Link Prediction Schemes Contra Weisfeiler-Leman Models

Katie Brodhead
Department of Mathematics
Florida A&M University
Tallahassee, United States

***Abstract*—Link prediction is of particular interest to the data mining and machine learning communities. Until recently all approaches to the problem used embedding-based methods which leverage either node similarities or latent group memberships towards link prediction. Chen and Zhang recently developed a class of non-embedding approaches called Weisfeiler-Leman (WL) Models. WL-Models extract subgraphs around links and then encode subgraph patterns via adjacency matrices using the so-called Palette-WL algorithm. A training stage then learns nonlinear graph topological features for link prediction. Chen and Zhang compared two WL-Models – a linear regression model ("WLLR") and a neural networks model ("WLNM") – against 12 different common link prediction schemes. In this paper, all author claims are validated for WLLR. Additionally, WLLR is tested against 22 additional embedding-based link prediction techniques arising from common neighbor-, path- and random walk-based schemes. WLLR is shown not to be superior when calculable. In fact, in 80% of the datasets where comparisons were possible, one of our added implementations proved superior.**

Keywords*—Weisfeiler-Leman; link prediction; machine learning; linear regression; common walk; path-based; random walk; stochastic block; matrix factorization

I. Introduction

Improvement in effective link prediction has been of particular interest to the data mining and machine learning communities. Much interest arises from diverse real-world applications. Particular applications include friend recommendation in social networks [2], product recommendation in e-commerce [3], knowledge graph completion [4], finding interactions between proteins [5], and recovering missing reactions in metabolic networks [6]. Additional interests arise from the search to overcoming a central challenge for researchers: determining which method is best for a particular situation, especially when each scheme is grounded in a particular heuristic.

Heuristics range in complexity from the more complex (e.g. stochastic block models [5], probabilistic matrix factorization [7]) to the more simplistic (e.g. common neighbors (CN) [1], Katz index [9]). Heuristics with mid-level complexities include methods which calculate node proximity scores via network topologies or random walks. Amongst the diverse methods which exist, the following two challenges have always persisted.

1) Heuristic complexity does not often translate into corresponding performance. The more simplistic often work well, are more interpretable, and scalable. The Katz and CN indices are exemplary examples. The latter asserts higher link probability as the number of common neighbors increases and is reasonably accurate with respect to links on social networks.

2) All known heuristics lack universal applicability to different kinds of networks. CN is again a prime example: its performance electrical grids and biological networks is quite poor [10] notwithstanding its excellent aforementioned successes. Resistance distance (RN) is a converse example: it performs poor where CN thrives [11]. A study of over 20 different heuristics found flaws in each, making none universally effective performance models [10].

Hitherto, the only resolutions to (1) and (2) have been expert selection or trial-and-error.

A recent KDD paper [40] modifies the Weisfeiler-Leman (WL) algorithm from graph theory towards making link predictions. The modified algorithm is called Palette-WL. Additional algorithmic additional machinery is then built on top which allow for machine learning implements to operate. The authors claim an establishment of new universal model which learns a suitable heuristic directly from a given network, thereby demolishing challenge 2. In addition, reported results demonstrate a superior performance over a wide variety of known link prediction methods, thereby ensuring the demolishment of challenge 1.

In this paper, we implement Palette-WL in MATLAB and train a linear regression model (i.e., the authors' WLLR model) towards validating author claims which the authors test on 12 common link prediction schemes. We also expand testing scope and implement 22 additional tests towards developing a more complete picture of author claims. All 34 aforementioned link prediction schemes are "lean" – that is, they do not require a neural network or advanced support for parallelism or distributed computing. The goal of this work, then, is to test author claims on lean prediction schemes contra WLLR. To that end, we test five of the authors' lean data sets (USAir, NS, PB, Yeast, C.ele). These are described in Section II with results presented in Table V. Three of the authors data sets (Power, Router, and E.coli) are not tested in this paper as these are not "lean". Our future work will perform the same type of analysis on the full WLNM model, and additional non-lean data sets. See Section VI on Future Work.

This paper is organized as follows. In Section II, a high-level overview of link prediction is presented. In Section III, we present Palette-WL and the implementation of WL-Models. Section IV details the many specific link prediction models implemented in this paper along with the key results that were obtained. Section V gives a conclusion, while Section VI presents directions for future work. The tail end of this work, following the References Section, includes an Appendix where the full set of computation results from this paper is presented in various tables.

II. Overview of Link Prediction

Historically, link prediction models have been feature-based (a.k.a. embedding-based) arising either from (1) topological features or (2) latent features.

1) Topological feature models. These models leverage node similarities, either locally or globally. Topological models do not perform well when similarity scores do not capture the network formation mechanisms. Common neighbor-based methods (e.g. CN [1], Adamic-Adar [2]), Path-based methods (e.g., Katz [9]), and random-walk based methods (e.g. PageRank [1]) all fall within this category. A breakdown of each of these categories is given in Section IV.

2) Latent feature Models. These models assume that latent groups exist for nodes and that links are determined by group memberships. Latent models extract group memberships via the low-rank decomposition of a network adjacency matrix [3] or via training which fits probabilistic models [5]. Given these models' focus on individual nodes, a central weakness arises in understanding how networks are formed. Popular methods include ranking methods [17], learning to rank methods [17], matrix factorization [16], and stochastic block methods [5], [18]. This paper implements methods from the latter two.

Weisfeiler-Leman models for link prediction are not feature-based. The ideas which motivated its implementation arose from two research areas related to graph classification: design of efficient graph kernels [14], [19], and effective graph labeling schemes [15] arising from impositions of vertex orderings. Niepert et al. [15], in particular, focus on orderings towards defining receptive fields around node pixels; the fields are then used to learn a convolutional neural network for graph classification. WL-models [40] work by instead extracting subgraphs around links instead of node pixels, and by focusing on link prediction rather than on graph classification.

