PNAS
www.pnas.org

Supplementary Information for

Stacking Models for Nearly Optimal Link Prediction in Complex Networks

Amir Ghasemian, Homa Hosseinmardi, Aram Galstyan, Edoardo M. Airoldi and Aaron Clauset

To whom correspondence should be addressed.
E-mail: amir.ghasemianlangroodi@colorado.edu and aaron.clauset@colorado.edu

This PDF file includes:

Supporting Information Text

A. Methods for predicting missing links

Here, we describe in detail the three families of link predictors and their specific members used in the analysis, including the abbreviations used in the main text. In addition, we describe in more detail the setup of the supervised stacked generalization algorithm we use to combine individual predictors into a single algorithm.

Topological predictors. Topological predictors are simple functions of the observed network topology, e.g., counts of edges, measures of overlapping sets of neighbors, and measures derived from simple summarizations of the network's structure. We consider 42 topological predictors, which come in three types: global, pairwise, and node-based. Within these groups, the "pairwise" predictors include a number of topological features that are often used in the literature to directly predict missing links (1), e.g., the number of shared neighbors of i, j. A listing of all topological predictors is given in Table S1, along with corresponding literature references.

Global predictors. These predictors quantify various network-level statistics and are inherited by each pair of nodes i, j that is a candidate missing link. Their primary utility is to provide global context to other predictors under supervised learning. For example, a predictor that performs well on small networks, but poorly on larger networks, can be employed appropriately under a supervised model when the global measure of the network's size is available. Or, a large variance in the degree distribution would imply that a predictor based on degree product may be useful. Or, a large clustering coefficient would imply that an assortative community detection algorithm like modularity is likely to be useful. For this reason, global predictors are not expected by themselves to be accurate predictors of missing links (see Figs. S3 and S4 and Tables S6 and S7). These global predictors are generally useful in capturing missingness in unseen networks and not on the same network link prediction experiments. These features help to learn from existing configurations in training networks and generalize them to unseen networks in experiments such as transfer learning.

The 8 global predictors are the number of nodes (N), number of observed edges (OE), average degree (AD), variance of the degree distribution (VD), network diameter (ND), degree assortativity of graph (DA), network transitivity or clustering coefficient (NT), and average (local) clustering coefficient (ACC) (1–4).

Pairwise predictors. These predictors are functions of the joint topological properties of the pair of nodes i, j being considered.

The 14 pairwise predictors are the number of common neighbors of i, j (CN), shortest path between i, j (SP), Leicht-Holme-Newman index of neighbor sets of i, j (LHN) (Eq. 22 in Ref. (6)), personalized page rank (PPR), * preferential attachment or degree product of i, j (PA), Jaccard coefficient of the neighbor sets of i, j (JC), Adamic-Adar index of i, j (AA), resource allocation index of i, j (RA), the entry i, j in a low rank approximation (LRA) via a singular value decomposition (SVD) (LRA), the dot product of the i, j columns in the LRA via SVD (dLRA), the mean of entries i and j's neighbors in the LRA (mLRA), and simple approximations of the latter three predictors (LRA-approx, dLRA-approx, mLRA-approx) (1–4).

We omit from consideration several pairwise predictors found in the literature, e.g. edge betweenness centrality, due to their large computational complexity for an evaluation as large as ours.

Node-based predictors. These predictors are functions of the independent topological properties of the individual nodes i and j, and thus produce a pair of predictor values. Unlike many of the pairwise predictors, which can be used as standalone algorithms to predict missing links, these node-based predictors do not directly score the likelihood that i, j is a missing link. Instead, the particular function that converts the pair of node-based predictors into a score is learned within the supervised framework.

The 20 node-based predictors are two instance each of the local clustering coefficient (LCC), average neighbor degree (AND), shortest-path betweenness centrality (SPBC), closeness centrality (CC), degree centrality (DC), eigenvector centrality (EC), Katz centrality (KC), local number of triangles (LNT), Page rank (PR), and load centrality (LC) (1–4).

Model-based predictors. Model-based predictors are a broad class of prediction algorithms that rely on models of large-scale network structure to score pairs i, j that are more or less likely to be missing. To make link predictions, model-based algorithms employ one of two strategies: likelihood or optimization. In the first case, a method estimates a parametric probability $\Pr(i \to j \,|\, \theta)$ that a node pair should be connected, given a decomposition of a network into communities, as in the stochastic block model and its variants. In the second it predicts a link as missing if it would improve its measure of community structure, as in Infomap and modularity.

We consider 11 model-based predictors for missing links, which include many state-of-the-art in community detection algorithms (7), are sufficiently scalable to be applied in an evaluation as large as ours, and each of which has previously been used as a standalone link prediction algorithm. A listing of all model-based predictors is given in Table S2, along with the corresponding literature references.

For the model-based predictors that make predictions by likelihood, we follow Ref. (7) to employ a "model-specific" score function for each method. Under this approach, a particular method first decomposes the network into a set of communities using its corresponding parametric model, and then extracts from that same parametric model a score $\Pr(i \to j \,|\, \theta)$ for each candidate pair i, j. See Ref. (7) for additional details.

* By using a biased random walk we can find the personalized PageRank algorithm. This centrality could reflect the importance of nodes with respect to a biased set of specific nodes or a single specific node. In this paper using personalized page rank we consider j-th entry of the personalized page rank for node i as one of the edge-based features.

Embedding-based predictors. Embedding-based predictors are derived from graph embedding techniques, which attempt to automate the feature engineering phase of learning with graphs by projecting a network's nodes into a relatively low-dimensional latent space, with the goal of locally preserving the node neighborhoods. Embedding-based predictors are thus either node coordinates in such an embedding, or measure of distance between embedded pairs. We consider a total of 150 embedding-based predictors, all derived from 2 popular graph embedding algorithms, DeepWalk (emb-DW) (21)—a special case of node2vec (emb-node2vec) (22)—and the variational graph auto encoder (emb-vgae) (23).

Using emb-DW and emb-vgae, we embed each network into a 128-dimensional and 16-dimensional space, respectively. For each pair of nodes i, j, we then apply a Hadamard product function to the corresponding pair of coordinates to obtain 144 link predictors as features for supervised learning (24). To these, we add 6 more predictors by applying, for each of the 2 embedding methods, a different distance or similarity function to the corresponding pair of coordinate vectors: an inner product, an inner product with a sigmoid function, and Euclidean distance.

Stacked generalization and meta-learning for link prediction. Meta-learning or ensemble techniques are a powerful class of supervised machine learning algorithms that can learn from data how to combine individual predictors into a single, more accurate algorithm (25–28). By treating the output of individual prediction algorithms as features of the input instances themselves, a supervised meta-learning algorithm can construct a correlation function that relates which individual algorithm is most accurate on which subset of inputs. Of the several approaches to meta-learning, we focus on the approach of stacked generalization or model "stacking" (29), and we consider two boosting approaches (see below) as a robustness check. We leave further investigation of other meta-learning algorithms for future work.

Stacking aims to minimize the generalization error of a set of component learners. In the classic setting, the two training levels can be summarized as follows. Given a dataset $\mathcal{D} = \{(y_\ell, x_\ell), \ell \in \{1, ..., L\}\}$, where x_ℓ is the feature vector of the ℓ-th example and y_ℓ is its label, randomly split $\mathcal{D}$ into J "folds" appropriate for J-fold cross validation. Each fold j contributes once as a test set $\mathcal{D}^j$ and the rest contributes once as a training set $\mathcal{D}^{-j} = \mathcal{D} \setminus \mathcal{D}^j$. For each base classifier r, where $r \in \{1, ..., R\}$, called a level-0 generalizer, we fit it to the jth fold in the training set $\mathcal{D}^{-j}$ to build a model $\mathcal{M}_r^{-j}$, called a level-0 model. Now

Table S1. Abbreviations and descriptions of 42 topological predictors, across three types: *global* predictors (7), which are functions of the entire network and whose utility is in providing context to other predictors; *pairwise* predictors (15), which are functions of the joint topological properties of the pair i, j; and node-based predictors (20), which are functions of the independent topological properties of the nodes i and j, producing one value for each node in the pair i, j.

Abbreviation	Description	Global	Pairwise	Node-based	Ref.
N	number of nodes	•			(2)
OE	number of observed edges	•			(2)
AD	average degree	•			(2)
VD	variance of degree distribution	•			(2)
ND	network diameter	•			(2)
DA	degree assortativity of graph	•			(5)
NT	network transitivity (clustering coefficient)	•			(2)
ACC	average (local) clustering coefficient	•			(2)
CN	common neighbors of i, j		•		(1)
SP	shortest path between i, j		•		(1)
LHN	Leicht-Holme-Newman index of neighbor sets of i, j		•		(6)
PPR	j-th entry of the personalized page rank of node i		•		(5)
PA	preferential attachment (degree product) of i, j		•		(1)
JC	Jaccard's coefficient of neighbor sets of i, j		•		(1)
AA	Adamic/Adar index of i, j		•		(1)
RA	resource allocation index of i, j		•		(5)
LRA	entry i, j in low rank approximation (LRA) via singular value decomposition (SVD)		•		(4)
dLRA	dot product of columns i and j in LRA via SVD for each pair of nodes i, j		•		(4)
mLRA	average of entries i and j's neighbors in low rank approximation		•		(4)
LRA-approx	an approximation of LRA		•		(4)
dLRA-approx	an approximation of dLRA		•		(4)
mLRA-approx	an approximation of mLRA		•		(4)
LCC_i, LCC_j	local clustering coefficients for i and j			•	(5)
AND_i, AND_j	average neighbor degrees for i and j			•	(5)
$SPBC_i$, $SPBC_j$	shortest-path betweenness centralities for i and j			•	(5)
CC_i, CC_j	closeness centralities for i and j			•	(5)
DC_i, DC_j	degree centralities for i and j			•	(5)
EC_i, EC_j	eigenvector centralities for i and j			•	(5)
KC_i, KC_j	Katz centralities for i and j			•	(5)
LNT_i, LNT_j	local number of triangles for i and j			•	(5)
PR_i, PR_j	Page rank values for i and j			•	(5)
LC_i, LC_j	load centralities for i and j			•	(5)

Table S2. Abbreviations and descriptions of 11 model-based predictors, across two types: *likelihood* predictors (7), which score each pair i, j according to a parametric model $\Pr(i \to j \,|\, \theta)$ learned by decomposing the network under a probabilistic generative model of network structure such as the stochastic block model or its variants; and, *optimization* predictors (4), which score each pair i, j according to whether adding them would increase a corresponding (non-probabilistic) community structure objective function, as in the Map Equation or the modularity function. The "Code Ref." column provides the references used for implementation of these predictors.

