Similarity-based Link Prediction from Modular Compression of Network Flows

Christopher Blöcker
Integrated Science Lab, Department of Physics
Umeå University
christopher.blocker@umu.se

Jelena Smiljanić[*]
Integrated Science Lab, Department of Physics
Umeå University
jelena.smiljanic@umu.se

Ingo Scholtes
Center for Artificial Intelligence and Data Science
University of Würzburg
ingo.scholtes@uni-wuerzburg.de

Martin Rosvall
Integrated Science Lab, Department of Physics
Umeå University
martin.rosvall@umu.se

Abstract

Node similarity scores are a foundation for machine learning in graphs for clustering, node classification, anomaly detection, and link prediction with applications in biological systems, information networks, and recommender systems. Recent works on link prediction use vector space embeddings to calculate node similarities in undirected networks with good performance, but they have several disadvantages: limited interpretability, they require hyperparameter tuning, manual model fitting through dimensionality reduction, and poor performance of *symmetric* similarities in *directed* link prediction. We propose *MapSim*, an information-theoretic measure to assess node similarities based on modular compression of network flows. Different from vector space embeddings, *MapSim* represents nodes in a discrete, non-metric space of communities and yields *asymmetric* similarities in an unsupervised fashion. We compare *MapSim* on a link prediction task to popular embedding-based algorithms across 47 data sets of networks and find that *MapSim*'s average performance across all networks is more than 7% higher than its closest competitor, outperforming all embedding methods in 11 of the 47 networks. Our method demonstrates the potential of compression-based approaches in graph representation learning, with promising applications in other graph learning tasks.

1 Introduction

Calculating similarity scores between objects is a fundamental problem in machine learning tasks from clustering, anomaly detection, and text mining to classification and recommender systems. In Euclidean feature spaces, similarities between feature vectors are commonly calculated as lengths, norms, angles, or other geometric concepts, possibly using *kernel functions* that perform implicit non-linear mappings to high-dimensional feature spaces [1]. For relational data represented as graphs, methods using the graph topology to calculate pairwise node similarities can address learning problems such as graph clustering, node classification, and link prediction. For link prediction, recent works take a multi-step approach and separate *representation learning* and *link prediction* [2, 3]: First, they learn a latent-space node embedding from the graph's topology, using methods such as graph or matrix factorisation [4, 5], or random-walk based techniques [6–8]. Then, they interpret node positions as points in a high-dimensional feature space, possibly applying downstream dimensionality reduction. Finally, they use node positions in the resulting feature space to assign new "features" to pairs of nodes, which can be used to predict links. Taking an unsupervised approach, we predict links

[*]Also with the Center for the Study of Complex Systems, Institute of Physics, University of Belgrade.

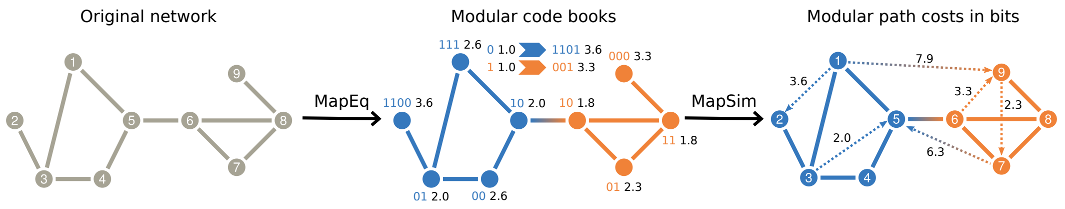

Figure 1: We calculate node similarities for predicting links based on a network's modular coding scheme of the map equation. Blue and orange nodes have a unique codeword within their module derived from their stationary visit rates. Decimal numbers show the theoretical lower limit for the codeword length in bits. *Map equation similarity* derives description lengths for predicted links, connecting more similar nodes uses fewer bits. Intra-community links tend to have shorter description lengths than inter-community links.

based on node similarities [9] by calculating distance metrics or similarity scores between node pairs to rank them. We can alternatively use a supervised approach [10] by (i) using binary operators like the Hadamard product [7], (ii) sampling negative instances (node pairs not connected by links), and (iii) using the features of positive and negative instances to train a supervised binary classifier [7].

Advances in graph embedding and representation learning have considerably improved our ability to predict links in networks, with applications in biological [11] and social [12] networks and in recommender systems [13]. However, these methods introduce challenges for real-world link-prediction tasks: First, they require specifying hyperparameters that control aspects regarding the *scale* of patterns in graphs, influence of local and non-local structures, and latent space dimensionality [14]. Network-specific hyperparameter tuning addresses these issues, but is challenging in real applications and aggravates the risk of overfitting; recent systematic comparisons reveal that the performance of different methods largely varies across data sets [2, 3]. Altogether, this makes it difficult for practitioners to choose and optimally parametrise an embedding method. Second, using latent metric spaces implies *symmetric* similarities, limiting the performance when predicting *directed* links [5, 15]. Third, compared with hand-crafted features, embeddings tend to have low interpretability: We can assess the similarity of nodes, but we cannot explain *why* some nodes are more similar than others [2–4]. Nevertheless, recent graph neural network-based approaches focus on learning features for link prediction from local subgraphs [16], overlapping node neighbourhoods [17], or based on shortest paths [18], achieving favourable performance. Finally, recent works highlight fundamental limitations of low-dimensional representations of complex networks [19], questioning to what extent Euclidean embeddings can capture patterns relevant for link prediction.

Motivated by recent works highlighting the importance of community structures for link prediction [2, 20, 21], we propose a novel approach to similarity-based link prediction that addresses these issues. Our contributions are:

- We introduce map equation similarity, *MapSim* for short, an information-theoretic method to calculate asymmetric node similarities. *MapSim* builds on the map equation [22], a framework that applies coding theory to compress random walks based on hierarchical cluster structures.
- Different from other random walk-based embedding techniques, our work builds on an analytical approach to calculate the expected description length of random walks in the limit, and, thus, requires neither simulating random walks nor tuning hyperparameters.
- Following the minimum description length principle, *MapSim* incorporates Occam's razor and balances explanatory power with model complexity, making dimensionality reduction superfluous. With hierarchical cluster structures, *MapSim* captures patterns at multiple scales simultaneously and combines advantages of local and non-local similarity scores.
- We validate *MapSim* in an unsupervised, similarity-based link prediction task and compare its performance to six well-known embedding-based techniques in 47 empirical networks from different domains. Highlighting challenges in the generalisability of embedding techniques and parametrisations across different networks, this analysis is a contribution in itself.
- Confirming recent surveys, we find that, without network-specific hyperparameter tuning, the performance of popular embedding techniques for unsupervised link prediction depends on the data. In contrast, *MapSim* provides high performance across a wide range of networks, with an average performance 7.7% and 7.5% better than the best competitor in undirected and directed

networks, respectively. *MapSim* ourtperforms the chosen baseline methods in 11 of the 47 networks with a worst-case performance 44% and 33% better compared to popular embedding techniques in undirected and directed networks, respectively.

In summary, we take a novel perspective on graph representation learning that fundamentally differs from other random walk-based graph embeddings. Rather than embedding nodes into a metric space, leading to symmetric similarities, we develop an unsupervised learning framework where (i) positions of nodes in a coding tree capture their representation in a non-metric latent space, and (ii) node similarities are calculated based on how well transitions between nodes are compressed by a network's hierarchical modular structure (figure 1). Apart from node similarities that can be "explained" based on community structures captured in the coding tree, *MapSim* yields asymmetric similarity scores that naturally support link prediction in directed networks. We provide a simple, non-parametric, and scalable unsupervised method with high generalisability across data sets, and which should thus be of interest for practitioners. Our work demonstrates the power of compression-based approaches to graph representation learning, with promising applications in other graph learning tasks.

2 Related Work and Background

We first summarise recent works on graph embedding and similarity-based link prediction. Then, we review the map equation, an information-theoretic objective function for community detection and theoretical foundation of our compression-based similarity score.

2.1 Related Work

Focusing on unsupervised similarity-based link prediction, we consider methods that calculate a bivariate function $\text{sim}(u, v) \in \mathbb{R}^d$, where $u, v \in V$ are nodes in a directed or undirected, possibly weighted graph $G = (V, E)$ [23, 24]. While similarity metrics often consider scalar functions ($d = 1$), recent vector space embeddings use binary operators to assign vector-valued "features" with $d > 1$ to node pairs. Since vectorial features are typically used in downstream classification techniques, this can be seen as an *implicit* mapping to similarities, for example "similar" features being assigned similar class probabilities. We limit our discussion to *topological or structural approaches* [23], functions $\text{sim}(u, v)$ that can be calculated solely based on the edges E in graph G without requiring additional information such as node attributes or other non-topological graph properties.