WL-models are new to the link prediction landscape being only formally published in the recent Chen-Zhang paper [40] which this paper tests. Chen and Zhang, in particular, implement two key WL-Models, WLLR and WLNM, along with a third, Palette-WLNM, an extension of WLNM. Among the two, WLNM is superior. Area Under the receiver operating characteristic Curve (AUC) Results are listed in Table I, split in two parts, for five datasets and twelve non-WL methods. In short, WLNM outperforms nine state-of-the-art link-prediction methods developed by heuristic means (e.g. Katz, PageRank, SimRank, etc.), and three latent feature models (stochastic model block, and two matrix factorization methods); a full explanation of all methods will be given in Section IV. \WLLR is less successful, but nonetheless a strong adversarial method. Palette-WLNM is tested elsewhere in their paper, but not considered in this present paper given this paper's focus on WLLR.

The five datasets used above are USAir, NS, PB, Yeast, and C.ele. USAir is a network of US airlines. NS is a collaboration network of researchers who publish papers on network science. PB is a network of US political blogs. Yeast is a protein-protein interaction network in yeast. C.ele is a neural network of C. elegans. All evaluation methods (CN, Jac, AA, etc., will be described in full detail in Section IV.

TABLE I. Results from [40]

Data	CN	Jac	AA	RA	PA	Katz	RD	PR	SR	SBM	MF-c	MF-r	WLLR[10]	WLNM[10]
USAir	0.940	0.903	0.950	0.956	0.894	0.931	0.898	0.944	0.782	0.944	0.918	0.849	0.896	0.958
NS	0.938	0.938	0.938	0.938	0.682	0.940	0.582	0.940	0.940	0.920	0.636	0.720	0.862	0.984
PB	0.919	0.873	0.922	0.923	0.901	0.928	0.883	0.935	0.773	0.938	0.930	0.943	0.827	0.933
Yeast	0.891	0.890	0.891	0.892	0.824	0.921	0.880	0.927	0.914	0.914	0.831	0.881	0.854	0.956
C.ele	0.848	0.792	0.864	0.868	0.755	0.864	0.740	0.901	0.760	0.867	0.832	0.844	0.803	0.859
Rank	7.875	10.625	7.500	6.875	12.875	7.125	10.375	5.125	11.000	5.625	10.500	9.500	10.125	2.500

IV. PALETTE-WL AND WL-MODELS

WL-models use a modified version of the WL-algorithm, called Palette-WL, from graph theory towards making link predictions. Additional algorithmic additional machinery is then built on top which allow for machine learning implements to operate. Three steps, in particular, flesh out an entire WL-Model.

1) Extract enclosing subgraphs: generates K-vertex neighboring subgraphs.

2) Encode subgraph patterns: via adjacency matrices with vertex ordering given by Palette-WL.

3) Training: learns nonlinear graph topological features for link prediction.

To understand each step, a review of graph labeling functions, along with the base WL-Algorithm is in order. A graph labeling function is a map L: $V \rightarrow C$ from vertices V to an ordered set of colors $C = \{1, ..., n\}$. C uniquely determines the vertex order in an adjacency matrix whenever L is one-to-one. The WL-algorithm ("WL"), then, is a color refinement algorithm which iteratively updates vertex colors on a particular graph labeling function, specified below, until a fixed point is reached. (Palette-WL, discussed later, further ensures that the converged function is one-to-one.)

WL specifically works by iteratively augmenting vertex labels using neighbors' labels. It then compresses augmented labels into new "signature" labels until convergence. At first, all vertices are set to the same color "1". Each vertex gets its new signature string by concatenating its own color and the sorted colors of its immediate neighbors. Vertices are then sorted by the ascending order of their signature strings and assigned new colors 1, 2, 3. Vertices with the same signature strings get the same color. WL is formally presented below. See Fig. 1 for an example of the process.

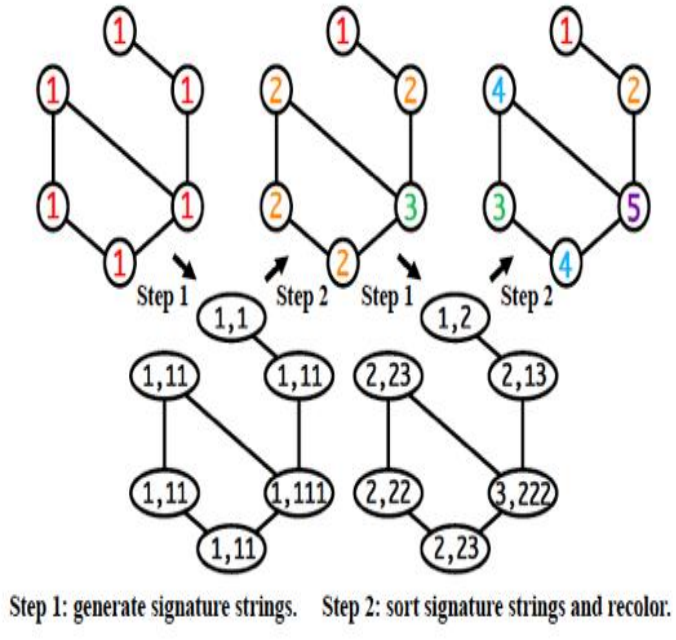

Fig. 1. An example of WL.

WL-Algorithm

1: input: graph G = (V, E), initial colors $c^0(v) = 1, \forall v \in V$

2: output: final colors c (v) for all $v \in V$

3: let $c(v) = c^0(v)$ for all $v \in V$

4: while c(v) has not converged do

5: for each $v \in V$ do

6: collect a multiset $\{c(v') | v' \in \Gamma(v)\}$
containing its neighbors' colors

7: sort the multiset in ascending order

8: concatenate the sorted multiset to c (v)
to generate a signature string
$s(v) = \langle c(v), \{c(v') | v' \in \Gamma(v)\}_{sort} \rangle$

9: end for

10: sort all s(v) in lexicographical ascending order

11: map all s(v) to new colors 1, 2, 3... sequentially;
same strings get the same color

12: **end while**

WL ensures that final colors encode the structural roles of vertices inside a graph. It also defines a relative ordering for vertices, with ties that is consistent across graphs. More specifically, vertices with the same final color share the same structural role within a graph.