Abbreviation	Description	Likelihood	Optimization	Ref.	Code Ref.
Q	modularity, Newman-Girvan		•	(8)	(9)
Q-MR	modularity, Newman's multiresolution		•	(10)	ourselves
Q-MP	modularity, message passing		•	(11)	from authors
B-NR (SBM)	Bayesian stochastic block model, Newman and Reinert	•		(12)	(13)
B-NR (DC-SBM)	Bayesian degree-corrected stochastic block model, Newman and Reinert	•		(12)	(13)
B-HKK (SBM)	Bayesian stochastic block model, Hayashi, Konishi and Kawamoto	•		(14)	(15)
cICL-HKK (SBM)	Corrected integrated classification likelihood, stochastic block model	•		(14)	(15)
Infomap	Map equation		•	(16)	(17)
MDL (SBM)	Minimum description length, stochastic block model	•		(18)	(19)
MDL (DC-SBM)	Minimum description length, degree-corrected stochastic block model	•		(18)	(19)
S-NB	Spectral with non-backtracking matrix	•		(20)	ourselves

for each data point x_ℓ in the jth test set, we employ these level-0 models $\mathcal{M}_r^{-j}$ to predict the output $z_{r\ell}$. The new data set $\mathcal{D}_{CV} = \{(y_\ell, z_{1\ell}, ..., z_{R\ell}), \ell \in \{1, ..., L\}\}$, is now prepared for the next training level, called a level-1 generalizer. In the second training phase, an algorithm learns a new model from this data, denoted as $\tilde{\mathcal{M}}$. Now, we again train the base classifiers using the whole data $\mathcal{D}$, noted as $\mathcal{M}_r$, we complete the training phase and the models are ready to classify a new data point x. The new data point will first be fed into the trained base classifiers $\mathcal{M}_r$ and then the output of these level-0 models will construct the input for the next level model $\tilde{\mathcal{M}}$.

In the network setting of link prediction, the classifiers (predictors) in the first level are all unsupervised, and therefore, we alter the stacked generalization algorithm as follows to account for this difference and to adapt it to a network setting. In this paper, we assume a missingness function f that samples edges uniformly at random from E so that each edge $(i, j) \in E$ is observed with probability α. Then for a given network $G = (V, E)$, the uniformly observed edges E' construct the observed network $G' = (V, E')$, where $|E'| = \alpha|E|$ ($\alpha = 0.8$ in our experiments). Here, we use only the uniform edge-removal model and leave the analysis of any non-uniform edge removal model for future work. The removed edges $E \setminus E'$ are considered as held-out data in the link prediction task. Then, in order to train a model, we remove $1 - \alpha'$ ($\alpha' = 0.8$ in our experiments) of the edges as our positive examples ($E'' \subset E'$) according to the same missingness model f, and take all non-edges $V \times V - E'$ in the observed network G' as negative examples. Although this procedure makes the negative samples noisy, since the networks are sparse, it introduces a negligible error in the learned model, and should not significantly effect the model's performance. In our setting, the unsupervised classifiers in the first level are our level-0 predictors, and we use the scores coming from these link prediction techniques as our meta features. The second training phase is conducted through supervised learning with 5-fold cross validation on the training set. We use a standard supervised random forest algorithm for the meta-learning step, and assess the learning process on 3 within-family models (topol. only, model-based only, and embed. only) and on 4 across-family models (all families, and each of topol. & model, topol. & embed, and model & embed.), for a total of 7 stacked models.

Model selection. In order to choose the best parameters of the model using 5-fold cross validation, we can choose the parameters of the model through optimizing the AUC performance or the F-measure. In the main text all figures and tables show results for a standard random forest with the parameters chosen through a cross validation to maximize F-measure on training set and the results are reported on holdout test set; the threshold for reported precision and recall is chosen to maximize F-measure on holdout test set. Results for, instead, optimizing using the AUC are given in Table S19 which can be compared with Table 1 in the main text.

Alternative meta-learning algorithms. In addition to a standard random forest, we also evaluate two methods of boosting, XGBoost (30) and AdaBoost (31), for learning a single algorithm over the individual predictors. The results from these meta-learning algorithms are provided in Tables S20-S23 for different choices of model selection through AUC or F-measure.

B. Tests on synthetic data

We evaluate individual predictors and their stacked generalization on a set of synthetic networks with known structure that varies along three dimensions: (i) the degree distribution's variability, being low (Poisson), medium (Weibull), or high (power law); (ii) the number of "communities" or modules $k \in \{1, 2, 4, 16, 32\}$; and (iii) the fuzziness of the corresponding community boundaries $\epsilon = m_{\text{out}}/m_{\text{in}}$, the ratio of edges between the clusters to edges inside the clusters, being low, medium, or high. These synthetic networks thus range from homogeneous to heterogeneous random graphs (degree distribution), from no modules to many modules (k), and from weakly to strongly modular structure (ϵ).

We generate these networks using the degree-corrected stochastic block model (DC-SBM) (32), which allows us to systematically control each of these parameters to generate a synthetic network. Moreover, because both the data generating process and the missing function f (here, uniform at random) are known, we may exactly calculate the theoretical upper limit that any link prediction algorithm could achieve for a given parameterization of the generative process. This upper bound provides an unambiguous reference point for how optimal any particular link prediction algorithm is, and the three structural dimensions of the synthetic networks allow us to extract some general insights as to what properties increase or decrease the predictability of missing links.

In this section, we first describe the generative processes, and then detail the calculations for optimal predictions. For completeness, we first specify the mathematical forms of the Weibull and power-law degree distributions used in some settings. The Poisson distribution is fully specified by the choice of the mean degree parameter c.

The Weibull distribution can be written as

$$f(r) = cr^{\beta-1}e^{-\lambda r^{\beta}} \quad , \qquad [1]$$

where the constant c is the corresponding normalization constant when r is the degree of a node, and the parameters λ, β specify the shape of the distribution. When $\beta < 1$, this distribution decays more slowly than simple exponential, meaning it exhibits greater variance, but not as much variance as can a power-law distribution. See Table S3 for the particular values used in our synthetic data.

The power-law distribution can be written as

$$f(r) = cr^{-\gamma} \quad , \qquad [2]$$

where, again, c is the corresponding normalization constant when r is the degree of a node, and γ is the "scaling" exponent that governs the shape of the distribution. When $\gamma \in (2, 3)$, the mean is finite but the variance is infinite. See Table S3 for the particular values used in our synthetic data.

Generating synthetic networks. Although each type of synthetic network can be generated under the DC-SBM model, for some choices of the number of communities k and degree distribution, the generative process simplifies greatly. Below, we describe the generation procedures for the synthetic networks according to the simplest generative model available for a given choice of parameters, which is noted in the subsection heading.

Generating ER networks ($k = 1$, Poisson)

- choose number of nodes n, and average degree c, or the interaction probability $p = c/(n-1)$,
- connect each pair of nodes independently with probability p.

Generating DC-ER networks ($k = 1$, Weibull or power law)

- choose number of nodes n, and average degree c,
- compute the parameters of degree distribution for the average degree c,
- generate a degree sequence with length n with the computed parameters in the previous step,
- compute the number of edges for the network as $m = \frac{1}{2}\sum_i d_i$,
- make a multi-edge between each pair of nodes i, j independently with the Poisson probability with rate $\lambda = \frac{d_i}{d_{g_i}}\frac{d_j}{d_{g_j}}\omega$, where $\omega = 2m$.
  We then convert this multigraph into a simple (unweighted) network by collapsing multi-edges. Because these networks are parameterized to be sparse, this operation does not substantially alter the network's structure, as only a small fraction of all edges are multi-edges.

Generating SBM networks ($k > 1$, Poisson)

- choose number of nodes n, number of clusters k, average degree c, and $\tilde{\epsilon}$, the ratio of number of edges connected to a node outside and inside its cluster, i.e., $\tilde{\epsilon} = \frac{p_{\text{out}}(n/k)}{p_{\text{in}}(n/k)} = \frac{p_{\text{out}}}{p_{\text{in}}}$; by choosing c and $\tilde{\epsilon}$, the mixing probabilities can then be computed as $p_{\text{in}} = \frac{c}{(n/k)(1+\tilde{\epsilon}(k-1))}$ and $p_{\text{out}} = \tilde{\epsilon}\, p_{\text{in}}$,

Table S3. Parameters used to generate the synthetic networks, via the DC-SBM structured random graph model, used to evaluate the link prediction methods studied here. Redundant information (derivable from the other parameters) is listed parenthetically, for convenience. See Section B.