Several works define scalar similarities based on local topological characteristics such as the Jaccard index of neighbour sets, degrees of nodes, or degree-weighted measures of common neighbours [25]. Other methods define similarities based on random walks, paths, or topological distance between nodes [9, 26–28]. Compared to purely local approaches, an advantage of random walk-based methods is their ability to incorporate both local and non-local information, which is crucial for sparse networks where nodes may lack common neighbours. Since walk-based methods reveal cluster patterns in networks [22], they generally perform well in downstream tasks such as link prediction and graph clustering [2]. Graph factorisation approaches that use eigenvectors of different types of *Laplacian matrices* that represent relationships between nodes share this high performance [29], likely because (i) Laplacians capture the dynamics of continuous-time random walks [30], and (ii) spectral methods can capture *small cuts* in graphs [31].

Building on these ideas, recent works on *graph representation learning* combine random walks and deep learning to obtain high-dimensional vector space embeddings of nodes, serving as features in downstream learning tasks [3, 14]: Perozzi et al. [6] generate a large number of short random walks to learn latent space representations of nodes by applying a word embedding technique that considers node sequences as word sequences in a sentence. This corresponds to an implicit factorisation of a matrix whose entries capture the logarithm of the expected probabilities to walk between nodes in a given number of steps [32]. Following a similar walk-based approach, Grover and Leskovec [7] generate node sequences with a biased random walker whose exploration behaviour can be tuned by *search bias* parameters p and q. The resulting walk sequences are used as input for the word embedding algorithm `word2vec` [33], which embeds objects in a latent vector space with configurable dimensionality. Tang et al. [8] construct vector space embeddings of nodes that simultaneously preserve first- and second-order proximities between nodes. Similar to Adamic and Adar [25], second-order node proximities are defined based on common neighbours. Extending the random walk approach in [6], Perozzi et al. [34] learn embeddings from so-called walklets, random walks that skip some nodes, resulting in embeddings that capture structural features at multiple scales.

The abovementioned graph embedding methods compute a representation of nodes in a, compared to the number of nodes in the network, low-dimensional Euclidean space. A suitably defined metric for *similarity* or *distance* of nodes enables recovering the link topology with high fidelity [35], forming the basis for similarity-based link prediction. In contrast, [10] argued for a new perspective that uses supervised classifiers based on (i) multi-dimensional features of node pairs, and (ii) an undersampling of negative instances to address inherent class imbalances in link prediction. Recent applications of graph embedding to link prediction have taken a similar supervised approach, for example using vector-valued binary operators to construct features for node pairs from node vectors [6, 7, 24]. Despite good performance, recent works have cast a more critical light on such applications of low-dimensional graph embeddings. Questioning the distinction between deep learning-based embeddings and graph factorisation techniques, Qiu et al. [4] show that popular embedding techniques can be understood as (approximate) factorisations of matrices that capture graph topology. Thus, low-dimensional embeddings can be viewed as a (lossy) compression of graphs, while link prediction or graph reconstruction can be viewed as the decompression step. Fitting this view, a recent study of the topological characteristics of networks' low-dimensional Euclidean representations has highlighted fundamental limitations of embeddings to capture complex structures found in real networks [19].

Techniques like node2vec, LINE, or DeepWalk have been reported to perform well for link prediction despite those limitations. However, recent surveys concur that finetuning their hyperparameters to the specific data set is required [2, 21, 36], which can be problematic in large data sets and increases the risk of overfitting. Comparative studies found that the performance of Euclidean graph embeddings for link prediction is connected to their ability to represent communities in the graph as clusters in the feature space [2], which, due to the non-linear nature of graph data [37], strongly depends on their topology. Using symmetric operators or distance measures in metric spaces limits their ability to predict *directed* links because the ground truth for (u, v) can differ from (v, u) [15].

These issues raise the general question whether we should address link prediction based on low-dimensional Euclidean embeddings. Recent works addressed some of those open questions, for example with hyperbolic or non-linear embeddings [20, 37], extensions of Euclidean embeddings for directed link prediction [15], or embeddings that explicitly account for community structures [21, 38, 39]. However, existing works still use hyperparameters, require separate dimensionality reduction or model selection techniques to identify the optimal number of dimensions, fail to capture rich hierarchically nested community structures present in real-world networks [40], or do not integrate community detection with the actual representation learning. Addressing all issues at once, we take a novel approach that treats graph representation learning as a compression problem: We use the map equation [22], an analytical information-theoretic approach to compress flows of random walks in directed or undirected, possibly weighted networks based on their modular structure. The resulting hierarchical coding tree with node assignments can be viewed as an embedding in a discrete, non-metric latent space of (possibly hierarchical) community labels with automatically optimised dimensionality using a minimum description length approach. As an analytical approach, our method neither introduces hyperparameters nor needs to simulate random walks, and naturally yields asymmetric node similarities suitable to predict directed links.

2.2 Background: the map equation

The map equation is an information-theoretic objective function for community detection that, conceptually, models network flows with random walks [22]. To detect communities, the map equation compresses the random walks' per-step description length by searching for sets of nodes with long flow persistence: network areas where a random walker tends to stay for a longer time.

Consider a communication game where the sender observes a random walker on a network, and uses binary codewords to update the receiver about the random walker's location. In the simplest case, all nodes belong to the same module and we use a Huffman code to assign unique codewords to the nodes based on their stationary visit rates. With a one-module partition, the sender communicates one codeword per random walk step to the receiver. The theoretical lower limit for the per-step description length, we call it *codelength*, is the entropy of the nodes' visit rates [41],

$$L(\mathsf{M}_1) = \mathcal{H}(P) = -\sum_{u \in V} p_u \log_2 p_u. \tag{1}$$

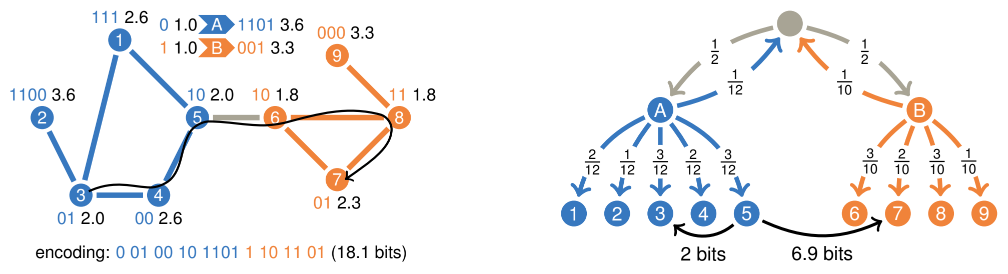

Figure 2: Map equation coding principles. **Left:** An example network with nine nodes, ten links, and two communities, A and B, indicated by colours. Each random-walker step is encoded by one codeword for intra-module transitions or three codewords for inter-module transitions. Codewords are shown next to nodes in colours, their length in the information-theoretic limit in black. Module entry and exit codewords are shown to the left and right of the coloured arrows, respectively. The black trace shows a possible section of a random walk with its encoding at the bottom. **Right:** The corresponding coding tree. Links are annotated with transition rates to calculate similarities in the information-theoretic limit. Each coding tree path corresponds to a network link, which may or may not exist. The coder remembers the random walker's module but not the most recently visited node. Describing the intra-module transition from node 5 to 3 requires $-\log_2(3/12) = 2$ bits. The inter-module transition from node 5 to 7 requires three steps and $-\log_2(1/12 \cdot 1/2 \cdot 2/10) \approx 6.9$ bits.

Here M_1 is the one-module partition, $\mathcal{H}$ is the Shannon entropy, P is the set of the nodes' visit rates, and p_u is node u's visit rate.

In networks with modular structure, we can compress the random walks' description by grouping nodes into more than one module such that a random walker tends to remain within modules and module switches become rare. This lets us re-use codewords across modules and design a codebook per module based on the nodes' module-normalised visit rates. However, sender and receiver need a way to encode module switches. The map equation uses a designated module exit codeword per module and an index-level codebook with module entry codewords. In a two-level partition, the sender communicates one codeword for intra-module random-walker steps to the receiver and three codewords for inter-module steps (figure 2). The lower limit for the codelength is given by the sum of entropies associated with module and index codebooks, weighted by their usage rates. Given a partition of the network's nodes into modules, M, the map equation [22] formalises this relationship,

$$L(\mathsf{M}) = q\mathcal{H}(Q) + \sum_{\mathsf{m}\in\mathsf{M}} p_{\mathsf{m}}\mathcal{H}(P_{\mathsf{m}}). \tag{2}$$

Here $q = \sum_{\mathsf{m}\in\mathsf{M}} q_{\mathsf{m}}$ is the index-level codebook usage rate, q_{m} is the entry rate for module m, and $Q = \{q_{\mathsf{m}} \mid \mathsf{m} \in \mathsf{M}\}$ is the set of module entry rates; m_{exit} is the exit rate for module m, $p_{\mathsf{m}} = \mathsf{m}_{\text{exit}} + \sum_{u\in\mathsf{m}} p_u$ is the codebook usage rate for module m, and $P_{\mathsf{m}} = \{\mathsf{m}_{\text{exit}}\} \cup \{p_u \mid u \in \mathsf{m}\}$ is the set of node visit rates in m, including m's module exit rate.