We now fully outline the three-step (1-3) process for implementing a WL-model. Step 1 extracts K-vertex enclosing subgraphs via the "Extract Enclosing Subgraphs Algorithm" below. Step 2 then encodes subgraph patterns via adjacency matrices with vertex ordering given by the Palette-WL, also noted below. Fig. 2 gives an overview of these steps in action. Step 3, training, is via any viable machine learning algorithm. The authors use a neural network, WLNM, to achieve superior results. They also train via linear regression, in a method called WLLR.

Extract Enclosing Subgraphs Algorithm

1: input: target link (x,y), network *G*, integer *K*

2: output: enclosing subgraph $G(V_K)$ for (x,y)

3: $V_K = \{x,y\}$

4: fringe = {x,y}

5: while $|V_K| < K$ and $|fringe| > 0$ do

6: $fringe = (\cup_{v \in fringe} \Gamma(v)) \setminus V_K$

7: $V_K = V_K \cup fringe$

8: end while

9: return enclosing subgraph $G(V_K)$

Palette-WL Algorithm

1: input: enclosing subgraph $G(V_K)$ centered at link (x, y),
which is extracted by the EES Algorithm

2: output: final colors c (v) for all $v \in V_K$

3: calculate $d(v) = sqrt[d(v,x)* d(v, y)]$ for all v in V_K

4: get initial colors c(v) = f(d(v))

5: while c(v) has not converged do

6: calculate hashing values h(v) for all v ∈ V_K by (2)

7: get updated colors c(v) = f(h(v))

8: end while

9: return c(v)

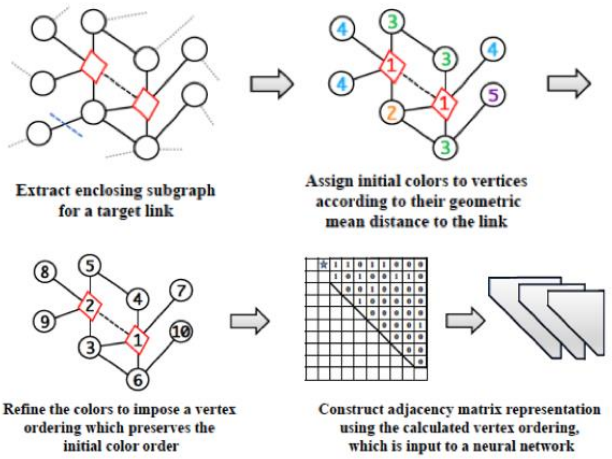

Fig. 2. An overview of a WL-model, Steps 1-2.

Chen and Zhang show, via mathematical proof that the Palette-WL graph labeling function converges in at most K iterations to a one-to-one function for a graph with K vertices. Furthermore, the function is color-order preserving: vertices' color orderings are preserved from state-to-state. Both of these facts enable WL-models to successfully predict links.

V. Assessment of WL-Models

We use Area Under the receiver operating characteristic Curve (AUC) to measure results. AUC measures on the probability that a randomly chosen missing link is given a higher score than a randomly chosen nonexistent link. More precisely, if among n independent comparisons, there are n' times the missing links having a higher score, n'' times those have the same score, the AUC value is $AUC = (n' + 0.5n'')/n$.

A. Assessment Methods

In our experiments we compare the authors WL-model implemented with linear regression (WLLR), against methods from four traditional link-based assessment areas: common neighbor-based methods, path-based methods, random walk-based methods, and latent feature-based methods. For each test, we compute the AUC value and tabulate the results. The Appendix includes tables with all of our results. We outline the algorithms used in assessment below. Our implementation is motivated largely by two recent papers [38], [39].

B. Common Neighbor (CN)-based Methods

For a node x, let Γ(x) be the set of neighbors of x. The idea is that two nodes x and y are more likely to share a link if they have many common neighbors. The most basic measure CN(x,y), defined to be $|\Gamma(x) \cap \Gamma(y)|$, asserts this. Note that if A is the adjacency matrix for the corresponding graph, then $CN(x, y) = A^2(x, y)$.

Now let α be a scaler for a link measure M_α so that $M_\alpha(x,y) = CN(x, y)/\alpha$, and let dx denote the degree of node x. For various choices of α, different CN-measures, noted in Table II, are obtained.

TABLE II. CN-Measures, $|\Gamma(x) \cap \Gamma(y)|$ /alpha

α =	CN	Jac	SltOna	Sora
	1	$\lvert\Gamma(x)\cup\Gamma(y)\rvert$	$sqrt(d_x \times d_y)$	$(d_x + d_y)/2$

α =	HPI[a]	HDI[a]	LHN[a]
	$\min\{d_x, d_y\}$	$\max\{d_x, d_y\}$	$d_x \times d_y$

[a] Measurements not provided in the KDD paper [40]

The Jaccard Index (Jac) [23] gives the probability that x and y are adjacent, given an edge of either x or y. The Salton Index (SltOn) [20] is often also called cosine similarity in the literature. Sørensen Index (Sor) [24] is primarily used for ecological data. Hub Promoted Index (HPI) [25] is used to quantify the topological overlap of pairs of substrates in metabolic network; under this scheme, links adjacent to hubs are more likely to be assigned to be assigned high scores. Hub Depressed Index (HDI) is the complementary measure to HPI. Finally, the Leicht-Holme-Newman Index (LHN) [22] assigns high similarity to node pairs that have many common neighbors compared to the expected number of such neighbors; in particular, $d_x \times d_y$ is proportional to the expected number of common neighbors of nodes x and y in the configuration model [26].