Region	Model	Number of modules k	Parameters
low ϵ	Poisson	1	$n = 505, p = 0.008$
low ϵ	Poisson	2	$n = 512, p_{\text{in}} = 0.03, p_{\text{out}} = 0.0003 \ \ (\epsilon = 0.009)$
low ϵ	Poisson	4	$n = 512, p_{\text{in}} = 0.06, p_{\text{out}} = 0.0003 \ \ (\epsilon = 0.015)$
low ϵ	Poisson	16	$n = 512, p_{\text{in}} = 0.25, p_{\text{out}} = 0.0003 \ \ (\epsilon = 0.015)$
low ϵ	Poisson	32	$n = 512, p_{\text{in}} = 0.49, p_{\text{out}} = 0.0003 \ \ (\epsilon = 0.019)$
low ϵ	Weibull	1	$n = 497, \lambda = 1, \beta = 0.5, \omega = 2350$
low ϵ	Weibull	2	$n = 520, \lambda = 1, \beta = 0.4, \epsilon = 0.002$
low ϵ	Weibull	4	$n = 604, \lambda = 1, \beta = 0.4, \epsilon = 0.002$
low ϵ	Weibull	16	$n = 773, \lambda = 1, \beta = 0.4, \epsilon = 0.04$
low ϵ	Weibull	32	$n = 939, \lambda = 1, \beta = 0.15, \epsilon = 0.0005$
low ϵ	power law	1	$n = 507, \beta = 1.6, \omega = 5436$
low ϵ	power law	2	$n = 511, \beta = 1.7, \epsilon = 0.0003$
low ϵ	power law	4	$n = 511, \beta = 1.8, \epsilon = 0.002$
low ϵ	power law	16	$n = 983, \beta = 1.6, \epsilon = 0.0015$
low ϵ	power law	32	$n = 1029, \beta = 1.41, \epsilon = 0.0015$
moderate ϵ	Poisson	1	$n = 511, p = 0.016$
moderate ϵ	Poisson	2	$n = 512, p_{\text{in}} = 0.03, p_{\text{out}} = 0.005 \ \ (\epsilon = 0.20)$
moderate ϵ	Poisson	4	$n = 512, p_{\text{in}} = 0.04, p_{\text{out}} = 0.006 \ \ (\epsilon = 0.39)$
moderate ϵ	Poisson	16	$n = 512, p_{\text{in}} = 0.16, p_{\text{out}} = 0.006 \ \ (\epsilon = 0.6)$
moderate ϵ	Poisson	32	$n = 511, p_{\text{in}} = 0.31, p_{\text{out}} = 0.006 \ \ (\epsilon = 0.62)$
moderate ϵ	Weibull	1	$n = 510, \lambda = 1, \beta = 0.7, \omega = 1424$
moderate ϵ	Weibull	2	$n = 501, \lambda = 1, \beta = 0.4, \epsilon = 0.06$
moderate ϵ	Weibull	4	$n = 593, \lambda = 1, \beta = 0.4, \epsilon = 0.08$
moderate ϵ	Weibull	16	$n = 589, \lambda = 1, \beta = 0.4, \epsilon = 0.2$
moderate ϵ	Weibull	32	$n = 640, \lambda = 1, \beta = 0.22, \epsilon = 0.05$
moderate ϵ	power law	1	$n = 545, \beta = 1.9, \omega = 1428$
moderate ϵ	power law	2	$n = 506, \beta = 1.7, \epsilon = 0.05$
moderate ϵ	power law	4	$n = 540, \beta = 1.8, \epsilon = 0.05$
moderate ϵ	power law	16	$n = 655, \beta = 1.7, \epsilon = 0.01$
moderate ϵ	power law	32	$n = 702, \beta = 1.41, \epsilon = 0.01$
high ϵ	Poisson	1	$n = 512, p = 0.03$
high ϵ	Poisson	2	$n = 512, p_{\text{in}} = 0.025, p_{\text{out}} = 0.006 \ \ (\epsilon = 0.25)$
high ϵ	Poisson	4	$n = 512, p_{\text{in}} = 0.04, p_{\text{out}} = 0.007 \ \ (\epsilon = 0.48)$
high ϵ	Poisson	16	$n = 512, p_{\text{in}} = 0.14, p_{\text{out}} = 0.007 \ \ (\epsilon = 0.75)$
high ϵ	Poisson	32	$n = 512, p_{\text{in}} = 0.27, p_{\text{out}} = 0.007 \ \ (\epsilon = 0.93)$
high ϵ	Weibull	1	$n = 489, \lambda = 1, \beta = 0.9, \omega = 1216$
high ϵ	Weibull	2	$n = 506, \lambda = 1, \beta = 0.4, \epsilon = 0.2$
high ϵ	Weibull	4	$n = 590, \lambda = 1, \beta = 0.4, \epsilon = 0.32$
high ϵ	Weibull	16	$n = 600, \lambda = 1, \beta = 0.4, \epsilon = 0.5$
high ϵ	Weibull	32	$n = 631, \lambda = 1, \beta = 0.22, \epsilon = 0.13$
high ϵ	power law	1	$n = 514, \beta = 2.2, \omega = 1722$
high ϵ	power law	2	$n = 536, \beta = 1.7, \epsilon = 0.08$
high ϵ	power law	4	$n = 526, \beta = 1.8, \epsilon = 0.14$
high ϵ	power law	16	$n = 626, \beta = 1.7, \epsilon = 0.1$
high ϵ	power law	32	$n = 673, \beta = 1.5, \epsilon = 0.05$

- generate the type of the nodes independently with prior probabilities q_r for $r = \{1, ..., k\}$,
- connect each pair of nodes i, j independently with probability $p_{g_i g_j}$, where

$$p_{g_i, g_j} = \begin{cases} p_{\text{in}} & \text{if } g_i = g_j \\ p_{\text{out}} & \text{if } g_i \neq g_j \end{cases} .$$

Generating DC-SBM networks ($k > 1$, Weibull or power law)

- choose number of nodes n, average degree c, and ϵ, the ratio of number of edges between the clusters to edges inside the clusters, i.e., $\epsilon = m_{\text{out}}/m_{\text{in}}$, where m_{in} is the number of edges inside the clusters, and m_{out} is the number of edges between the clusters,
- generate the type of nodes independently with prior probabilities q_r for $r = \{1, ..., k\}$,

- compute the parameters of degree distribution for average degree c,
- generate a degree sequence with length n with the computed parameters in previous step, and compute the aggregate degrees for each cluster noted as $d_r = \sum_{i:g_i=r} d_i$,
- compute the total number of edges for the network as $m = \frac{1}{2}\sum_i d_i$,
- using ϵ, compute the number of edges inside and outside the clusters, denoted as m_{in}, and m_{out}, as $m_{\text{in}} = m/(1+\epsilon)$ and $m_{\text{out}} = \epsilon\, m_{\text{in}}$,
- because we do not assume heterogeneity for the size and volume of clusters in the generating process (node types are randomized uniformly and edges are created uniformly inside and between clusters), then we may approximate the number of edges inside each cluster r as $m_{\text{in}}^{(r)} = m_{\text{in}}/k$, the number of edges between cluster r and any other cluster as $m_{\text{out}}^{(r)} = \dfrac{m_{\text{out}}}{k/2}$, and the number of edges between each pair of clusters r and s as $m_{\text{out}}^{(rs)} = m_{\text{out}} \Big/ \binom{k}{2}$,
- make a multi-edge between each pair of nodes i, j with types r, s, independently with the Poisson probability with rate $\lambda_{r,s}(d_i, d_j) = \dfrac{d_i}{d_r}\dfrac{d_j}{d_s}\omega_{r,s}$, where

$$\omega_{r,s} = \begin{cases} 2m_{\text{in}}^{(r)} & \text{if } r = s \\ m_{\text{out}}^{(rs)} & \text{if } r \neq s \end{cases} .$$

  We then convert this multigraph into a simple (unweighted) network by collapsing multi-edges. Because these networks are parameterized to be sparse, this operation does not substantially alter the network's structure, as only a small fraction of all edges are multi-edges.

It is worthwhile to mention that ϵ in DC-SBM is related to $\tilde{\epsilon}$ in SBM as $\epsilon = m_{\text{out}}/m_{\text{in}} = (k-1)p_{\text{out}}/p_{\text{in}} = (k-1)\tilde{\epsilon}$. Therefore, for the results, we used $\epsilon = m_{\text{out}}/m_{\text{in}}$ for both SBM and DC-SBM.

Optimal link prediction accuracy on a synthetic network. To calculate an upper bound on link prediction accuracy that any algorithm could achieve in one of our synthetic networks, we exploit the mathematical equivalence of the Area Under the ROC Curve (AUC) and the binary classification probability that a prediction algorithm $\mathcal{A}$ assigns a higher score to a missing link (true positive) than to a non-edge (true negative):

$$\text{AUC} = \Pr(\text{tes} > \text{tnes}) \ , \qquad [3]$$

where tes and tnes denote the scores assigned to a missing edge (te; true positive) and to a non-edge (tne; true negative). To derive the optimal AUC for any possible link prediction algorithm, it suffices to calculate this probability under a given parametric generative model $\mathcal{M}(\theta)$ and missingness function f.

Assumptions and definitions. In the calculations that follow, we treat separately the three generative process subcases of the DC-SBM described above, and we define $n = |V|$, $m = |E|$. If two edges are assigned the same score by the generative model, we assume that such ties are broken uniformly at random.

For these calculations, we also assume that algorithm $\mathcal{A}$ has access to the planted partition assignment $\mathcal{P}$ of the k clusters used to generate the edges. In practice, this assumption implies that our upper bound may be unachievable in cases where the detectability of $\mathcal{P}$ is either computational hard or information-theoretically impossible (see Ref. (33)), e.g., when community boundaries are fuzzy (high ϵ).