The map equation can detect communities in simple, weighted, directed, and higher-order networks. To make use of node metadata for detecting communities, we can either incorporate a corresponding term in the map equation [42], design metadata-informed flow models [43], or introduce a prior network and reinforce link weights between nodes with the same metadata label [44].

3 MapSim: node similarities from modular flow compression

Compression-based similarity measures consider pairs of objects more similar if they jointly compress better. Extending this idea to networks, we exploit the coding of network flows based on the map equation, and use it to calculate information-theoretic pairwise similarities between nodes: *MapSim*. We interpret a network's community structure as an implicit embedding and, roughly speaking, consider nodes in the same community as more similar than nodes in different communities.

To calculate node similarities, we begin with a network partition and its corresponding modular coding scheme[2], which can be visualised as a tree, annotated with the transition rates defined by

[2]In principle, arbitrary network partitions can be used, regardless of the used community detection method.

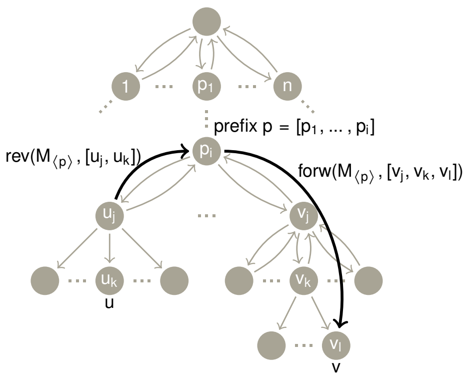

Figure 3: Illustration of map equation similarity between nodes u and v with addresses $\operatorname{addr}(\mathsf{M}, u) = [p_1, \ldots, p_i, u_j, u_k]$ and $\operatorname{addr}(\mathsf{M}, v) = [p_1, \ldots, p_i, v_j, v_k, v_l]$. M is the complete network partition. The longest common prefix between the addresses for u and v is $p = [p_1, \ldots, p_i]$, and $\mathsf{M}_{\langle p \rangle}$ is the sub-module at address p within M, that is the smallest module that contains u and v.

the link patterns in the network (figure 2). While the network's topology constrains random walks to transitions along existing links, the coding scheme is more flexible and can describe transitions between *any* pair of nodes. To describe the transition from node u to v, we find the corresponding path in the partition tree and multiply the transition rates along that path, that is, we use the *coarse-grained description* of the network's community structure, not the network's actual link pattern; it can describe any transition regardless of whether the link (u, v) exists in the network or not. The description length in bits for a path with transition rate r is $-\log_2(r)$. For example, consider the scenario in figure 2 where we calculate similarity scores for the two directed links $(5, 3)$ and $(5, 7)$, neither of which exists in the network. Nodes 5 and 3 are in module A, and the rate at which a random walker in A visits node 3 is $3/12$, requiring $-\log_2(3/12) = 2$ bits to describe that transition. Node 7 is in module B, and a random walker in A exits A at rate $1/12$, enters B at rate $1/2$, and then visits node 7 at rate $2/10$, that is, at rate $1/120$, requiring $-\log_2(1/120) \approx 6.9$ bits.

Paths to derive similarities emanate from modules, not from nodes, because the model must generalise to unobserved data. If compression was our sole purpose, we would use node-specific codebooks containing codewords for neighbouring nodes, no longer detect communities, and be able to describe merely observed links. Instead, the map equation's coding scheme is designed to capitalise on modular network structures: The modular code structure provides a model that generalises to unobserved data, coarse-grains the path descriptions, and prevents overfitting.

For the general case, where M can be a hierarchical network partition, we number the sub-modules within each module m from 1 to n_{m} – we refer to these numbers as addresses – such that an ordered sequence of addresses uniquely identifies a path starting at the root of the partition tree. We let $\operatorname{addr}\colon \mathsf{M} \times N \to \mathit{List}(\mathbb{N})$ be a function that takes a network partition and a node as input, and returns the node's address in the partition. To calculate the similarity of node v to u, we identify the longest common prefix p of the nodes' addresses, $\operatorname{addr}(\mathsf{M}, u)$ and $\operatorname{addr}(\mathsf{M}, v)$, and select the partition tree's sub-tree $\mathsf{M}_{\langle p \rangle}$ that corresponds to the prefix p: $\mathsf{M}_{\langle p \rangle}$ is the smallest sub-tree that contains u and v. We obtain the addresses for u and v within sub-tree $\mathsf{M}_{\langle p \rangle}$ by removing the prefix p from their addresses. That is, $\operatorname{addr}(\mathsf{M}, u) = p +\!\!+ \operatorname{addr}(\mathsf{M}_{\langle p \rangle}, u)$ and $\operatorname{addr}(\mathsf{M}, v) = p +\!\!+ \operatorname{addr}(\mathsf{M}_{\langle p \rangle}, v)$, where $+\!\!+$ is list concatenation. The rate at which a random walker transitions from u to v is the product of (i) the rate at which the random walker moves along the path $\operatorname{addr}(\mathsf{M}_{\langle p \rangle}, u)$ in *reverse direction*, $\operatorname{rev}(\mathsf{M}_{\langle p \rangle}, \operatorname{addr}(\mathsf{M}_{\langle p \rangle}, u))$, that is from u to the root of $M_{\langle p \rangle}$, and (ii) the rate at which the random walker moves along the path $\operatorname{addr}(\mathsf{M}_{\langle p \rangle}, v)$ in *forward direction*, $\operatorname{forw}(\mathsf{M}_{\langle p \rangle}, \operatorname{addr}(\mathsf{M}_{\langle p \rangle}, v))$, that is from the root of $\mathsf{M}_{\langle p \rangle}$ to v, where

$$\operatorname{rev}(\mathsf{M}, a) = \begin{cases} 1 & \text{if } a = [x] \\ \mathsf{M}_{\langle [x] \rangle, \text{exit}} \cdot \operatorname{rev}(\mathsf{M}_{\langle [x] \rangle}, a') & \text{if } a = [x] +\!\!+ a' \end{cases} \tag{3}$$

$$\operatorname{forw}(\mathsf{M}, a) = \begin{cases} p_{\langle [x] \rangle} / p_{\mathsf{M}} & \text{if } a = [x] \\ \mathsf{M}_{\langle [x] \rangle, \text{enter}} \cdot \operatorname{forw}(\mathsf{M}_{\langle [x] \rangle}, a') & \text{if } a = [x] +\!\!+ a' \end{cases} \tag{4}$$

and a' denotes non-empty sequences. Here p_{M} is the codebook use rate for module M and $p_{\langle [x] \rangle}$ is the visit rate for the node identified by address x within the given module. The final addresses in equation 3 and equation 4 are treated differently, reflecting that the map equation forgets the most recently visited node.

We illustrate these ideas in a generic example (figure 3). In short, we define map equation similarity,

$$\mathrm{MapSim}\,(M, u, v) = \mathrm{rev}(\mathsf{M}_{\langle p \rangle}, \mathrm{addr}(\mathsf{M}_{\langle p \rangle}, u)) \cdot \mathrm{forw}(\mathsf{M}_{\langle p \rangle}, \mathrm{addr}(\mathsf{M}_{\langle p \rangle}, v)), \tag{5}$$

where p is the longest common prefix shared by the addresses of u and v in the partition tree defined by M. To express map equation similarity in terms of description length, we take the $-\log_2$ of MapSim and regard pairs of nodes that yield a shorter description length as more similar.

MapSim is asymmetric since module entry and exit rates are, in general, different and u and v can have different visit rates. MapSim is zero if one node is in a disconnected component; the exit rate for regions without out-links is zero, so the corresponding description length is infinitely long. This issue can be addressed with the regularised map equation [44], which addresses networks with incomplete observations using weak links between all pairs of nodes to connect all modules with modelled link strengths that depend on the connection patterns of each node.

We calculate node similarities in three steps: (i) inferring a network's community with Infomap [45], a greedy, search-based optimisation algorithm for the map equation, (ii) representing the corresponding coding scheme in a suitable data structure, and (iii) using *MapSim* to computing similarities based on the coding scheme. The overall approach is illustrated in figs. 1 – 3 and algorithm 1.