Three related CN-based methods we considered are Preferential Attachment Index (PA), Adamic-Adar Index (AA), and Resource Allocation Index (RA); these are noted in Table III. PA is motivated by a preferential attachment mechanism which ensures that the probability that a new link to be added connects x and y is proportional to $d_x \times d_y$. PA is often used to quantify the functional significance of links subject to various network-based dynamics such as percolation [27], synchronization [28], and transportation [29]. AA is a refinement of simple counting of common neighbors; it assigns less-connected neighbors more weight [2]. RA [8] is motivated by the resource allocation dynamics on complex networks. Suppose x and y are not directly connected and x can send a resource to y, with common neighbors playing the role of transmitters. If each transmitter has a unit of resource, and distributes equally to all its neighbors, then RA is the amount of resource y received from x.

TABLE III. Three Related CN-based Measures

PA	AA	RA
$d_x \times d_y$	$\sum_{z \in \Gamma(x)\cap\Gamma(y)} (1/\log(d_z))$	$\sum_{z \in \Gamma(x)\cap\Gamma(y)} (1/d_z)$

We also considered three local naïve Bayes methods with common neighbor, Adar-Adamic index, and resource allocation, respectively. These are listed as LNBCN, LNAA, and LNBRA in the tables provided in the Appendix.

In our experimental runs on the five data sets, we able to validate all of the KDD results [40] on the measures that were used in that paper: CN, Jac, AA, RA, and PA. See the Appendix, Tables VI, VII and VIII. Numerical values were

rarely the same, but sufficiently close. In addition, in each of our experimental runs inclusive of all additional measures, the RA index generally performs best, while the AA, CN, and LNBAA indices follow closely behind in best overall performance. This "best performance" neglects comparisons against the WLNM test runs. Because WLNM outperforms even AA and CN, WLNM still provides superior performance according to KDD data. A caveat is that we only ran the WL-model with linear regression, called WLLR, which fared worse amongst the various CN-measures above. In fact, almost without exception, all 13 common neighbor methods implemented exhibited superior performance over WLLR on the NS, PB, and Yeast data sets. Exceptions occurred with PA, Jac, LHN, and LNBRA on certain data sets.

C. Path-based Methods

The Katz Index [9] is based on an ensemble of all paths. It is a sum over the collection of all paths with a damping factor β providing shorter paths more weight. Letting A be the adjacency matrix, $Katz(x, y) = \sum_{i \geq 1}(\beta A)^i = (I - \beta A)^{-1} - I$.

The Local Path Index (LP) [8, 33] takes local paths into additional consideration beyond CN and is defined as $LPI(x, y) = A^2 + \epsilon A^3$ where ϵ is a free parameter; note that when $\epsilon = 0$, the index is just CN. A more expanded version allows for n sum factors, and as $n \rightarrow \infty$ the index becomes Katz. Experimental results show that the optimal n is positively correlated with the average shortest distance of the network [33].

The Leicht-Holme-Newman Index (LHNII) [22] is a variant of the Katz index and is based on the concept that two nodes are similar if their immediate neighbors are themselves similar. A self-consistent matrix formulation is $S = \beta AS + \psi I = \psi(I - \beta A)^{-1} = \psi(I + Katz(x, y))$ where ψ and β are free parameters controlling the balance between the two components of the similarity. An formulation useful for computations is $S = 2m\lambda D^{-1}(I - \beta A/\lambda)^{-1}D^{-1}$ where λ is the largest eigenvalue of A, m is the total number of edges in the network, D is the degree matrix, and β is a free parameter. The choosing of β depends on the investigated network, and smaller β assigns more weights on shorter paths.

The KDD paper [40] only implements Katz with $\beta = 0.01$. In our experiments we also run Katz with $\beta = 0.001$, as well as LocalPath, and LHNII with $\beta = 0.9$, 0.95, and 0.99. See Appendix, Tables IX and X. In our implementations, on four data sets (USAir, NS, PB, and Yeast), Katz (both versions) and LocalPath always exhibited superior performance to WLLR, with the exception of Katz ($\beta = 0.01$) on USAir. For the NS dataset, LHNII actually exhibited top performance on WLLR over all methods discussed in this section.

D. Random Walk-based Methods

Resistance Distance (RD) is often called Average Commute Time (ACT) in other contexts and equal to $s(x, y) + s(y, x)$ where $s(x, y)$ denotes the average number of steps required by a random walker starting from node x to reach node y. The pseudoinverse L^+ of the Laplacian matrix $L = D - A$, is easily computable as $m(L^+(x, x) + L^+(y, y) - 2L^+(x, y))$ [11, 30]. ACT(x,y) is defined as the reciprocal of this with m=1 in order to ensure that two nodes are more similar whenever they have a smaller average commute time.

The PageRank (PR) algorithm [13] may be directly applied using Random Walk with Restart (RWR). Consider a random walker starting from node x, who will iteratively move to a random neighbor with probability c and return to node x with probability 1 – c. Let p_{xy} denote the probability that a random walker locates at node y in the steady state. Then $\boldsymbol{p_x} = \langle p_{x1}, p_{x2}, ... \rangle$ is given by $\boldsymbol{p_x} = c \cdot P^T\boldsymbol{p_x} + (1 - c) \cdot e_x$ where e_x is the n × 1 vector in Euclidean n-space which is 1 at entry x and zero elsewhere, and P is the transition matrix with $P(x, y) = 1/d_x$ if x and y are connected and 0 otherwise. The solution, given by $\boldsymbol{p_x} = (1 - c)(I - cP^T)^{-1}e_x$, is used in the code for this project. Define RWR as $RWR(x, y) = p_{xy} + p_{yx}$.

SimRank (SR) [31] is defined in a self-consistent way similar to LHNI as $SR(x, y) = \beta (\sum_{z \in \Gamma(x)} \sum_{z' \in \Gamma(y)} SR(z, z'))/(d_x d_y)$ where $SR(x, x) = 1$ and $\beta \in [0, 1]$ is a decay factor. The underlying idea is that two nodes are similar if they are connected to similar nodes. From a random-walk perspective, SR measures how soon two random walkers, respectively starting from nodes x and y, are expected to meet at a certain node.