Given this partition, we define n_i, m_i, and $\tilde{m}_i$ to be the number of nodes, number of edges, and number of non-edges, respectively, within community i. And, we define m_{ij} and $\tilde{m}_{ij}$ to be the number of edges and non-edges, respectively, that span communities i and j.

Finally, when we estimate Eq. (3) via Monte Carlo sampling, we select 100,000 uniformly random te (true positive) and tne (true negative) pairs.

Optimal AUC for ER. The AUC for an Erdős-Rényi random graph is

$$\text{AUC} = \Pr(\text{tes} > \text{tnes}) = 1/2 \ . \qquad [4]$$

In words: because the generative model assigns the same score $p = c/(n-1)$ to every edge and every non-edge, and because ties are broken at random, the maximum AUC can be no better than chance.

Optimal AUC for DC-ER. As in the ER case, this random graph has $k = 1$ communities, but unlike the ER case, the degree distribution here is heterogeneous (Weibull or power law). We calculate the maximum AUC for any algorithm $\mathcal{A}$ on this synthetic network via Eq. (5) (below), which we estimate numerically via Monte Carlo sampling on Eq. (3).

$$
\begin{aligned}
\mathrm{AUC} &= \Pr(\mathrm{tes} > \mathrm{tnes}) \\
&= \sum_{u_1.v_1,u_2,v_2} p(u_1v_1 > u_2v_2, d_{i_1} = u_1, d_{j_1} = v_1, d_{i_2} = u_2, d_{j_2} = v_2 \mid (i_1,j_1) \in E, (i_2,j_2) \notin E) \\
&= \sum_{u_1,v_1,u_2,v_2} \mathbb{1}(u_1v_1 > u_2v_2) p(d_{i_1} = u_1, d_{j_1} = v_1, d_{i_2} = u_2, d_{j_2} = v_2 \mid (i_1,j_1) \in E, (i_2,j_2) \notin E) \\
&\overset{\text{(Bayes Thm.)}}{=} \sum_{u_1,v_1,u_2,v_2} \mathbb{1}(u_1v_1 > u_2v_2) \frac{p((i_1,j_1) \in E, (i_2,j_2) \notin E \mid d_{i_1} = u_1, d_{j_1} = v_1, d_{i_2} = u_2, d_{j_2} = v_2)}{p((i_1,j_1) \in E, (i_2,j_2) \notin E)} \\
&\quad \times p(d_{i_1} = u_1) p(d_{j_1} = v_1) p(d_{i_2} = u_2) p(d_{j_2} = v_2) \ . \qquad [5]
\end{aligned}
$$

Optimal AUC for SBM. In the general case ($k > 1$) of the stochastic block model (SBM), the te and tne probabilities under the generative model depend on the mixing matrix of edge densities between and within communities. When these densities are set such that the planted partition $\mathcal{P}$ is easily recoverable by a community detection algorithm (a range of parameters called the "deep detectable regime" (DDR) (33), where $\epsilon \to 0$), Eq. (3) can be rewritten as Eq. (6):

$$
\begin{aligned}
\mathrm{AUC} &= \Pr(\mathrm{tes} > \mathrm{tnes}) \\
&= \Pr(\mathrm{tes} > \mathrm{tnes} | \text{both inside}) \Pr(\text{both inside}) \times \text{number of possibilities} \\
&\quad + \Pr(\mathrm{tes} > \mathrm{tnes} | \text{both outside}) \Pr(\text{both outside}) \times \text{number of possibilities} \\
&\quad + \Pr(\mathrm{tes} > \mathrm{tnes} | \text{te inside, tne outside}) \Pr(\text{te inside, tne outside}) \times \text{number of possibilities} \\
&\quad + \Pr(\mathrm{tes} > \mathrm{tnes} | \text{te outside, tne inside}) \Pr(\text{te outside, tne inside}) \times \text{number of possibilities} \ . \qquad [6]
\end{aligned}
$$

The four terms of Eq. (6) can then be computed as follows, where α is the sampling rate of observed edges:

- First term:

$$
\begin{aligned}
&\Pr(\mathrm{tes} > \mathrm{tnes} \mid \text{both inside}) \Pr(\text{both inside}) \\
&= \frac{m_i\alpha}{\sum_i m_i\alpha + \sum_{i\neq j} m_{ij}\alpha} \times \frac{\tilde{m}_i}{\sum_i \tilde{m}_i + \sum_{i\neq j} \tilde{m}_{ij}} \\
&= \frac{\binom{n_i}{2} p_{\mathrm{in}}}{\sum_i \binom{n_i}{2} p_{\mathrm{in}} + \sum_{i\neq j} n_i n_j p_{\mathrm{out}}} \times \frac{\binom{n_i}{2}(1-p_{\mathrm{in}})}{\binom{n_i}{2}(1-p_{\mathrm{in}}) + n_i n_j (1-p_{\mathrm{out}})} \\
&= \frac{1}{2}\left(\frac{p_{\mathrm{in}}}{k^3(p_{\mathrm{in}} + (k-1)p_{\mathrm{out}})}\right) = \frac{c_{\mathrm{in}}}{2k^4 c} \ , ^{\dagger} \qquad [7]
\end{aligned}
$$

  Finally, because the number of possibilities is k^2, the first term simplifies as $\frac{1}{2}\left(\frac{p_{\mathrm{in}}}{k^3(p_{\mathrm{in}}+(k-1)p_{\mathrm{out}})}\right) \times k^2 \approx 1/2k$.

- Second term:

$$
\begin{aligned}
&\Pr(\mathrm{tes} > \mathrm{tnes} \mid \text{both outside}) \Pr(\text{both outside}) \\
&= \frac{m_{ij}\alpha}{\sum_i m_i\alpha + \sum_{i\neq j} m_{ij}\alpha} \times \frac{\tilde{m}_{ij}}{\sum_i \tilde{m}_i + \sum_{i\neq j} \tilde{m}_{ij}} \\
&= \frac{n_i n_j p_{\mathrm{out}}}{\sum_i \binom{n_i}{2} p_{\mathrm{in}} + \sum_{i\neq j} n_i n_j p_{\mathrm{out}}} \times \frac{n_i n_j (1-p_{\mathrm{out}})}{\binom{n_i}{2}(1-p_{\mathrm{in}}) + n_i n_j (1-p_{\mathrm{out}})} \\
&= 2\left(\frac{p_{\mathrm{out}}}{k^3(p_{\mathrm{in}} + (k-1)p_{\mathrm{out}})}\right) = \frac{2c_{\mathrm{out}}}{k^4 c}. \qquad [8]
\end{aligned}
$$

  Finally, because the number of possibilities is $\binom{k}{2}^2 \approx \frac{k^4}{4}$ the second term simplifies as $\frac{2p_{\mathrm{out}}}{k^3(pin+(k-1)p_{\mathrm{out}})} \times \frac{k^4}{4} = \frac{kp_{\mathrm{out}}}{(p_{\mathrm{in}}+(k-1)p_{\mathrm{out}})} \approx 0$.

[†] Note that in the last line we assume equally-sized homogeneous clusters i.e. $n_i = n/k$.

- Third term:

$$\begin{aligned}
&\Pr(\text{tes} > \text{tnes} \,|\, \text{tes inside, tnes outside}) \Pr(\text{tes inside, tnes outside}) \\
&= \frac{m_i\alpha}{\sum_i m_i\alpha + \sum_{i\neq j} m_{ij}\alpha} \times \frac{\tilde{m}_{ij}}{\sum_i \tilde{m}_i + \sum_{i\neq j} \tilde{m}_{ij}} \\
&= \frac{\binom{n_i}{2}p_{\text{in}}}{\sum_i \binom{n_i}{2}p_{\text{in}} + \sum_{i\neq j} n_i n_j p_{\text{out}}} \times \frac{n_i n_j (1-p_{\text{out}})}{\binom{n_i}{2}(1-p_{\text{in}}) + n_i n_j (1-p_{\text{out}})} \\
&= 2\left(\frac{p_{\text{in}}}{k^3(p_{\text{in}} + (k-1)p_{\text{out}})}\right) = \frac{2c_{\text{in}}}{k^4 c}.
\end{aligned} \tag{9}$$

  Finally, because the number of possibilities is $k\binom{k}{2} = k^2(k-1)/2$ the third term simplifies as $\frac{2p_{\text{in}}}{k^3(p_{\text{in}}+(k-1)p_{\text{out}})} \times \frac{k^2(k-1)}{2} = (k-1)/k$.

- Last term:

$$\Pr(\text{tes} > \text{tnes} \,|\, \text{tes outside, tnes inside}) \Pr(\text{tes outside, tnes inside}) = 0 \tag{10}$$

  Finally, when the SBM parameters are such that the model is in the deep detectable regime (DDR), the fourth term is zero, because the assigned scores to the outer edges are smaller than the assigned scores to inner edges, under the assumption that the algorithm $\mathcal{A}$ can recover the planted partition (which occurs with probability 1 in DDR).

Given the above simplifications, we arrive at the final expression to compute the optimal AUC for the SBM in the DDR:

$$\begin{aligned}
\text{AUC} &= \Pr(\text{tes} > \text{tnes}) \\
&= \frac{1}{2k} + \frac{k-1}{k} \\
&= \frac{2k-1}{2k}.
\end{aligned} \tag{11}$$

For example, the upper bounds on link predictability under this model for $k = \{2, 4, 8, 16, 32\}$ are AUC $= \{0.75, 0.875, 0.94, 0.97, 0.98\}$, respectively. Because these values are computed in the deep detectable regime, they are accurate only when ϵ is low (sharp community boundaries, or $\mathcal{P}$ is known or recoverable).

For any value of ϵ, we may numerically calculate the upper bound on AUC using Monte Carlo sampling via the Eq. (3), applied to the generated networks. The corresponding values represent conservative upper bounds on the maximum AUC because under Monte Carlo because we assume that $\mathcal{P}$ is known.