Algorithm 1: Pseudo-code of function MapSim calculating similarity score for node pair (u, v).

```
Input  : graph $G$ and pair of nodes $(u, v)$
Output : similarity score of $(u, v)$
// Use InfoMap to construct coding tree for compression
modules = infomap.$minimiseMapEquation(G)$
tree = buildPartitionTree($G$, modules)
p = longestCommonPrefix(tree, $u$, $v$)
tree$_{\langle p \rangle}$ = smallestSubtree(tree, p)
// calculate code length of random walks from $u$ to $v$
addrU = addr(tree$_{\langle p \rangle}$, $u$)
addrV = addr(tree$_{\langle p \rangle}$, $v$)
revRate = rev(tree$_{\langle p \rangle}$, addrU)
fwdRate = forw(tree$_{\langle p \rangle}$, addrV)
return $-\log_2$(revRate $\cdot$ fwdRate)
```

4 Experimental Validation

We evaluate the performance of *MapSim* in unsupervised, similarity-based link prediction for 47 real-world networks, 35 directed (table 1) and 12 undirected (table 3), retrieved from Netzschleuder [46] and Konect [47]. Details of the directed and undirected networks are shown in tables 2 and 4, respectively. Our analysis is based on a Python-implementation available on GitHub[3], building on Infomap, a fast and greedy search algorithm for minimising the map equation with an open source implementation in C++ [45, 48]. As baseline, we use four random walk-based embedding methods: DeepWalk [6], node2vec [7], LINE [8], and NERD [15], using the respective author's implementation. We also include results for *MapSim* based on the one-module partition for comparison. Adopting the argument by [7], we exclude graph factorisation methods and simple local similarity scores because they have already been shown to be inferior to node2vec. We include NERD because it is a recent random walk-based embedding method proposed for directed link prediction with higher reported performance than other walk-based embeddings [15].

4.1 Unsupervised Link Prediction

Different from works that use graph embeddings for *supervised* link prediction, we address *unsupervised* link prediction. Like Goyal and Ferrara [2] and Khosla et al. [15], we take a similarity-based approach that does not require training a classifier. We compute similarity scores based on node embeddings, rather than applying a supervised classifier to features computed for node pairs. We

[3]https://github.com/mapequation/map-equation-similarity

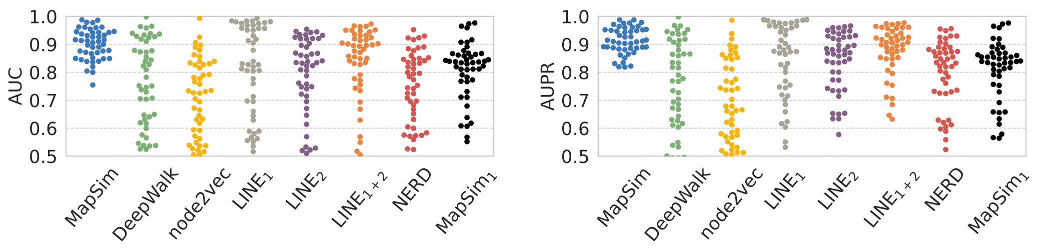

Figure 4: Link-prediction performance of MapSim, DeepWalk, node2vec, LINE$_1$, LINE$_2$, LINE$_{1+2}$, and NERD on 47 real-world networks. **Left:** AUC performance. **Right:** AUPR performance.

adopt the approach by Khosla et al. [15] and calculate node similarities as the sigmoid over the feature vectors' dot product.

Considering how different embedding techniques generalise across data sets, *we purposefully refrained from hyperparameter tuning*. We chose a single set of hyperparameters for each method, informed by the default parameters given by the respective authors and recent surveys' discussion regarding which hyperparameter values generally provide good link prediction performance. For DeepWalk and node2vec, we sample $r = 80$ random walks of length $l = 40$ per node, and use a window size of $w = 10$. For both methods, the underlying word embedding is applied using the default model parameters fixed by the authors, $skipgram = 1$, $k = 10$ and $mincount = 0$. For node2vec we set the return parameter to $p = 1$. Since for $q = p = 1$ node2vec is identical to DeepWalk, we use $q = 4$, which was found to provide good performance for link prediction [2]. We run LINE with first-order (LINE$_1$), second-order (LINE$_2$), and combined first-and-second-order proximity (LINE$_{1+2}$), use $1{,}000$ samples per node, and $s = 5$ negative samples. For NERD, we use 800 samples and $\kappa = 3$ negative samples per node. We set the number of neighbourhood nodes to $n = 1$, as suggested by the authors for link prediction. We use $d = 128$ dimensions for all embeddings. Since *MapSim* is a non-parametric method, it does not require setting any hyperparameters. However, to avoid local optima when heuristically minimising the map equation, we run Infomap 100 times and select the partition with the shortest description length.

We use 5-fold cross-validation to split links into train and test sets, treating weighted links as indivisible. We calculate the node embedding (for *MapSim* the coding tree) in the training network, derive predictions based on node similarities, and evaluate them based on the links in the validation set. For each fold, we restrict the resulting training network to its largest (weakly) connected component. For a validation set with k positive links, we sample k negative links uniformly at random, and calculate scores for all $2k$ links. In undirected networks, for each positive link (u, v), we also consider (v, u) as positive, and, therefore, sample two negative links per positive link. Varying the discrimination threshold, we obtain a receiver operator characteristic (ROC) per fold, and calculate the area under the curve (AUC). Detailed results, including average and worst-case performance, are shown in tables 2 and 4; we also report precision-recall performance (table 5). We include *MapSim* based on the one-module partition[4] in the results and note that it performs better than using a modular partition in some cases: this suggests that the network does not have a strong community structure, which could be addressed with the regularised map equation [44]. When mentioning *MapSim* in the following, we refer to using modular partitions.

On average, *MapSim* outperforms all baseline methods across the 47 data sets in terms of AUC and AUPR (figure 4); for detailed results on a per-network bases see appendix. Using a one-sided two-sample t-test, we find that *MapSim*'s average performance across all networks is significantly higher than that of the best graph embedding method, LINE$_{1+2}$, both in directed and undirected networks ($p \approx 0.008$ and $p \approx 0.039$, respectively). *MapSim* provides the best performance in 11 of the 47 networks, with a standard deviation of the AUC score less than half of that of the best embedding-based method (LINE$_{1+2}$). For undirected networks, *MapSim* achieves the best performance for five of the 12 networks, while none of the embedding methods beats *MapSim's* performance in more than two networks. We find the largest performance gain in the directed network *linux*, where *MapSim* yields an increase of AUC of approximately 22.6% compared to the best embedding (NERD). *MapSim*'s worst-case performance across all networks is approximately 44% and 33% above that of the best-performing embedding for directed and undirected networks,

[4]With the one-module partition, MapSim becomes equivalent to preferential attachment.

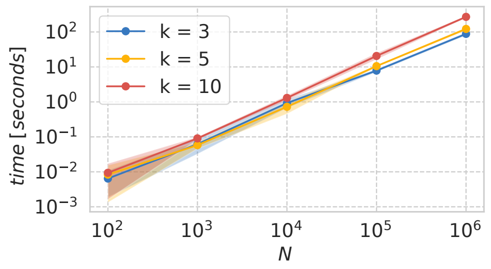

Figure 5: Runtime behavior for inferring the community structure with Infomap and constructing the coding tree for MapSim in synthetic k-regular networks with different size.

respectively. *MapSim*' performance advantage can be as high as 84%, for example $AUC = 0.988$ of MapSim in *foursquare-friendships-new* vs. $AUC = 0.537$ for node2vec. While node2vec performs best in the largest directed network, *MapSim* performs best in the largest undirected network and in several small networks, suggesting that *MapSim* works well both for small and large networks.

We attribute those encouraging results to multiple features of our method: Different from graph embedding techniques that require downstream dimensionality reduction, *MapSim*'s compression approach implicitly includes model selection and avoids overfitting. Moreover, the representation of nodes in the coding tree is integrated with the optimisation of hierarchical community structures in the network. Due to its non-parametric approach and the use of the analytical map equation, *MapSim* performs well in absence of tuning to the specific data set.

4.2 Scalability Analysis

We analyse *MapSim*'s scalability in synthetically generated networks with modular structure and tunable size and link density. We generate k-regular random graphs with N nodes and (mean) degree k. To avoid trivial configurations where a modular structure is absent, we create a network by first generating two k-regular random graphs with $\frac{N}{2}$ nodes each and "cross" two links, one from each of the two graphs, to obtain a single connected network with strong community structure. We then apply Infomap to (i) minimise the map equation and extract the network's modular structure, and (ii) construct the coding tree for calculating node similarities. We repeat this 10 times for random networks with different numbers of N nodes and degrees k. The average run times are reported in figure 5, which shows that, for sparse networks, the runtime of *MapSim* is linear in the size of the network. Edler et al. [48] report that the theoretical asymptotic bound of computational complexity for the optimisation of the map equation is in $\mathcal{O}(NlogN)$, which is the same as for vector space embedding techniques like node2vec and DeepWalk[5]. Thus, *MapSim* does not entail higher computational complexity compared to popular graph embeddings. This makes it an interesting choice for practitioners looking for a simple and scalable method that works well in small, large, directed, and undirected networks.

