribution*. Springer, 2016.

[3] A. Hajibagheri, G. Sukthankar, and K. Lakkaraju, "Leveraging network dynamics for improved link prediction," in *Social, Cultural, and Behavioral Modeling: 9th International Conference, SBP-BRiMS 2016, Washington, DC, USA, June 28-July 1, 2016, Proceedings 9*, pp. 142–151, Springer, 2016.

[4] P. Wang, B. Xu, Y. Wu, and X. Zhou, "Link prediction in social networks: the state-of-the-art," *Science China Information Sciences*, vol. 58, no. 1, pp. 1–38, 2015.

[5] A. Hajibagheri, G. Sukthankar, K. Lakkaraju, H. Alvari, R. T. Wigand, and N. Agarwal, "Using massively multiplayer online game data to analyze the dynamics of social interactions,"

[6] P. R. da Silva Soares and R. B. C. Prudêncio, "Time series based link prediction," in *Neural Networks (IJCNN), The 2012 International Joint Conference on*, pp. 1–7, IEEE, 2012.

[7] Z. Huang and D. K. Lin, "The time-series link prediction problem with applications in communication surveillance," *INFORMS Journal on Computing*, vol. 21, no. 2, pp. 286–303, 2009.

[8] L. Munasinghe and R. Ichise, "Time score: A new feature for link prediction in social networks," *IEICE TRANSACTIONS on Information and Systems*, vol. 95, no. 3, pp. 821–828, 2012.

[9] A. Potgieter, K. A. April, R. J. Cooke, and I. O. Osunmakinde, "Temporality in link prediction: Understanding social complexity," *Emergence: Complexity and Organization*, vol. 11, no. 1, p. 69, 2009.

[10] T. Tylenda, R. Angelova, and S. Bedathur, "Towards time-aware link prediction in evolving social networks," in *Proceedings of the 3rd workshop on social network mining and analysis*, p. 9, ACM, 2009.

[11] X. Li, N. Du, H. Li, K. Li, J. Gao, and A. Zhang, "A deep learning approach to link prediction in dynamic networks," in *Proceedings of the 2014 SIAM International Conference on Data Mining*, pp. 289–297, SIAM, 2014.

[12] P. Sarkar, D. Chakrabarti, and M. Jordan, "Nonparametric link prediction in dynamic networks," *arXiv preprint arXiv:1206.6394*, 2012.

[13] D. M. Dunlavy, T. G. Kolda, and E. Acar, "Temporal link prediction using matrix and tensor factorizations," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 5, no. 2, p. 10, 2011.

[14] M. E. Newman, "Clustering and preferential attachment in growing networks," *Physical review E*, vol. 64, no. 2, p. 025102, 2001.

[15] M. Dillon, "Introduction to modern information retrieval: G. salton and m. mcgill. mcgraw-hill, new york (1983). xv+ 448 pp., 32.95 isbn 0-07-054484-0," 1983.

[16] L. A. Adamic and E. Adar, "Friends and neighbors on the web," *Social networks*, vol. 25, no. 3, pp. 211–230, 2003.

[17] A.-L. Barabâsi, H. Jeong, Z. Néda, E. Ravasz, A. Schubert, and T. Vicsek, "Evolution of the social network of scientific collaborations," *Physica A: Statistical mechanics and its applications*, vol. 311, no. 3-4, pp. 590–614, 2002.

[18] M. Pujari, *Link Prediction in Large-scale Complex Networks (Application to bibliographical Networks)*. PhD thesis, Université Sorbonne Paris Cité, 2015.

[19] S. Wheelwright and S. Makridakis, *management. Systems and controls for financial management serieS*. Simon and Schuster, 1985.

[20] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen, *et al.*, "Mllib: Machine learning in apache spark," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1235–1241, 2016.

[21] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *ACM SIGKDD explorations newsletter*, vol. 11, no. 1, pp. 10–18, 2009.