In practice, a community detection algorithm would need to infer that from the observed data, and this event is not guaranteed when ϵ is higher (33), due to a phase transition in the detectability (recoverability) of the planted partition structure that maximizes the predictability of missing links. We suggest that the gap observed in Fig. 2 between this conservative upper bound and accuracy of the best stacked models in the high-ϵ settings can be attributed to this difference. That is, the stacked models are closer to the true upper bound than our calculations suggest.

Optimal AUC for DC-SBM. In the general case ($k > 1$) of the degree-corrected SBM, the te and tne probabilities under the generative model depend on the specified degree distribution (Weibull or power law) and the mixing matrix of edge densities between and within communities.

In this setting, Eq. (3) can be rewritten as Eq. (6) when $\epsilon \to 0$; however to compute each term we must also condition on the degrees of the nodes. Following the same logic as in the SBM analysis, we compute each term separately as follows.

- First term:

$$\begin{aligned}
&\Pr(\text{tes} > \text{tnes} \,|\, \text{both inside}) \\
&= \sum_{u_1,v_1,u_2,v_2} \mathbb{1}(u_1v_1 > u_2v_2) \frac{p((i_1,j_1)\in E, (i_2,j_2)\notin E \mid d_{i_{1,2}} = u_{1,2}, d_{j_{1,2}} = v_{1,2})}{p((i_1,j_1)\in E, (i_2,j_2)\notin E)} \\
&\quad \times p(d_{i_1} = u_1)p(d_{j_1} = v_1)p(d_{i_2} = u_2)p(d_{j_2} = v_2) \ ,
\end{aligned} \tag{12}$$

  where $d_{i_{1,2}} = u_{1,2}$ means $d_{i_1} = u_1$ and $d_{i_2} = u_2$.

- Second term:

$$\begin{aligned}
&\Pr(\text{tes} > \text{tnes} \,|\, \text{both outside}) \\
&= \sum_{u_1,v_1,u_2,v_2} \mathbb{1}(u_1v_1 > u_2v_2) \frac{p((i_1,j_1)\in E, (i_2,j_2)\notin E \mid d_{i_{1,2}} = u_{1,2}, d_{j_{1,2}} = v_{1,2})}{p((i_1,j_1)\in E, (i_2,j_2)\notin E)} \\
&\quad \times p(d_{i_1} = u_1)p(d_{j_1} = v_1)p(d_{i_2} = u_2)p(d_{j_2} = v_2) \ .
\end{aligned} \tag{13}$$

- Third term:

$$\begin{aligned}&\Pr(\text{tes} > \text{tnes} \,|\, \text{te inside}, \text{tne outside})\\ &= \sum_{u_1,v_1,u_2,v_2} \mathbb{1}(u_1 v_1 m_{rr} > u_2 v_2 m_{rs}) \frac{p((i_1,j_1) \in E, (i_2,j_2) \notin E \mid d_{i_{1,2}} = u_{1,2}, d_{j_{1,2}} = v_{1,2})}{p((i_1,j_1) \in E, (i_2,j_2) \notin E)}\\ &\quad \times p(d_{i_1} = u_1) p(d_{j_1} = v_1) p(d_{i_2} = u_2) p(d_{j_2} = v_2) \ . \end{aligned} \tag{14}$$

- Fourth term:

$$\begin{aligned}&\Pr(\text{tes} > \text{tnes} \,|\, \text{te outside}, \text{tne inside})\\ &= \sum_{u_1,v_1,u_2,v_2} \mathbb{1}(u_1 v_1 m_{rs} > u_2 v_2 m_{rr}) \frac{p((i_1,j_1) \in E, (i_2,j_2) \notin E \mid d_{i_{1,2}} = u_{1,2}, d_{j_{1,2}} = v_{1,2})}{p((i_1,j_1) \in E, (i_2,j_2) \notin E)}\\ &\quad \times p(d_{i_1} = u_1) p(d_{j_1} = v_1) p(d_{i_2} = u_2) p(d_{j_2} = v_2) \ . \end{aligned} \tag{15}$$

We compute these terms numerically using Monte Carlo samples of the generated networks to calculate Eq. (3).

C. Empirical corpus for link prediction evaluations

To evaluate and compare the different link prediction algorithms in a practical setting, we have provided 550 networks that is a slightly expanded version of the "CommunityFitNet corpus" (7), a novel data set drawn from the Index of Complex Networks (ICON) (34). This corpus spans a variety of network sizes and structures, with 23% social, 32% biological, 23% economic, 12% technological, 3% information, and 7% transportation graphs (Fig. S1). For replication, reuse, and extension of our work, we make publicly available the entire network corpus [‡].

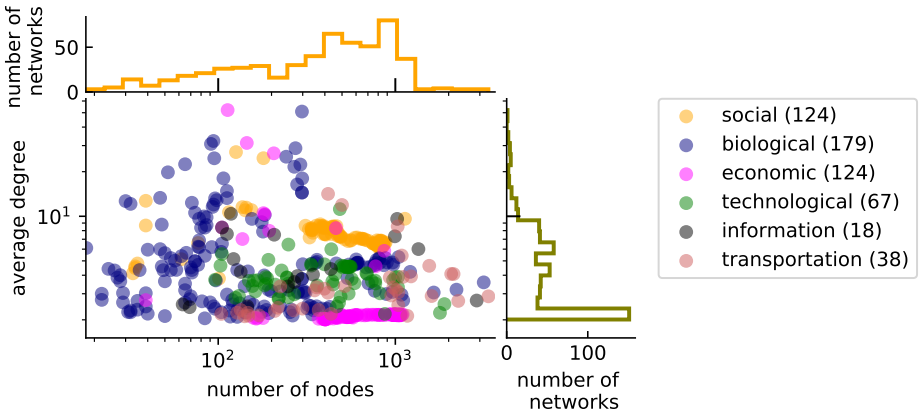


Fig. S1. Average degree versus number of nodes for our corpus, a slightly expanded version of the CommunityFitNet corpus (7), consisting of 550 real-world networks drawn from the Index of Complex Networks (ICON) (34), including social, biological, economic, technological, information, and transportation graphs.

[‡] Available at https://github.com/Aghasemian/OptimalLinkPrediction

D. Evaluation of the link prediction algorithms

In practical settings, the true missingness function f may not be known, and f is likely to vary with the scientific domain, the manner in which the network data is collected, and the scientific question of interest. Here, we do not consider all possible functions f, and instead analyze an f that samples edges uniformly at random from E so that each edge $(i, j) \in E$ is observed with probability α.[§] This choice presents a hard test for link prediction algorithms, as f is independent of both observed edges and metadata. Other models of f, e.g., in which missingness correlates with edge or node characteristics, may better capture particular scientific settings and are left for future application-specific work. Our results thus provide a general, application-agnostic assessment of link predictability and method performance.

Most of the predictors we consider are predictive only under a supervised learning approach, in which we learn a model of how these node-pair features correlate with edge missingness. This supervised approach to link prediction poses one specific technical challenge. Under supervised learning, we train a method using 5-fold cross validation by choosing as positive examples a subset of edges $E'' \subset E'$ according to the same (uniformly random) missingness model f. But applying this missingness function can only create positive training examples (missing links), while supervised learning also needs negative examples (non-links). Other approaches to supervised link prediction have made specific assumptions to mitigate this issue. For example, in a temporal network, an algorithm can be trained using the links and non-links observed during an earlier training time frame (35, 36), or if some missing links are known *a priori*, they may be used as training examples (37). However, such approaches require information, e.g., the evolution of a network over time, that are not commonly available, and thus they do not generalize well to the broad evaluation setting of this study. Here, we use a different, more general approach to evaluate and compare supervised link prediction methods on a large set of static networks.

Specifically, we exploit two features of our empirical networks (Fig. S1) to construct reasonably reliable training sets. First, all observed non-edges $V \times V - E'$ in observed graph $G' = (V, E')$ are taken as negative examples (non-links). If G is a snapshot of an evolving network, then links that form in the future of G will form between pairs that are not currently connected. Therefore, the non-links of G can reasonably be considered as negative examples up to the time of observation. Second, most real-world networks, for which link prediction is relevant, are sparse. In this case, considering the non-links as negative examples includes only a small number of negative examples in the training set, which are in fact positive examples in the test set. Although these mislabeled edges are not true negative examples, their sparsity in the training set (because the size of the non-links set is $O(n^2)$ compared to the $O(n)$ size of the missing links set, this approach can only induce a $O(1/n)$ bias in the learning) is likely compensated for by the improved generalizability of taking a supervised learning approach compared to an unsupervised approach.

[§] Unless otherwise specified, results reflect a choice of $\alpha = 0.8$, i.e., 20% of edges are unobserved (holdout set); other values produce qualitatively similar results.

E. Diversity in prediction error

A Lorenz curve is a standard method to visualize the skewness of predictor importances on individual networks. Fig. S2 shows the set of 550 curves for the learned importances for each of the networks in our empirical corpus, along with the average curve across the ensemble (red solid line). This ensemble exhibits a mean Gini coefficient of 0.83 ± 0.095 (mean±S.D.), and illustrates that the importances tend to be highly skewed, such that a relatively small subset of predictors account for the majority of prediction accuracy.

The entropy of a distribution is a standard summary statistic of such variation, and provides a compact measure for comparing different distributions. Given a discrete random variable X drawn from a probability distribution p, the entropy is defined as $H(X) = E_{p(X)}[-\log(p(X))]$, and can be interpreted as the amount of uncertainty in X, the average number of bits we need to store X, or the minimum number of binary questions on average to guess a draw from X (38). The maximum entropy of discrete random variable occurs for a uniform distribution, and is simply $\log_2(L)$, where L is the number of possible outcomes for X.