Cosine Similarity based on L^+ (Cos+) [30] is defined using pseudoinverse L^+ of the Laplacian matrix $L = D - A$ so that

$Cos+(x, y) = L^+(x, y)/\text{sqrt}(L^+(x, x) \cdot L^+(y, y))$.

Local Random Walk with step s (LRWs) [34] measures the similarity between nodes x and y when random walker is initially put on node x and proceeds for s steps. The density vector is defined by $V_x(0) = e_x$ and $V_x(t + 1) = P^T \cdot V_x(t)$ for $t \geq 0$. Then $LRWs(x, y) = init(x) \cdot V_{xy}(s) + init(y) \cdot V_{yx}(s)$ where $init$ is the initial configuration function. In [34] Liu and Lü determine $init(x)$ by node degree so that $init(x) = d_x/m$.

Superposed Random Walk with step s [34] is similar to the LRWs index and defined as $SRWs(x, y) = \sum_{1 \leq i \leq s} LRWs(x, y)$. Here a random walker is continuously released at the starting point. A higher similarity is between the target node and the nodes nearby results.

Matrix Forest Index [32] is defined by $MFI = (I + L)^{-1}$. MFI gives the ratio of the number of spanning rooted forests (such that nodes x and y belong to the same tree rooted at x) to all spanning rooted forests of the network.

Transfer Similarity with CN (TS) is defined, using a parametrized version of MFI, by $TS = (I + \lambda \cdot CN)^{-1} * CN$.

The KDD paper [40] only implements RD, PR, and SR. In our experiments we also implemented Cos+, RWR with $\beta = 0.95$, LRW with 3-5, SRW with 3-5, MFI, and TS. See Appendix, Tables XI, XII, and XIII. In our implementations, random walk-based methods could not be calculated on two datasets (NS, Yeast) due to memory limitations. On the other three sets (USAir, PB, and Celegens), every random walk method was superior to the WLLR model, with the exception of RD and SR and TS on all sets, and RWR 0.95 and MFI on USAir.

E. Latent Feature-based Methods

Latent (present participle of lateo, "lie hidden") feature-based models attempt recover hidden features which are then used to predict graphs links.

Stochastic Block Model (SBM) [5], [35]-[37] partitions nodes into groups and the probability that two nodes are connected depends solely on the groups to which they belong. Let M be a partition, Q_{ab} denote the probability that groups a and b are connected (so that $Q_{aa} = 1$ for all a), and c_{ab} denote the number of connections (i.e. edges) between groups. The likelihood $L(A|M)$ of observed structure A given M is therefore $\Pi_{a \le b} (Q_{ab})^{c\_(ab)} \cdot (1 - Q_{ab})^{1 - c\_(ab)}$. From this $SBM(x, y)$ [21] is defined via Bayes' Theorem as

$$\int_{\Omega} L(A_{xy} = 1|M)L(A|M)p(M)dM \div \int_{\Omega} L(A|M')p(M')dM'$$

where Ω is the set of all partitions and $p(M)$ is a constant assuming no prior knowledge about the model.

With Matrix Factorization (MF) [39], some entries in A are unknown. MF attempts to approximate A, using only known entries, into two low-dimensional matrices so that $A \approx FG^T$ with F being $N \times K$, G being $N \times K$, and K being the number of latent features. The squared error is thus given by $(e_{ij})^2 = (a_{ij} - \sum_{k \le K} f_{ik} g_{kj})^2$. A regularization technique adds a factor to avoid over-fitting so that instead $(e_{ij})^2 = (a_{ij} - \sum_{k \le K} f_{ik} g_{kj})^2 + (\beta/2) \sum_{k \le K} (\|F\|^2 + \|G\|^2)$. The goal is to minimize the sum of all squared errors to obtain optimal F and G. The gradient at current values is calculated via partial differentiation with respect to f_{ik} and separately with respect to g_{kj}. Weights are then updated in the direction opposite the gradient and this gives rules $f_{ik}' = f_{ij} + \alpha (2 e_{ij} g_{kj} - \beta f_{ik})$ and $g_{kj}' = g_{kj} + \alpha (2 e_{ij} f_{ik} - \beta g_{kj})$ which are then used iteratively until error converges to a minimum. Implicit in the above formulation is that squared errors must be known elements of A, so a_{ij} is in the training set.

Amongst the two methods, our results showed that SB generally provided better results with respect to link prediction. See Appdendix, Table XIV. For the USAir and Celegens data sets, SB outperformed the WLLR. For the PB and Celegens datasets, MF likewise outperformed WLLR.

F. Summary

Each of the aforementioned methods were implemented with various parameters as outlined in the Appendix. In particular, our implementation extends the KDD authors' implementation by testing all data sets on the additional common neighbor schemes (13 instead of 5), path-based methods (6 instead of 1), random walk-based methods (13 instead of 3). We also implemented all of the authors' latent feature models. On the five datasets sets tested, all author data on these methods were able to be validated. In our implementation, a WL-model is run as linear regression and noted as WLLR. WLLR did not exhibit superior performance in any run for which comparisons were possible. Amongst the comparisons possible, in 80% of these our implementations (not implemented in the KDD paper) demonstrated superior performance. In particular, WLLR AUC results for NS, PB, Yeast, and Celegens were 0.865, 0.838, 0.860, and 0.804, respectively; the corresponding results for LHRII (all cases), LRW3, LNBAA, and LRW3 were 0.9690, 0.9367, 0.8990, and 0.9197, respectively. For USAir, the WLLR AUC score was 0.930 and RA demonstrated superior performance with an AUC score of 0.9540. Common neighbor methods were superior in 40% of the cases. Random walk-based methods were superior in another 40% of the cases. Finally, the Path-based methods were superior in the remaining 20% of cases. Table IV demonstrates these results.