5 Conclusion and Outlook

We propose *MapSim*, a novel information-theoretic approach to compute node similarities based on a modular compression of network flows. Different from vector space embeddings, *MapSim* represents nodes in a discrete, non-metric space of communities that yields *asymmetric* similarities suitable to predict links in *directed* and *undirected* networks. The resulting similarities can be explained based on a network's modular structure. Using description length minimisation, *MapSim* naturally accounts for Occam's razor, which avoids overfitting and yields a parsimonious coding tree. Addressing unsupervised link prediction, we compare *MapSim* to popular embedding-based algorithms on 47 data sets that cover networks from a few hundred to hundreds of thousand nodes and millions of edges. Our analysis shows that the average performance of *MapSim* is more than 7% higher than its closest competitor, outperforming all competing methods in 11 of the 47 networks. Taking a new perspective on graph representation learning, our work demonstrates the potential of compression-based methods, with promising applications in other graph learning tasks. Moreover, recent generalisations of the map equation to temporal and higher-order networks [48] suggest that our method can be applied to graphs with non-dyadic or time-stamped relationships.

[5] [2, 3] report linear complexity, which we could not confirm in the literature.

Acknowledgements

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. Ingo Scholtes acknowledges financial support from the Swiss National Science Foundation through grant No. 176938. Martin Rosvall was supported by the Swedish Research Council, Grant No. 2016-00796.

References

[1] Thomas Hofmann, Bernhard Schölkopf, and Alexander J Smola. Kernel methods in machine learning. *The annals of statistics*, 36(3):1171–1220, 2008. 1

[2] Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78–94, 2018. ISSN 0950-7051. 1, 2, 3, 4, 7, 8, 9

[3] Nino Arsov and Georgina Mirceva. Network embedding: An overview. *arXiv preprint arXiv:1911.11726*, 2019. 1, 2, 3, 9

[4] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *Proc. of the Eleventh ACM Intl. Conf. on Web Search and Data Mining*, WSDM '18, page 459–467, New York, 2018. ACM. ISBN 9781450355810. 1, 2, 4

[5] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016. 1, 2

[6] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM, 2014. 1, 3, 4, 7

[7] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864. ACM, 2016. 2, 3, 4, 7

[8] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pages 1067–1077, 2015. 1, 3, 7

[9] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031, 2007. 2, 3

[10] Ryan N. Lichtenwalter, Jake T. Lussier, and Nitesh V. Chawla. New perspectives and methods in link prediction. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, page 243–252, New York, NY, USA, 2010. ACM. ISBN 9781450300551. 2, 4

[11] Zachary Stanfield, Mustafa Coşkun, and Mehmet Koyutürk. Drug response prediction as a link prediction problem. *Scientific reports*, 7(1):1–13, 2017. 2

[12] Mohammad Al Hasan and Mohammed J Zaki. A survey of link prediction in social networks. In *Social network data analytics*, pages 243–275. Springer, 2011. 2

[13] Hsinchun Chen, Xin Li, and Zan Huang. Link prediction approach to collaborative filtering. In *Proceedings of the 5th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL'05)*, pages 141–142. IEEE, 2005. 2

[14] Mengjia Xu. Understanding graph embedding methods and their applications. *SIAM Review*, 63(4):825–853, 2021. 2, 3

[15] Megha Khosla, Jurek Leonhardt, Wolfgang Nejdl, and Avishek Anand. Node representation learning for directed graphs. In *Machine Learning and Knowledge Discovery in Databases*, pages 395–411, Cham, 2020. Springer. ISBN 978-3-030-46150-8. 2, 4, 7, 8

[16] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 5171–5181, Red Hook, NY, USA, 2018. Curran Associates Inc. 2

[17] Seongjun Yun, Seoyoon Kim, Junhyun Lee, Jaewoo Kang, and Hyunwoo J. Kim. Neo-GNNs: Neighborhood overlap-aware graph neural networks for link prediction. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. 2

[18] Zhaocheng Zhu, Zuobai Zhang, Louis-Pascal Xhonneux, and Jian Tang. Neural bellman-ford networks: A general graph neural network framework for link prediction. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 29476–29490. Curran Associates, Inc., 2021. 2

[19] C. Seshadhri, Aneesh Sharma, Andrew Stolman, and Ashish Goel. The impossibility of low-rank representations for triangle-rich complex networks. *Proceedings of the National Academy of Sciences*, 117(11):5631–5637, 2020. ISSN 0027-8424. 2, 4

[20] Ali Faqeeh, Saeed Osat, and Filippo Radicchi. Characterizing the analogy between hyperbolic embedding and community structure of complex networks. *Phys. Rev. Lett.*, 121:098301, Aug 2018. 2, 4

[21] Yi-Jiao Zhang, Kai-Cheng Yang, and Filippo Radicchi. Systematic comparison of graph embedding methods in practical tasks. *Phys. Rev. E*, 104, 2021. ISSN 2470-0053. 2, 4

[22] Martin Rosvall and Carl T Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the national academy of sciences*, 105(4):1118–1123, 2008. 2, 3, 4, 5

[23] Linyuan Lü and Tao Zhou. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications*, 390(6):1150 – 1170, 2011. ISSN 0378-4371. 3

[24] Víctor Martínez, Fernando Berzal, and Juan-Carlos Cubero. A survey of link prediction in complex networks. *ACM Computing Surveys*, 49(4), 2016. 3, 4

[25] Lada A Adamic and Eytan Adar. Friends and neighbors on the web. *Social networks*, 25(3): 211–230, 2003. 3

[26] Zhenjiang Lin, Michael R Lyu, and Irwin King. Pagesim: a novel link-based measure of web page similarity. In *Proceedings of the 15th international conference on World Wide Web*, pages 1019–1020, 2006. 3

[27] Weiping Liu and Linyuan Lü. Link prediction based on local random walk. *EPL (Europhysics Letters)*, 89(5):58007, mar 2010.

[28] Joydeep Chandra, Ingo Scholtes, Niloy Ganguly, and Frank Schweitzer. A tunable mechanism for identifying trusted nodes in large scale distributed networks. In *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*, pages 722–729. IEEE, 2012. 3

[29] Matthew Brand and Kun Huang. A unifying theorem for spectral embedding and clustering. In *International Workshop on Artificial Intelligence and Statistics*, pages 41–48. PMLR, 2003. 3

[30] Naoki Masuda, Mason A. Porter, and Renaud Lambiotte. Random walks and diffusion on networks. *Physics Reports*, 716-717:1–58, 2017. ISSN 0370-1573. 3

[31] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. *Advances in neural information processing systems*, 14, 2001. 3

[32] Cheng Yang and Zhiyuan Liu. Comprehend deepwalk as matrix factorization. *arXiv preprint arXiv:1501.00358*, 2015. 3

[33] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013. 3

[34] Bryan Perozzi, Vivek Kulkarni, Haochen Chen, and Steven Skiena. Don't walk, skip! online learning of multi-scale network embeddings. In *Proc. of the 2017 IEEE/ACM Intl. Conf. on Advances in Social Networks Analysis and Mining 2017*, ASONAM '17, page 258–265, New York, NY, USA, 2017. ACM. ISBN 9781450349932. 4

[35] Sudhanshu Chanpuriya, Cameron Musco, Konstantinos Sotiropoulos, and Charalampos E. Tsourakakis. Node embeddings and exact low-rank representations of complex networks, 2020. 4

[36] Thanh Tam Nguyen and Chi Thang Duong. A comparison of network embedding approaches. Technical report, School of Computer and Communication Sciences, EPFL, Technical Report, 2018. 4

[37] Xin Sun, Zenghui Song, Yongbo Yu, Junyu Dong, Claudia Plant, and Christian Böhm. Network embedding via deep prediction model. *arXiv preprint arXiv:2104.13323*, 2021. 4

[38] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. Community preserving network embedding. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 203–209. AAAI Press, 2017. 4

[39] Sandro Cavallari, Vincent W. Zheng, Hongyun Cai, Kevin Chen-Chuan Chang, and Erik Cambria. Learning community embedding with community detection and node embedding on graphs. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, page 377–386, New York, NY, USA, 2017. ACM. ISBN 9781450349185. 4

[40] Martin Rosvall and Carl T Bergstrom. Multilevel compression of random walks on networks reveals hierarchical organization in large integrated systems. *PloS one*, 6(4):e18209, 2011. 4

[41] Claude Elwood Shannon. A mathematical theory of communication. *Bell Labs Tech. J.*, 27(3): 379–423, 7 1948. 4

[42] Scott Emmons and Peter J. Mucha. Map equation with metadata: Varying the role of attributes in community detection. *Phys. Rev. E*, 100:022301, Aug 2019. doi: 10.1103/PhysRevE.100.022301. 5