To compute the feature importance entropies for each domain in each family (Table S4), we first compute the feature importances for each domain. To this end, we first choose all the networks in a domain j, as either social, biological, economic, technological, information, or transportation networks) and a set of predictors ℓ, as either (i) all 203 predictors, (ii) the 42 topological predictors, (iii) the 11 model-based predictors, or (iv) the 150 embedding predictors. We then learn the feature importances of this set for each network at each domain via supervised learning, as described above.[¶] And finally, we aggregate the Gini importances of the networks at that domain. We use a weighted rank aggregation algorithm called Cross-Entropy Monte Carlo (39). This algorithm uses an iterative procedure to search for the rank aggregation that is as close as possible to the rank importances of all given networks at each domain. We denote the aggregated feature importances of all predictors in a family ℓ for networks in domain j by a vector $X_j^{(\ell)}$. The "probability" associated with the i-th predictor in family ℓ and for networks in domain j is then computed as $p_{ij}^{(\ell)} = X_{ij}^{(\ell)} \Big/ \sum_i X_{ij}^{(\ell)}$. For each setting, the entropy of the corresponding distribution is reported in Table S4.

Comparing the entropy of the learned importances with the simple upper-limit entropy given by a uniform distribution illustrates the diversity of learned importances among the predictors. To provide a more intuitive sense of how skewed the distribution is, we compare the empirical entropy value with that of a simple piece-wise artificial distribution. Specifically, we consider a distribution in which at least 90% of the density is allocated uniformly across the best x% of the predictors, with the remaining density allocated uniformly across the rest. We then choose the x that minimizes the difference between this model entropy and the empirical entropy.

All predictors. Applied to the importances of all predictors, only 14% of predictors account for 90% of the importance in social networks. Other domains require far more predictors, e.g., 34% for biological and 40% for technological networks. Notably, the top x% in each family of predictors are different across domains. The values in Table S4 show that most of the variation in importances can be explained by at most 82 of 203 total predictors for transportation networks, 21 out of 42 topol. predictors for biological, and information networks, 9 of 11 model-based predictors for biological, technological, and information networks, and 91 of 150 embed. predictors for economic networks. Also we see that across these predictor sets most of the uncertainty can be explained by at least 29 out of 203, 10 out of 42, 8 out of 11, and 30 out of 150 predictors for social networks, which illustrates the simplicity of link prediction in social networks.

Feature-wise entropy. We also compute the feature-wise entropy for each family ℓ, which captures the distribution of aggregated predictor importances, summing across domains. We denote the predictor importance of all predictors in a family ℓ by a vector X^{ℓ}. The importance "probability" in the i-th entry of this vector for family ℓ can be computed as $p_i^{(\ell)} = \sum_j X_{ij}^{(\ell)} \Big/ \sum_{ij} X_{ij}^{(\ell)}$, which quantifies the proportion of total importance of predictor i in all domains versus the total importance of all predictors. For each family, the entropy of the corresponding probability distribution is reported in Table S4. And, as before, comparing the entropy of the learned importances with the simple upper-limit entropy given by a uniform distribution illustrates that regardless of the domain, the importances are spread widely across predictors.

Family-wise entropy. Finally, we compute the family-wise entropies for each domain j, under an alternative formulation. Denoting the importance of predictor i in domain j as X_{ij}, and the set of all predictors in family ℓ as $\mathcal{P}^{\ell}$, we compute the importance "probability" of the predictor i in domain j as $p_{ij} = \sum_{i \in \mathcal{P}^{\ell}} X_{ij} \Big/ \sum_i X_{ij}$. Then, the family-wise entropy can be defined using this distribution (see Table S5). As before, comparing the entropy of each domain j with the simple upper-limit entropy given by a uniform distribution illustrates the variance of predictor importances among different families. Moreover, these entropies also illustrate that the variation of importances in social networks is smaller (the most important predictors are in topological and embedding families [see Fig. 2 in the main text]), compared to that of non-social networks.

Taking the learned importances for all predictors, Fig. S3 plots the distributions of the importance-ranks (how often predictor i was the jth most important predictor) for all 203 predictors, applied to all 550 networks. This visualization reveals that among the most important predictors (high importance-rank across networks) are those in the model-based family, along with a subset of topological predictors, and the six notions of distance or similarity for embedding-based predictors. The

[¶] Unless otherwise noted, the reported results are based on training a random forest. We also used AdaBoost and XGBoost and similar results have been observed (see Tables S20-S23).

Table S4. Predictor importance entropy for each domain in each family. For each family, the "entropy" column measures the uncertainty in predictor importance of each domain. Also we consider an artificial distribution on predictors that explain (uniformly) the 90% of the probability by the best x% of the predictors, and (uniformly) 10% of the probability by the rest. We choose x such that the artificial entropy be as close as possible to the empirical entropy. The column "top x%", shows the percentage of best predictors with 90% probability. The (n) value shows the corresponding number for the top x%. The "uniform" column reports the entropy if predictor importance were uniform. The "feature-wise" row within each family reports the entropy held by each predictor, summing across domains. Entropies are reported in units of bits.

Family	Domain	Entropy	Top x% (n)	Uniform
all topol., model, and embed. predictors (203)	social	5.59	14.29 (29)	7.66
	biological	6.67	33.99 (69)	
	economic	6.06	20.69 (42)	
	technological	6.85	39.41 (80)	
	information	6.33	26.11 (53)	
	transportation	6.88	40.40 (82)	
	feature-wise	6.52	34.50 (68)	
Family	Domain	Entropy	Top x% (n)	Uniform
all topol. predictors (42)	social	3.98	23.81 (10)	5.39
	biological	4.84	50 (21)	
	economic	4.15	28.57 (12)	
	technological	4.7	42.86 (18)	
	information	4.88	50 (21)	
	transportation	4.69	42.86 (18)	
	feature-wise	4.64	40.48 (17)	
Family	Domain	Entropy	Top x% (n)	Uniform
all model-based predictors (11)	social	3.26	72.73 (8)	3.46
	biological	3.4	81.82 (9)	
	economic	3.16	63.64 (7)	
	technological	3.38	81.82 (9)	
	information	3.35	81.82 (9)	
	transportation	3.19	63.64 (7)	
	feature-wise	3.31	72.73 (8)	
Family	Domain	Entropy	Top x% (n)	Uniform
all embed. predictors (150)	social	5.58	20 (30)	7.23
	biological	6.77	53.33 (80)	
	economic	6.91	60.67 (91)	
	technological	6.81	55.33 (83)	
	information	6.47	41.34 (62)	
	transportation	6.73	51.33 (77)	
	feature-wise	6.64	47.33 (71)	

Table S5. Family wise entropy. Importance entropy of all features in a family for each domain. The "uniform" column reports the entropy if predictor importance were uniform. Entropies are reported in units of bits.

	Domain	Entropy	Uniform
family wise	social	0.67	1.58
	biological	1.10	
	economic	0.93	
	technological	1.15	
	information	1.03	
	transportation	1.2	

least important predictors (low importance-rank across networks) fall primarily in the topological family. None of embedding predictors ranked among the least important, and instead nearly all of them rank in the broad middle of overall importance. Most embedding-based predictors do rank highly for a few individual networks, but it is a different subset of embedding predictors for each network. Thus, across networks, embedding predictors are uniformly middling in their importance, and none are dominant in a network domain.

Categorizing predictors by their importance-rank distributions. To analyze and identify the most important predictors in comparison with the least important predictors on average, we extract a hierarchical clustering of the rank similarities of Fig. S3, which is shown in Fig. S4. The first group (purple cluster) corresponds to the predictors that are nearly always the least important across networks, such as VD, OE, DA, and ACC (Fig. S4, inset panel 1). The large group of predictors (green cluster) on the middle of the hierarchy correspond to the embedding-based predictors, whose distribution of importances is concentrated in the middle range (Fig. S4, inset panel 2). And a third group (red cluster) includes to predictors that receive high importance across nearly all 550 networks (Fig. S4, inset panels 5–7), as well as some with more bimodal importance (Fig. S4, inset panels 3–4).

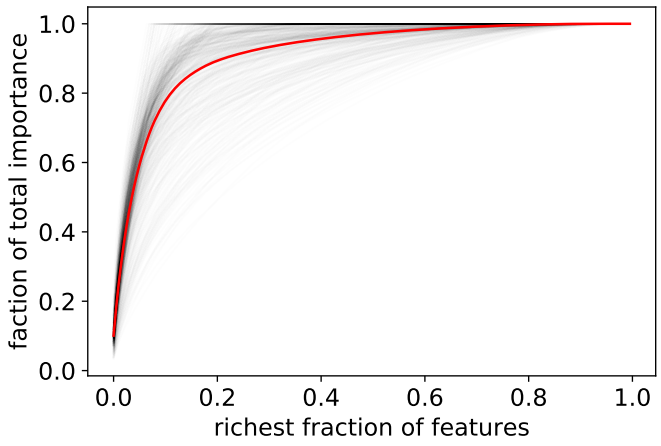


Fig. S2. Lorenz curves of the importance of the features. These curves illustrate that in a large portion of empirical networks, a very large fraction of learned "importance" belongs to a small fraction of predictors. The red solid line shows the average Lorenz curve over the 550 networks.