TABLE IV. COMPUTATIONS

Data Set	WLLR	Top Score	Models with Top Score	Top Score Class	Top Score in this Paper
USAir	0.930	0.9540	RA	Common neighbor	
NS	0.865	0.9690	LHNII B ∈{3,4,5}	Path-based	✔
PB	0.838	0.9367	LRW3	Random-walk	✔
Yeast	0.860	0.8990	AA, RA, LNBAA	Common neighbor	✔ [a]
C.ele	0.804	0.9197	LRW3	Random-walk	✔

[a.] AA and RA were implemented in [40]; LNBAA was implemented in this paper

VI. CONCLUSION

Chen and Zhang [40] develop novel WL-link prediction scheme which to-date best predicts links in real world graph data, according to experimental data. An innovative feature is that link formation mechanisms are learned, not assumed. WL-link prediction schemes work by encoding enclosed subgraphs as adjacency matrices. Encoding occurs via the author's modification of the Weisfeiler-Leman (WL) algorithm [12] from graph theory. The authors' modification, Palette-WL, labels vertices according to their structural roles in the subgraph and preserves subgraph intrinsic directionality. Training on adjacency matrices then learns a predictive model. When training is done on the authors' neural network, the model is called WLNM, i.e. WL Neural Machine, and this requires advanced support for parallelism or distributed computing. When training is done with linear regression, the model is called WLLR, i.e. WL Linear Regression, and this was the focus of our work; forthcoming work will tackle WLNM.

We extended the authors' implementations by testing all data sets on additional CN-, path- and random walk-based schemes. In particular, we implemented 32 such schemes compared to the nine from the KDD authors. We also implemented all of the authors' latent feature models. On five data sets, all author data on these methods were validated. The WLNM model still demonstrates superiority even when all additional schemes are implemented, according to data provided by the KDD authors. The linear regression version of the model, WLLR, on the other hand, was not superior when calculable. In fact, in 80% of the datasets where comparisons were possible, one of our added implementations proved superior.

VII. Future Work

Given the successes in validating the results from [40] for WLLR and in demonstrating a multitude of results which supplant that model, our next step will be to implement and validate WLNM. That is, the current work is limited in scope to only the WLLR model. Therefore, the goal will be to determine if the WLNM model can also be supplanted. We will also run all experiments on the three additional data sets (Power, Router, and E.coli) which the authors test in [40] and which require advanced support for parallelism or distributed computing.

Other opportunities for future work also abound. For instance, in what ways can might the graph coloring scheme be applied to other algorithms in data science (e.g. clustering)? Also, as the authors' algorithm pertains only to undirected graphs, how might the WLNM be modified in order to apply to directed graphs? It is also plausible that for specific classes of graphs, a parred set of calculations might suffice towards leading to manageable neural network computations; one direction for future work would be to identify such classes and prove optimal computational bounds. Another line of work would be to determine whether there might be a class of circumstances in which a heuristic method (or an embedding model) may provide a better result. If so, what properties, would such a class or model have, mathematically, and could an example be found in nature?

References

[1] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks", Journal of the American society for information science and technology, Vol. 58, No. 7, pp. 1019–1031, 2007.

[2] L. A. Adamic and E. Adar, "Friends and neighbors on the web", Social networks, Vol. 25, No. 3, pp. 211–230, 2003.

[3] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems", Computer, Vol. 8, pp. 30–37, 2009.

[4] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich, "A review of relational machine learning for knowledge graphs", arXiv preprint, 1503.00759, 2015.

[5] E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing, "Mixed membership stochastic blockmodels", Journal of Machine Learning Research, Vol. 9, pp. 1981–2014, September 2008.

[6] T. Oyetunde, M. Zhang, Y. Chen, Y. Tang, and C. L. Boostgapfill, "Improving the fidelity of metabolic network reconstructions through integrated constraint and pattern-based methods", Bioinformatics, 2016.

[7] R. Salakhutdinov and A. Mnih, "Bayesian probabilistic matrix factorization using markov chain monte carlo", Proceedings of the 25th international conference on Machine learning (ICML), ACM, pp. 880–887, 2008.

[8] T. Zhou, L. Lü, and Y. Zhang, "Predicting missing links via local information", European Physical Journal B, Vol. 71, No. 4, pp. 623–630, 2009.

[9] L. Katz, "A new status index derived from sociometric analysis", Psychometrika, Vol. 18, No. 1, pp. 39–43, 1953.

[10] L. Lü and T. Zhou, "Link prediction in complex networks: A survey", Physica A: Statistical Mechanics and its Applications, Vol. 390, No. 6, pp. 1150–1170, 2011.

[11] D. J. Klein and M. Randic, "Resistance distance", Journal of Mathematical Chemistry, Vol. 12, No. 1, pp. 81–95, 1993.

[12] B. Weisfeiler and A. A. Lehman, "A reduction of a graph to a canonical form and an algebra arising during this reduction", Nauchno-Technicheskaya Informatsia, Vol. 2, No. 9, pp. 12–16, 1968.

[13] S. Brin and L. Page, Reprint of: "The anatomy of a large-scale hypertextual web search engine", Computer networks, Vol. 56, No. 18, pp. 3825–3833, 2012.

[14] N. Shervashidze, P. Schweitzer, E. J. van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-lehman graph kernels", Journal of Machine Learning Research, Vol. 12, pp. 2539–2561, September 2011.

[15] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs", Proceedings of the 33rd annual international conference on machine learning, ACM, 2016.

[16] K. Miller, M. I. Jordan, and T. L. Griffiths, "Nonparametric latent feature models for link prediction", Advances in neural information processing systems, pp. 1276–1284, 2009.

[17] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback", Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence, AUAI Press, pp. 452–461. 2009.

[18] C. Aicher, A. Z. Jacobs, and A. Clauset, "Learning latent block structure in weighted networks", Journal of Complex Networks, Vol. 3, No. 2, pp. 221–248, 2015.