[43] Aleix Bassolas, Anton Eriksson, Antoine Marot, Martin Rosvall, and Vincenzo Nicosia. Metadata-informed community detection with lazy encoding using absorbing random walks. *arXiv preprint arXiv:2111.05158*, 2021. 5

[44] Jelena Smiljanić, Christopher Blöcker, Daniel Edler, and Martin Rosvall. Mapping flows on weighted and directed networks with incomplete observations. *Journal of Complex Networks*, 9 (6), 12 2021. ISSN 2051-1329. 5, 7, 8

[45] D. Edler, A. Eriksson, and M. Rosvall. The infomap software package. `https://www.mapequation.org`, 2020. 7

[46] Tiago P. Peixoto. The netzschleuder network catalogue and repository. `https://networks.skewed.de/`, 2020. 7

[47] Jérôme Kunegis. Konect: The koblenz network collection. In *Proceedings of the 22nd International Conference on World Wide Web*, WWW '13 Companion, page 1343–1350, New York, NY, USA, 2013. ACM. ISBN 9781450320382. 7, 15, 17

[48] Daniel Edler, Ludvig Bohlin, and Martin Rosvall. Mapping Higher-Order Network Flows in Memory and Multilayer Networks with Infomap. *Algorithms*, 10:112, 2017. 7, 9

[49] R. Guimerà, L. Danon, A. Díaz-Guilera, F. Giralt, and A. Arenas. Self-similar community structure in a network of human interactions. *Phys. Rev. E*, 68:065103, Dec 2003. 15

[50] Lada A. Adamic and Natalie Glance. The political blogosphere and the 2004 u.s. election: Divided they blog. In *Proceedings of the 3rd International Workshop on Link Discovery*, LinkKDD '05, New York, 2005. ACM. ISBN 1595932151. 15

[51] Ulrich Stelzl, Uwe Worm, Maciej Lalowski, Christian Haenig, Felix H Brembeck, Heike Goehler, Martin Stroedicke, Martina Zenkner, Anke Schoenherr, Susanne Koeppen, et al. A human protein-protein interaction network: a resource for annotating the proteome. *Cell*, 122 (6):957–968, 2005. 15

[52] Rob M Ewing, Peter Chu, Fred Elisma, Hongyan Li, Paul Taylor, Shane Climie, Linda McBroom-Cerajewski, Mark D Robinson, Liam O'Connor, Michael Li, et al. Large-scale mapping of human protein–protein interactions by mass spectrometry. *Molecular systems biology*, 3(1):89, 2007. 15

[53] Bureau of Transportation Statistics. T-100 domestic market. `https://www.transtats.bts.gov/DL_SelectFields.asp?Table_ID=310`, 2017. 15

[54] Ron Milo, Shalev Itzkovitz, Nadav Kashtan, Reuven Levitt, Shai Shen-Orr, Inbal Ayzenshtat, Michal Sheffer, and Uri Alon. Superfamilies of evolved and designed networks. *Science*, 303 (5663):1538–1542, 2004. 15

[55] The openflights.org website. `https://openflights.org/data.html`, 2022. 15

[56] Paolo Massa, Martino Salvetti, and Danilo Tomasoni. Bowling alone and trust decline in social network sites. In *2009 Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing*, pages 658–663, 2009. 15

[57] Michael Ley. The dblp computer science bibliography: Evolution, research issues, perspectives. In *String Processing and Information Retrieval*, pages 1–10, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg. ISBN 978-3-540-45735-0. 15

[58] Michael Fire, Rami Puzis, and Yuval Elovici. *Link Prediction in Highly Fractional Data Sets*, pages 283–300. Springer New York, New York, NY, 2013. ISBN 978-1-4614-5311-6. 15

[59] B. Stabler. Transportation network test problems. `https://github.com/bstabler/TransportationNetworks`, 2022. 15

[60] M. Zaversnik V. Batagelj, A. Orvar. Network analysis of texts. *Language Technologies*, pages 143–148, 2002. 15

[61] Gergely Palla, Illés J Farkas, Péter Pollner, Imre Derényi, and Tamás Vicsek. Directed network modules. *New Journal of Physics*, 9(6):186–186, jun 2007. 15

[62] George R Kiss, Christine Armstrong, Robert Milroy, and James Piper. An associative thesaurus of english and its computer analysis. *The computer and literary studies*, pages 153–165, 1973. 15

[63] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163, 2000. 15

[64] Johannes Gehrke, Paul Ginsparg, and Jon Kleinberg. Overview of the 2003 kdd cup. *SIGKDD Explor. Newsl.*, 5(2):149–151, dec 2003. ISSN 1931-0145. 15

[65] Munmun De Choudhury, Hari Sundaram, Ajita John, and Dorée Duncan Seligmann. Social synchrony: Predicting mimicry of user actions in online social media. In *2009 International Conference on Computational Science and Engineering*, volume 4, pages 151–158, 2009. 15

[66] Bryan Klimt and Yiming Yang. The enron corpus: A new dataset for email classification research. In *Machine Learning: ECML 2004*, pages 217–226, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. ISBN 978-3-540-30115-8. 15

[67] Furkan Gursoy and Dilek Gunnec. Influence maximization in social networks under deterministic linear threshold model. *Knowledge-Based Systems*, 161:111–123, 2018. ISSN 0950-7051. 15

[68] Oliver Richters and Tiago P Peixoto. Trust transitivity in social networks. *PloS one*, 6(4): e18384, 2011. 15

[69] Bimal Viswanath, Alan Mislove, Meeyoung Cha, and Krishna P. Gummadi. On the evolution of user interaction in facebook. In *Proceedings of the 2nd ACM Workshop on Online Social Networks*, page 37–42, New York, 2009. ACM. ISBN 9781605584454. 15

[70] Vicenç Gómez, Andreas Kaltenbrunner, and Vicente López. Statistical analysis of the social network and discussion threads in slashdot. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, page 645–654, New York, NY, USA, 2008. ACM. ISBN 9781605580852. 15

[71] Kevin Gullikson. Python dependency analysis. `http://kgullikson88.github.io/blog/pypi-analysis.html`, 2016. 15

[72] Matthew Richardson, Rakesh Agrawal, and Pedro Domingos. Trust management for the semantic web. In *The Semantic Web - ISWC 2003*, pages 351–368, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg. ISBN 978-3-540-39718-2. 15

[73] Michael Fire, Lena Tenenboim-Chekina, Rami Puzis, Ofrit Lesser, Lior Rokach, and Yuval Elovici. Computationally efficient link prediction in a variety of social networks. *ACM Trans. Intell. Syst. Technol.*, 5(1), jan 2014. ISSN 2157-6904. 15, 17

[74] Manlio De Domenico, Antonio Lima, Paul Mougel, and Mirco Musolesi. The anatomy of a scientific rumor. *Scientific reports*, 3(1):1–9, 2013. 15

[75] Jure Leskovec, Lada A. Adamic, and Bernardo A. Huberman. The dynamics of viral marketing. *ACM Trans. Web*, 1(1):5–es, may 2007. ISSN 1559-1131. 15

[76] Réka Albert, Hawoong Jeong, and Albert-László Barabási. Diameter of the world-wide web. *nature*, 401(6749):130–131, 1999. 15

[77] Munmun De Choudhury. Discovery of information disseminators and receptors on online social media. In *Proceedings of the 21st ACM Conference on Hypertext and Hypermedia*, page 279–280, New York, 2010. ACM. ISBN 9781450300414. 15

[78] Samin Aref, David Friggens, and Shaun Hendy. Analysing scientific collaborations of new zealand institutions using scopus bibliometric data. In *Proceedings of the Australasian Computer Science Week Multiconference*, ACSW '18, New York, NY, USA, 2018. ACM. ISBN 9781450354363. 17

[79] Paolo Crucitti, Vito Latora, and Sergio Porta. Centrality measures in spatial networks of urban streets. *Phys. Rev. E*, 73:036125, Mar 2006. 17

[80] Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393(6684):440–442, 1998. 17

[81] Michael Fire, Rami Puzis, and Yuval Elovici. Organization mining using online social networks. *CoRR*, abs/1303.3741, 2013. 17

[82] G. Joshi-Tope, M. Gillespie, I. Vastrik, P. D'Eustachio, E. Schmidt, B. de Bono, B. Jassal, G.R. Gopinath, G.R. Wu, L. Matthews, S. Lewis, E. Birney, and L. Stein. Reactome: a knowledgebase of biological pathways. *Nucleic Acids Research*, 33(suppl_1):D428–D432, 01 2005. ISSN 0305-1048. 17

[83] Manlio De Domenico, Andrea Lancichinetti, Alex Arenas, and Martin Rosvall. Identifying modular flows on multilayer networks reveals highly overlapping organization in interconnected systems. *Phys. Rev. X*, 5:011027, Mar 2015. 17

[84] R. Alberich, J. Miro-Julia, and F. Rossello. Marvel universe looks almost like a real social network, 2002. 17

[85] Brian Karrer, M. E. J. Newman, and Lenka Zdeborová. Percolation on sparse networks. *Phys. Rev. Lett.*, 113:208702, Nov 2014. 17

[86] Dingqi Yang, Daqing Zhang, Longbiao Chen, and Bingqing Qu. Nationtelescope: Monitoring and visualizing large-scale collective behavior in lbsns. *Journal of Network and Computer Applications*, 55:170–180, 2015. ISSN 1084-8045. 17

A Appendix

Table 1: Properties of 35 directed networks, where weighted networks are marked with W, temporal link counts before aggregation into a static network are marked with $*$, and ρ is link reciprocity.