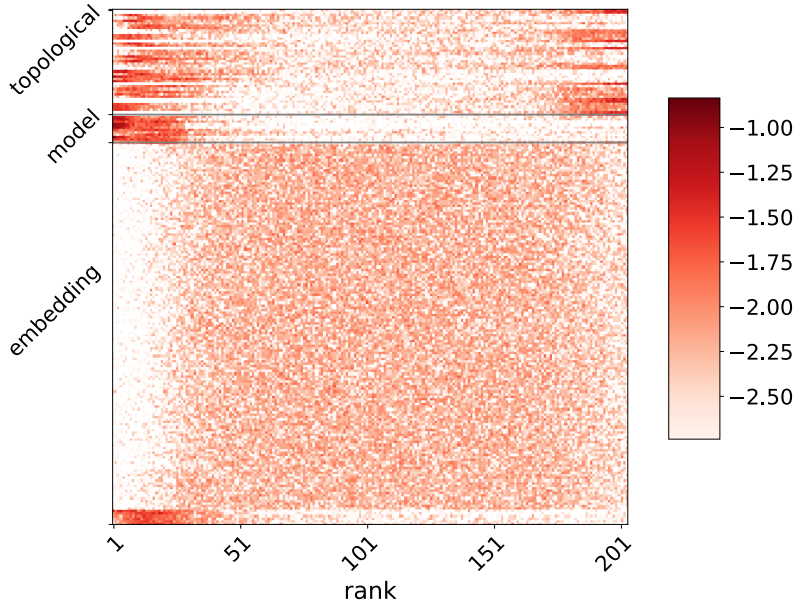


Fig. S3. Distribution of the importance ranks of each predictor across 550 networks. The most important features typically belong to model-based and topological predictor families. Among embedding predictor, the most important correspond to the distance measures among the embedded vectors. Almost all vector embedding predictors have middling levels of importance, although they are rarely the worst predictors. The distribution of the ranks is logarithmic (base 10).

Minimal number of features for stacking. Fig. S5 shows the distribution of the minimum number of predictors k^*, that is needed to achieve at least 95% of final AUC for each family of stacking methods. These curves highlight that in a large portion of networks, we can achieve high predictability using roughly 10 predictors.

Performance as weak learners. Considering each predictor as a "weak learner" from the perspective of the Adaboost theorem, Figs. S6 and S7 show the histogram of AUC performances of all model-based and topological individual predictors across the 550 networks in our empirical corpus. The large majority of these predictors have AUC larger than 0.5, while a modest portion of individual topological predictors fall below this threshold, meaning that they are not useful in link prediction for a given network. These topological predictors are of the "global" type (see SI Appendix, section A and Table S1) and are not expected to be individually predictive. We note, however, that these predictors are likely to be useful in any transfer learning setting, in which we train on a subset of networks and apply the model to unseen networks. Transfer learning for link prediction is out of scope of the present work and we leave it for future study.

AUC, precision, and recall across tests. Tables S6 and S7 present the link prediction performance measured by AUC, precision, and recall, for all individual topological predictors applied to the 550 real-world networks in our empirical corpus and the 45 generated synthetic data.

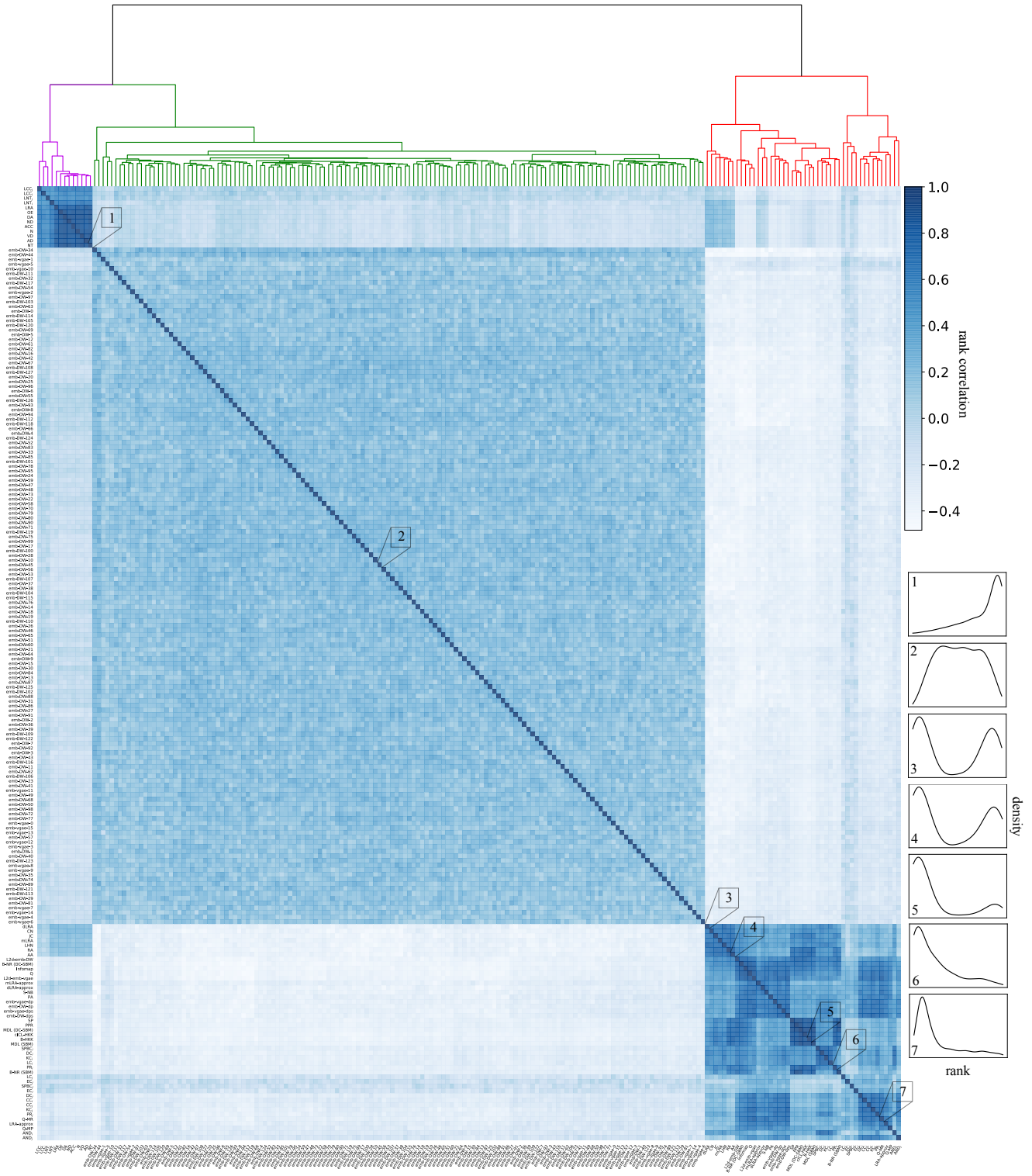


Fig. S4. A clustering of features based on the similarities of their rank in Gini importance. Clusters show similar importance distribution among 550 networks. Embedding predictors (green cluster) appear in middle ranks uniformly for different networks (inset 2). The purple cluster shows the worst importance among different networks (inset 1). The red cluster shows better importance and the most important features are located to the right of this group (insets 3–7).

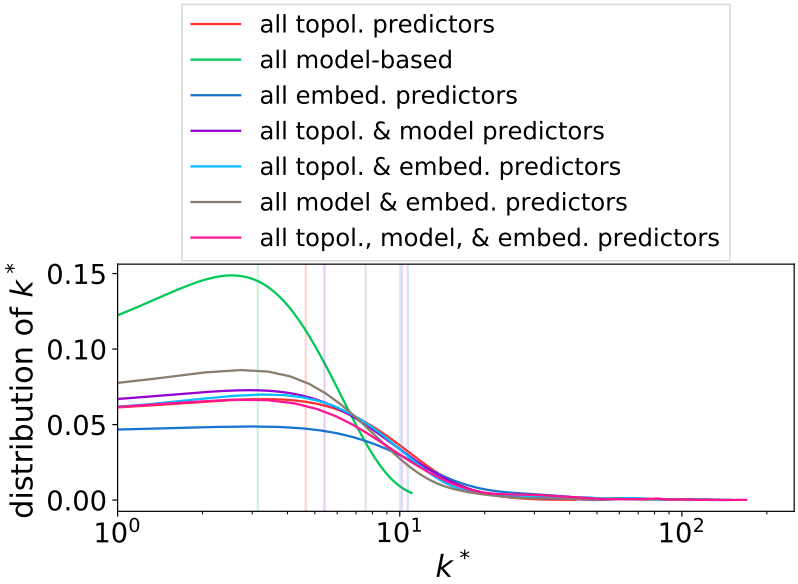


Fig. S5. Distribution of the minimum number of features k^* that is needed to achieve at least 95% of final AUC for each family of stacking methods. Each vertical line indicates the corresponding mean of each distribution.

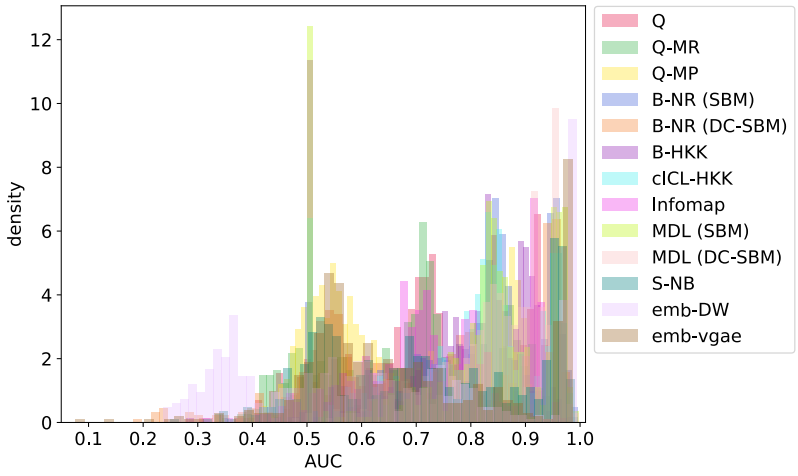


Fig. S6. Histogram of AUC performances on 550 empirical networks for all 11 model-based "weak learners" and two embedding link predictors used in model stacking.