[19] 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, April 2010.

[20] G. Salton, M. J. McGill, Introduction to Modern Information Retrieval, MuGraw-Hill, Auckland, 1983.

[21] R. Guimera, M. Sales-Pardo, "Missing and spurious interactions and the reconstruction of complex networks", Proc. Natl. Acad. Sci. U.S.A., Vol. 106, 2009.

[22] E. A. Leicht, P. Holme, M. E. J. Newman, "Vertex similarity in networks", Phys. Rev. E, Vol. 73, 2006.

[23] P. Jaccard, "E´tude comparative de la distribution florale dans une portion des Alpes et des Jura", Bulletin de la Societe Vaudoise des Science Naturelles, Vol. 37, No. 1901, pp. 547.

[24] T. Sørensen, "A method of establishing groups of equal amplitude in plant sociology based on similarity of species content and its application to analyses of the vegetation on Danish commons", Biol. Skr., Vol. 5, pp. 1, 1948.

[25] E. Ravasz, A. L. Somera, D. A. Mongru, Z. N. Oltvai, and A. L. Barabasi, "Hierarchical organization of modularity in metabolic networks", Science, Vol. 297, No. 1551, 2002.

[26] M. Molloy and B. Reed, "A critical point for random graphs with a given degree sequence", Random Structure and Algorithms, Vol. 6, No. 161, 1995.

[27] P. Holme, B. J. Kim, C. N. Yoon, and S. K. Han, "Attack vulnerability of complex networks", Phys. Rev. E, Vol. 65, 2002.

[28] C. Y. Yin, W. X. Wang, G. R. Chen, and B. H. Wang, "Decoupling process for better synchronizability on scale-free networks", Phys. Rev. E, Vol. 74, 2006.

[29] G. Q. Zhang, D. Wang, and G. J. Li, "Enhancing the transmission efficiency by edge deletion in scale-free networks", Phys. Rev. E, No. 76, 2007.

[30] F. Fouss, A. Pirotte, J. M. Renders, and M. Saerens, "Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation", IEEE Trans. Knowl. Data. Eng., Vol. 19, No. 355, 2007.

[31] G. Jeh and J. Widom, "SimRank: A measure of structural-context similarity", Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM Press, New York, pp. 271-279, 2002.

[32] P. Chebotarev and E. V. Shamis, "The matrix-forest theorem and measuring relations in small social groups", Automation and Remote Control, Vol. 58, No. 1505, 1997.

[33] L. Lü, C. H. Jin, and T. Zhou, "Similarity index based on local paths for link prediction of complex networks", Phys. Rev. E, Vol. 80, 2009.

[34] W. Liu and L. Lü, "Link prediction based on local random walk", EPL, Vol. 89, 2010.

[35] H. C. White, S. A. Boorman, and R. L. Breiger, "Social structure from multiple networks I: Blockmodels of roles and positions", Am. J. Sociol., Vol. 81, No. 730, 1976.

[36] P. W. Holland, K. B. Laskey, and S. Leinhardt, "Stochastic blockmodels: First steps", Social Networks, Vol. 5, No. 109, 1983.

[37] P. Dorelan, V. Batagelj, and A. Ferligoj, Generalized Blockmodeling, Cambridge University Press, Cambridge, UK, 2005.

[38] L. Lü and T. Zhao, "Link Prediction in Complex Networks: A survey", "Physica A: Statistical Mechanics and its Applications", Vol. 390, No. 6, pp. 1150-1170, 2011.

[39] Z. Wu and Y. Chen, "Link prediction using matrix factorization with bagging", 15th Int. Conf. on Computer and Inf. Sci. (ICIS), Okayama, pp. 1-6, 2016.

[40] Y. Chen and M. Zhang, Weisfeiler-Lehman Neural Machine for Link Prediction, 23rd ACM SIGKDD Int. Conf. on KDD Mining, New York, NY, USA, pp. 575-583, 2017.

APPENDIX

The full set of results from this paper are presented in various tables as Area Under the receiver operating characteristic Curve (i.e. AUC) measurements ("Test"). Calculations are towards validating the results in [40] ("Paper").

Entries with '–' require a Graphics Processing Unit for calculations and are outside the scope of this study. Tables or columns marked with * contain results run exclusively in this paper; no such computations were run in [40] ("Paper").

TABLE V. WEISFEILER-LEMAN BASED METHODS

No	Data	Source	WLLR 10[a]	WLNM 10[b]
1	USAir	Paper	0.896	0.958
		Test	0.930	–
2	NS	Paper	0.862	0.984
		Test	0.865	–
3	PB	Paper	0.827	0.933
		Test	0.838	–
4	Yeast	Paper	0.854	0.956
		Test	0.860	–
5	C.ele	Paper	0.803	0.859
		Test	0.804	–

[a.] Weisfeiler-Leman Linear Regression Model, K = 10 (WLLR 10)

[b.] Weisfeiler-Leman Neural Machine, K = 10 (WLMN 10)

TABLE VI. COMMON NEIGHBOR METHODS I

No	Data	Source	CN[a]	Jac[b]	AA[c]	RA[d]	PA[e]
1	USAir	Paper	0.940	0.903	0.950	0.956	0.894
		Test	0.939	0.905	0.950	0.954	0.899
2	NS	Paper	0.938	0.938	0.938	0.938	0.682
		Test	0.945	0.945	0.945	0.945	0.705
3	PB	Paper	0.919	0.873	0.922	0.923	0.901
		Test	0.912	0.867	0.915	0.917	0.897
4	Yeast	Paper	0.891	0.890	0.891	0.892	0.024
		Test	0.898	0.897	0.899	0.899	0.836
5	C.ele	Paper	0.848	0.792	0.864	0.868	0.755
		Test	0.842	0.782	0.858	0.862	0.761

[a.] Common Neighbor (CN)