Data	Ref	Nodes	Edges	ρ
uni-email	[49]	1,133	10,903	1.000
polblogs	[50]	1,490	19,090	0.243
interactome-stelzl	[51]	1,706	6,207	0.972
interactome-figeys	[52]	2,239	6,452	0.006
us-air-traffic^{W}	[53]	2,278	*6,390,340	0.757
word-adjacency-japanese	[54]	2,704	8,300	0.073
openflights^{W}	[55]	3,214	66,771	0.978
jdk	[47]	6,434	150,985	0.009
advogato^{W}	[56]	6,541	51,127	0.307
word-adjacency-spanish	[54]	11,586	45,129	0.091
dblp-cite	[57]	12,590	49,759	0.004
anybeat	[58]	12,645	67,053	0.535
chicago-road	[59]	12,982	39,018	0.943
foldoc^{W}	[60]	13,356	120,238	0.479
google	[61]	15,763	171,206	0.254
word-assoc^{W}	[62]	23,132	312,342	0.094
cora	[63]	23,166	91,500	0.051
arxiv-citation-HepTh	[64]	27,770	352,807	0.003
digg-reply^{W}	[65]	30,398	*87,627	0.002
linux	[47]	30,837	213,954	0.002
arxiv-citation-HepPh	[64]	34,546	421,578	0.003
email-enron	[66]	36,692	367,662	1.000
inploid	[67]	39,749	57,276	0.272
pgp-strong	[68]	39,796	301,498	0.660
facebook-wall^{W}	[69]	46,952	*876,993	0.588
slashdot-threads^{W}	[70]	51,083	*140,778	0.210
python-dependency	[71]	58,743	108,399	0.004
lkml-reply^{W}	[47]	63,399	*1,096,440	0.635
epinions-trust	[72]	75,888	508,837	0.405
prosper	[47]	89,269	3,394,979	< 0.001
google-plus	[73]	211,187	1,506,896	0.482
twitter-higgs-retweet^{W}	[74]	256,491	328,132	0.005
amazon-copurchases-302	[75]	262,111	1,234,877	0.543
notre-dame-web	[76]	325,729	1,497,134	0.507
twitter-followers	[77]	465,017	834,797	0.003

Table 2: ROC AUC for link prediction in 35 directed networks for DeepWalk (DW), node2vec (n2v), LINE$_1$ (L$_1$), LINE$_2$ (L$_2$), LINE$_{1+2}$ (L$_{1+2}$), NERD, *MapSim* based on the one-level partition (MapSim$_1$), and *MapSim* based on modular partitions. Networks marked with W are weighted. $\dagger$ marks cases with AUC < 0.5 where we flipped the predicted link scores for AUC > 0.5. The best results per network are shown in bold, second-best underlined, and then rounded.

Data	DW	n2v	L$_1$	L$_2$	L$_{1+2}$	NERD	MS$_1$	MS
uni-email	0.911	†0.505	**0.957**	0.903	$\underline{0.932}$	0.667	0.711	0.852
polblogs	0.705	0.695	0.804	0.823	0.841	0.652	$\underline{0.868}$	**0.914**
interactome-stelzl	0.810	†0.505	**0.913**	0.758	$\underline{0.849}$	0.524	0.710	0.755
interactome-figeys	0.524	†0.828	†**0.905**	0.529	$^\dagger\underline{0.850}$	0.605	0.773	0.839
us-air-traffic^W	0.649	0.572	0.563	**0.935**	$\underline{0.933}$	0.774	0.858	0.916
word-adjacency-japanese	†0.538	†0.645	†0.580	0.748	0.743	0.526	**0.811**	$\underline{0.800}$
openflights^W	0.782	†0.665	0.918	0.934	**0.948**	0.708	0.838	$\underline{0.941}$
jdk	0.746	0.857	0.820	0.695	0.755	0.725	$\underline{0.974}$	**0.986**
advogato^W	0.738	0.563	0.806	0.865	**0.883**	0.742	0.812	$\underline{0.878}$
word-adjacency-spanish	†0.538	0.672	†0.713	**0.824**	0.791	0.632	$\underline{0.811}$	0.805
dblp-cite	0.840	†0.537	†0.589	0.646	0.549	$\underline{0.877}$	0.823	**0.890**
anybeat	0.647	0.539	0.644	0.841	**0.857**	0.683	0.834	$\underline{0.850}$
chicago-road	**0.998**	0.816	$\underline{0.981}$	0.670	0.835	†0.583	†0.608	0.848
foldoc^W	$\underline{0.927}$	0.549	**0.951**	0.832	0.905	0.571	0.618	0.845
google	0.844	0.792	0.831	0.868	$\underline{0.896}$	0.697	0.867	**0.962**
word-assoc^W	0.729	0.830	0.813	0.869	**0.916**	$\underline{0.884}$	0.837	0.849
cora	$\underline{0.939}$	0.839	**0.950**	0.761	0.831	0.830	0.839	0.906
arxiv-citation-HepTh	0.878	0.839	**0.958**	0.857	0.901	0.850	0.842	$\underline{0.942}$
digg-reply^W	†0.546	0.618	†0.552	0.714	0.693	$\underline{0.841}$	**0.845**	0.836
linux	0.704	0.726	0.567	0.722	0.734	0.784	$\underline{0.959}$	**0.961**
arxiv-citation-HepPh	$\underline{0.959}$	0.897	**0.975**	0.835	0.898	0.860	0.830	0.942
email-enron	0.823	†0.594	**0.983**	0.946	$\underline{0.963}$	0.819	0.840	0.931
inploid	0.631	0.766	0.516	0.838	0.828	0.753	$\underline{0.845}$	**0.870**
pgp-strong	0.873	0.527	**0.984**	0.890	0.924	0.795	0.782	$\underline{0.925}$
facebook-wall^W	$\underline{0.877}$	0.789	**0.931**	0.809	0.855	0.813	0.768	0.867
slashdot-threads^W	0.565	0.781	0.629	0.748	0.771	0.796	**0.877**	$\underline{0.876}$
python-dependency	0.751	0.735	†0.556	0.520	†0.505	0.832	**0.965**	$\underline{0.913}$
lkml-reply^W	0.537	0.731	0.590	**0.945**	$\underline{0.944}$	0.724	0.908	0.933
epinions-trust	0.599	0.777	0.806	$\underline{0.943}$	**0.952**	0.887	0.916	0.937
prosper	0.828	0.631	0.697	†0.614	†0.518	**0.952**	0.891	$\underline{0.945}$
google-plus	0.752	0.725	**0.957**	0.787	0.893	0.891	0.862	$\underline{0.946}$
twitter-higgs-retweet^W	0.620	$\underline{0.879}$	†0.695	†0.522	†0.569	0.799	**0.977**	0.820
amazon-copurchases-302	$\underline{0.963}$	0.826	**0.980**	0.896	0.936	0.575	0.638	0.910
notre-dame-web	$\underline{0.965}$	0.926	**0.975**	0.919	0.964	0.923	0.867	0.962
twitter-followers	0.526	†**0.993**	$^\dagger\underline{0.993}$	0.510	†0.973	0.917	0.809	0.871
Average	0.750	0.719	0.802	0.786	$\underline{0.832}$	0.757	0.830	**0.892**
Worst	0.524	0.505	0.516	0.510	0.505	0.524	$\underline{0.608}$	**0.755**
Standard Deviation	0.148	0.131	0.164	0.129	0.128	0.118	$\underline{0.088}$	**0.054**

Table 3: Properties of 12 undirected networks, where weighted networks are marked with W.