[b.] Jaccard Index (Jac)

[c.] Adar-Adamic Index (AA)

[d.] Resource Allocation (RA)

[e.] Preferential Attachment (PA)

TABLE VII. COMMON NEIGHBOR METHODS II*

No	Data	Source	SltOn[a]	Sor[b]	HPI[c]	HDI[d]
1	USAir	Test	0.9037	0.8959	0.8621	0.8891
2	NS	Test	0.9450	0.945	0.9449	0.9449
3	PB	Test	0.8692	0.8673	0.8478	0.8637
4	Yeast	Test	0.8972	0.8972	0.8961	0.8971
5	C.ele	Test	0.7897	0.7816	0.7958	0.7685

[a.] Salton Index (SltOn)

[b.] Sorenson Index (Sor)

[c.] Hub Promoted Index (HPI)

[d.] Hub Depressed Index (HDI)

TABLE VIII. COMMON NEIGHBOR METHODS III

No	Data	Source	LHN*	LNBCN[a]	LNBAA[b]	LNBRA[c]
1	USAir	Test	0.7615	0.9434	0.9503	0.8943
2	NS	Test	0.9446	0.9452	0.9452	0.7045
3	PB	Test	0.7584	0.915	0.9165	0.8973
4	Yeast	Test	0.8932	0.8987	0.899	0.8355
5	C.ele	Test	0.716	0.858	0.8623	0.7605

[a.] Leicht-Holme-Newman (LHN)

[b.] Local naive bayes method with Common Neighbor (LNBCN)

[c.] Local naive bayes method with Adar-Adamic Index (LNBAA)

[d.] Local naive bayes method with Resource Allocation (LNBRA)

TABLE IX. PATH-BASED METHODS I

No	Data	Source	Katz with β= 0.01[a]
1	USAir	Paper	0.931
		Test	0.926
2	NS	Paper	0.940
		Test	0.947
3	PB	Paper	0.928
		Test	0.924
4	Yeast	Paper	0.921
		Test	–
5	C.ele	Paper	0.864
		Test	0.860

[a.] Katz Index with damping factor β = 0.01

TABLE X. PATH-BASED METHODS II

No	Data	Source	Katz β= 0.001[a]	LocalPath[b]	LHNII 0.9[c]	LHNII 0.95[d]	LHNII 0.99[e]
1	USAir	Test	0.9279	0.9306	0.6040	0.5870	0.5712
2	NS	Test	0.9474	0.9499	0.9690	0.9690	0.9690
3	PB	Test	0.9266	0.9273	0.6363	0.5810	0.5273
4	Yeast	Test	–	–	–	–	–
5	C.ele	Test	0.8614	0.8626	0.6070	0.5551	0.5003

[a] Katz Index with damping factor β = 0.001
[b] Local Path Index (LocalPath)
[c] Leicht-Holme-Newman II with 0.90
[d] Leicht-Holme-Newman II with 0.95
[e] Leicht-Holme-Newman II with 0.99

TABLE XI. RANDOM-WALK BASED METHODS I

No	Data	Source	RD[a]	PR[b]	SR[c]
1	USAir	Paper	0.898	0.944	0.782
		Test	0.911	0.931	0.775
2	NS	Paper	0.582	0.940	0.940
		Test	–	–	–
3	PB	Paper	0.883	0.935	0.773
		Test	0.879	0.930	0.771
4	Yeast	Paper	0.880	0.927	0.914
		Test	–	–	–
5	C.ele	Paper	0.740	0.901	0.760
		Test	0.726	0.899	0.758

[a] Resistance Distance, or Average Commute Time (RD)
[b] PageRank, or Random Walk with restart, with damping factor d = 0.85 (PR)
[c] SimRank with 0.6 (SR)

TABLE XII. RANDOM-WALK BASED METHODS II*

No	Data	Source	Cos+[a]	RWR[b]	LRW 3[c]	LRW 4[d]	LRW 5[e]
1	USAir	Test	0.9342	0.914	0.9389	0.9367	0.9337
2	NS	Test	–	–	–	–	–
3	PB	Test	0.9196	0.917	0.9367	0.9293	0.9325
4	Yeast	Test	–	–	–	–	–
5	C.ele	Test	0.865	0.857	0.9197	0.9034	0.9105

[a] Cos+ based on Laplacian matrix (Cos+)
[b] Random walk with restart with damping factor 0.95 (RWR 0.95)
[c] Local Random Walk with step 3 (LRW 3)
[d] Local Random Walk with step 4 (LRW 4)
[e] Local Random Walk with step 5 (LRW 5)

TABLE XIII. RANDOM-WALK BASED METHODS III

No	Data	Source[a]	SRW 3[a]	SRW 4[b]	SRW 5[c]	MFI[d]	TS
1	USAir	Test	0.9407	0.9389	0.9384	0.9129	0.589
2	NS	Test	–	–	–	–	–
3	PB	Test	0.9257	0.9272	0.9292	0.8959	0.4417
4	Yeast	Test	–	–	–	–	–
5	C.ele	Test	0.9009	0.9031	0.9063	0.8722	0.5076

[a] Superposed Random Walk with step 3 (SRW 3)
[b] Superposed Random Walk with step 4 (SRW 4)
[c] Superposed Random Walk with step 5 (SRW 5)
[d] Matrix Forest Index (MFI)
[e] Transfer Similarity (TS)

TABLE XIV. LATENT FEATURE BASED METHODS

No	Data	Source	SBM[a]	MF-c[b]
1	USAir	Paper	0.944	0.918
		Test	0.932	0.914
2	NS	Paper	0.920	0.636
		Test	–	0.620
3	PB	Paper	0.938	0.930
		Test	–	0.927
4	Yeast	Paper	0.914	0.831
		Test	–	–
5	C.ele	Paper	0.867	0.832
		Test	0.878	0.837

[a] Stochastic Block Method (SBM)
[b] Matrix Factorization with classification loss function (MF-c)