Data	Ref	Nodes	Edges
new-zealand-collab^W	[78]	1,511	4,273
urban-streets-venice	[79]	1,840	2,407
urban-streets-ahmedabad	[79]	2,870	4,387
power	[80]	4,941	6,594
facebook-organizations-L1	[81]	5,793	45,266
reactome	[82]	6,327	147,547
physics-collab-arXiv^W	[83]	14,488	59,026
marvel-universe	[84]	19,428	95,497
internet-as	[85]	22,963	48,436
marker-cafe	[73]	69,413	1,644,849
livemocha	[47]	104,103	2,193,083
foursquare-friendships-new	[86]	114,324	607,333

Table 4: ROC AUC on 12 undirected networks for DeepWalk (DW), node2vec (n2v), LINE$_1$ (L_1), LINE$_2$ (L_2), LINE$_{1+2}$ (L_{1+2}), NERD, *MapSim* based on the one-level partition (MS$_1$), and *MapSim* based on modular partitions (MS). Networks marked with W are weighted. † marks cases with AUC < 0.5 where we flipped the predicted link scores for AUC > 0.5. The best results per network are shown in bold, second-best underlined, and then rounded.

Data	DW	n2v	L_1	L_2	L_{1+2}	NERD	MS$_1$	MS
new-zealand-collab^W	0.616	0.734	†0.660	**0.921**	0.895	†0.559	0.834	0.839
urban-streets-venice	0.872	0.834	0.777	0.570	0.668	0.573	†0.607	**0.889**
urban-streets-ahmedabad	0.939	0.890	0.828	†0.533	0.629	†0.575	†0.731	0.897
power	0.919	0.863	0.827	0.741	0.777	0.600	0.552	**0.959**
facebook-organizations-L1	0.937	0.516	0.968	0.954	0.966	0.846	0.864	**0.979**
reactome	0.934	0.592	**0.983**	0.925	0.950	0.846	0.820	0.978
physics-collab-arXiv^W	0.929	0.521	**0.977**	0.807	0.871	0.695	0.568	0.955
marvel-universe	0.854	†0.633	0.879	0.834	**0.902**	0.852	0.679	0.900
internet-as	0.641	†0.705	0.535	0.921	0.920	0.744	0.766	**0.927**
marker-cafe	0.576	0.906	0.760	0.920	0.914	**0.930**	0.907	0.916
livemocha	0.708	0.758	0.839	0.861	0.876	**0.924**	0.855	0.876
foursquare-friendships-new	0.924	0.537	0.968	0.932	0.950	0.836	0.791	**0.988**
Average	0.821	0.707	0.834	0.826	0.860	0.748	0.748	**0.925**
Worst	0.576	0.521	0.535	0.533	0.629	0.559	0.552	**0.839**
Standard Deviation	0.136	0.140	0.132	0.137	0.106	0.136	0.116	**0.045**

Table 5: Average precision on 47 directed and undirected networks for DeepWalk (DW), node2vec (n2v), LINE$_1$ (L$_1$), LINE$_2$ (L$_2$), LINE$_{1+2}$ (L$_{1+2}$), NERD, *MapSim* based on the one-level partition (MapSim$_1$), and *MapSim* based on modular partitions. Weighted networks are marked with W. Results marked with † correspond to cases with AUC < 0.5 where we flipped the predicted link scores. Results are rounded, the best results are shown in bold, second-best are underlined.

Data	DW	n2v	L$_1$	L$_2$	L$_{1+2}$	NERD	MS$_1$	MS
uni-email	0.914	†0.513	**0.964**	0.916	$\underline{0.940}$	0.736	0.692	0.870
polblogs	0.627	0.631	0.817	0.838	$\underline{0.853}$	0.724	0.851	**0.903**
new-zealand-collab^{W}	0.643	0.661	†0.768	**0.925**	$\underline{0.907}$	†0.606	0.855	0.865
interactome-stelzl	0.835	†0.513	**0.944**	0.773	$\underline{0.853}$	0.612	0.757	0.820
urban-streets-venice	**0.897**	0.870	0.828	0.634	0.711	0.597	†0.564	$\underline{0.890}$
interactome-figeys	0.533	†0.703	†**0.889**	0.653	$^{\dagger}\underline{0.865}$	0.730	0.730	0.819
us-air-traffic^{W}	0.616	0.552	0.685	**0.937**	$\underline{0.934}$	0.835	0.833	0.903
word-adjacency-japanese	†0.494	†0.570	†0.623	0.801	0.796	0.628	**0.855**	$\underline{0.831}$
urban-streets-ahmedabad	**0.953**	0.919	0.864	†0.577	0.685	†0.523	†0.658	$\underline{0.915}$
openflights^{W}	0.767	†0.621	0.934	$\underline{0.950}$	**0.960**	0.798	0.840	0.950
power	$\underline{0.936}$	0.897	0.874	0.800	0.828	0.620	0.566	**0.962**
facebook-organizations-L1	0.919	0.508	**0.977**	0.966	0.974	0.882	0.835	$\underline{0.976}$
reactome	0.908	0.580	**0.985**	0.944	0.961	0.890	0.786	$\underline{0.978}$
jdk	0.777	0.862	0.891	0.737	0.807	0.761	$\underline{0.973}$	**0.987**
advogato^{W}	0.769	0.505	0.868	$\underline{0.892}$	**0.905**	0.805	0.810	0.890
word-adjacency-spanish	†0.496	0.652	†0.754	**0.863**	0.848	0.732	$\underline{0.863}$	0.851
dblp-cite	0.834	†0.485	†0.551	0.742	0.646	**0.908**	0.828	$\underline{0.905}$
anybeat	0.672	0.523	0.748	$\underline{0.884}$	**0.894**	0.784	0.867	0.883
chicago-road	**0.998**	0.863	$\underline{0.986}$	0.735	0.874	†0.559	†0.579	0.909
foldoc^{W}	$\underline{0.946}$	0.575	**0.966**	0.848	0.914	0.629	0.658	0.888
physics-collab-arXiv^{W}	0.939	0.592	**0.983**	0.858	0.899	0.725	0.634	$\underline{0.964}$
google	0.859	0.775	0.903	0.878	$\underline{0.907}$	0.775	0.889	**0.976**
marvel-universe	0.864	†0.666	**0.914**	0.840	0.899	0.884	0.615	$\underline{0.910}$
internet-as	0.685	†0.742	0.659	0.930	$\underline{0.930}$	0.822	0.817	**0.932**
word-assoc^{W}	0.727	0.846	0.873	0.896	**0.922**	$\underline{0.902}$	0.848	0.862
cora	$\underline{0.938}$	0.815	**0.958**	0.834	0.880	0.847	0.826	0.926
arxiv-citation-HepTh	0.865	0.812	**0.966**	0.896	0.925	0.868	0.839	$\underline{0.952}$
digg-reply^{W}	†0.501	0.585	†0.604	0.772	0.761	**0.873**	$\underline{0.835}$	0.834
linux	0.734	0.663	0.701	0.733	0.754	0.835	$\underline{0.959}$	**0.965**
arxiv-citation-HepPh	0.952	0.890	**0.975**	0.881	0.923	0.870	0.813	$\underline{0.952}$
email-enron	0.816	†0.541	**0.988**	0.963	$\underline{0.974}$	0.873	0.860	0.949
inploid	0.667	0.736	0.532	$\underline{0.879}$	0.875	0.819	0.869	**0.891**
pgp-strong	0.879	0.568	**0.989**	0.927	0.946	0.848	0.804	$\underline{0.954}$
facebook-wall^{W}	0.865	0.744	**0.951**	0.865	$\underline{0.890}$	0.833	0.753	0.890
slashdot-threads^{W}	0.604	0.769	0.744	0.835	0.848	0.855	$\underline{0.883}$	**0.886**
python-dependency	0.790	0.763	†0.715	0.653	†0.632	0.889	**0.965**	$\underline{0.915}$
lkml-reply^{W}	0.494	0.612	0.719	**0.959**	$\underline{0.958}$	0.821	0.920	0.942
marker-cafe	0.539	0.853	0.832	$\underline{0.921}$	0.917	**0.949**	0.901	0.912
epinions-trust	0.611	0.679	0.875	$\underline{0.960}$	**0.964**	0.925	0.921	0.947
prosper	0.818	0.531	0.616	†0.650	†0.465	**0.956**	0.855	$\underline{0.927}$
livemocha	0.694	0.737	0.880	0.868	$\underline{0.884}$	**0.930**	0.854	0.881
foursquare-friendships-new	0.918	0.520	$\underline{0.976}$	0.948	0.961	0.858	0.792	**0.988**
google-plus	0.704	0.679	**0.960**	0.870	0.921	0.921	0.870	$\underline{0.960}$
twitter-higgs-retweet^{W}	0.630	$\underline{0.880}$	†0.800	†0.634	†0.707	0.874	**0.976**	0.822
amazon-copurchases-302	$\underline{0.966}$	0.850	**0.987**	0.931	0.957	0.589	0.656	0.946
notre-dame-web	0.967	0.938	**0.980**	0.946	0.971	0.930	0.891	$\underline{0.971}$
twitter-followers	0.549	$^{\dagger}\underline{0.987}$	†**0.989**	0.714	†0.977	0.955	0.839	0.887