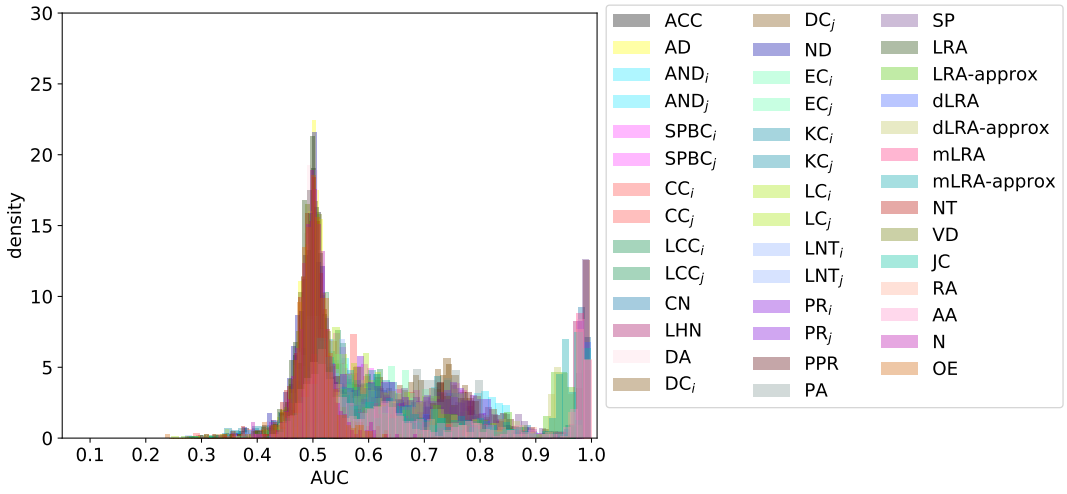


Fig. S7. Histogram of AUC performances on 550 empirical networks for all 42 individual topological "weak learners" used in model stacking.

Table S6. Link prediction performance (mean±S.D.) on holdout test set, measured by AUC, precision, and recall, for individual topological predictors applied to the 550 structurally diverse networks in our corpus.

algorithm	AUC	precision	recall
ACC	0.5 ± 0.04	0.02 ± 0.07	0.21 ± 0.29
AD	0.5 ± 0.03	0.03 ± 0.13	0.2 ± 0.28
AND_i	0.62 ± 0.13	0.05 ± 0.14	0.18 ± 0.23
AND_j	0.61 ± 0.11	0.04 ± 0.1	0.18 ± 0.23
SPBC_i	0.64 ± 0.1	0.06 ± 0.13	0.18 ± 0.19
SPBC_j	0.57 ± 0.08	0.04 ± 0.11	0.17 ± 0.21
CC_i	0.58 ± 0.08	0.06 ± 0.17	0.21 ± 0.25
CC_j	0.6 ± 0.11	0.04 ± 0.13	0.24 ± 0.25
LCC_i	0.56 ± 0.08	0.04 ± 0.11	0.18 ± 0.23
LCC_j	0.54 ± 0.07	0.03 ± 0.1	0.17 ± 0.24
CN	0.68 ± 0.19	0.23 ± 0.32	0.36 ± 0.35
LHN	0.67 ± 0.19	0.17 ± 0.24	0.31 ± 0.29
DA	0.5 ± 0.03	0.03 ± 0.12	0.2 ± 0.29
DC_i	0.68 ± 0.1	0.05 ± 0.11	0.17 ± 0.19
DC_j	0.66 ± 0.09	0.05 ± 0.1	0.17 ± 0.19
ND	0.5 ± 0.03	0.02 ± 0.09	0.22 ± 0.32
EC_i	0.58 ± 0.09	0.05 ± 0.12	0.18 ± 0.22
EC_j	0.61 ± 0.09	0.08 ± 0.19	0.19 ± 0.23
KC_i	0.58 ± 0.1	0.05 ± 0.12	0.21 ± 0.25
KC_j	0.61 ± 0.11	0.06 ± 0.16	0.18 ± 0.23
LC_i	0.64 ± 0.1	0.06 ± 0.13	0.16 ± 0.18
LC_j	0.57 ± 0.08	0.05 ± 0.12	0.17 ± 0.22
LNT_i	0.55 ± 0.08	0.03 ± 0.1	0.2 ± 0.25
LNT_j	0.55 ± 0.08	0.03 ± 0.08	0.18 ± 0.23
PR_i	0.66 ± 0.11	0.06 ± 0.12	0.17 ± 0.2
PR_j	0.63 ± 0.1	0.06 ± 0.15	0.18 ± 0.2
PPR	0.74 ± 0.17	0.17 ± 0.23	0.28 ± 0.25
PA	0.68 ± 0.1	0.07 ± 0.15	0.19 ± 0.2
SP	0.75 ± 0.15	0.06 ± 0.11	0.37 ± 0.35
LRA	0.5 ± 0.04	0.02 ± 0.09	0.21 ± 0.3
LRA-approx	0.68 ± 0.18	0.23 ± 0.3	0.23 ± 0.22
dLRA	0.68 ± 0.19	0.22 ± 0.31	0.34 ± 0.33
dLRA-approx	0.71 ± 0.16	0.19 ± 0.26	0.26 ± 0.25
mLRA	0.67 ± 0.19	0.22 ± 0.32	0.32 ± 0.33
mLRA-approx	0.7 ± 0.16	0.15 ± 0.25	0.26 ± 0.25
NT	0.5 ± 0.04	0.02 ± 0.09	0.22 ± 0.3
VD	0.5 ± 0.03	0.03 ± 0.12	0.23 ± 0.31
JC	0.67 ± 0.19	0.23 ± 0.34	0.32 ± 0.32
RA	0.67 ± 0.19	0.26 ± 0.37	0.36 ± 0.36
AA	0.68 ± 0.19	0.25 ± 0.35	0.35 ± 0.35
N	0.5 ± 0.03	0.02 ± 0.04	0.2 ± 0.29
OE	0.5 ± 0.04	0.03 ± 0.1	0.2 ± 0.29

Table S7. Link prediction performance (mean±S.D.) on holdout test set, measured by AUC, precision, and recall, for individual topological predictors applied to the 45 synthetic networks.

algorithm	AUC	precision	recall
ACC	0.5 ± 0.02	0.01 ± 0.01	0.12 ± 0.2
AD	0.5 ± 0.02	0.03 ± 0.15	0.12 ± 0.2
AND_i	0.61 ± 0.1	0.03 ± 0.1	0.15 ± 0.15
AND_j	0.6 ± 0.1	0.03 ± 0.11	0.16 ± 0.22
$SPBC_i$	0.64 ± 0.11	0.07 ± 0.18	0.15 ± 0.16
$SPBC_j$	0.63 ± 0.1	0.05 ± 0.13	0.1 ± 0.08
CC_i	0.6 ± 0.11	0.04 ± 0.15	0.22 ± 0.23
CC_j	0.59 ± 0.11	0.04 ± 0.11	0.17 ± 0.16
LCC_i	0.63 ± 0.11	0.06 ± 0.16	0.19 ± 0.2
LCC_j	0.63 ± 0.1	0.06 ± 0.15	0.16 ± 0.14
CN	0.71 ± 0.14	0.16 ± 0.18	0.18 ± 0.13
LHN	0.71 ± 0.14	0.05 ± 0.05	0.22 ± 0.17
DA	0.5 ± 0.01	0.0 ± 0.0	0.21 ± 0.31
DC_i	0.67 ± 0.13	0.04 ± 0.11	0.16 ± 0.17
DC_j	0.67 ± 0.12	0.06 ± 0.14	0.11 ± 0.1
ND	0.5 ± 0.02	0.01 ± 0.01	0.17 ± 0.27
EC_i	0.62 ± 0.11	0.04 ± 0.11	0.18 ± 0.21
EC_j	0.63 ± 0.12	0.05 ± 0.11	0.12 ± 0.13
KC_i	0.58 ± 0.09	0.02 ± 0.04	0.14 ± 0.18
KC_j	0.58 ± 0.08	0.03 ± 0.06	0.14 ± 0.13
LC_i	0.63 ± 0.11	0.05 ± 0.11	0.16 ± 0.2
LC_j	0.63 ± 0.11	0.06 ± 0.15	0.15 ± 0.12
LNT_i	0.64 ± 0.12	0.04 ± 0.12	0.24 ± 0.27
LNT_j	0.63 ± 0.11	0.04 ± 0.15	0.21 ± 0.18
PR_i	0.65 ± 0.12	0.05 ± 0.11	0.18 ± 0.2
PR_j	0.65 ± 0.12	0.06 ± 0.12	0.13 ± 0.14
PPR	0.76 ± 0.15	0.12 ± 0.19	0.19 ± 0.2
PA	0.72 ± 0.17	0.09 ± 0.13	0.17 ± 0.19
SP	0.74 ± 0.13	0.02 ± 0.02	0.24 ± 0.22
LRA	0.5 ± 0.02	0.03 ± 0.15	0.18 ± 0.27
LRA-approx	0.71 ± 0.14	0.1 ± 0.14	0.18 ± 0.17
dLRA	0.71 ± 0.14	0.14 ± 0.14	0.17 ± 0.13
dLRA-approx	0.75 ± 0.15	0.14 ± 0.16	0.23 ± 0.21
mLRA	0.69 ± 0.13	0.06 ± 0.06	0.15 ± 0.17
mLRA-approx	0.71 ± 0.13	0.03 ± 0.03	0.21 ± 0.2
NT	0.5 ± 0.02	0.01 ± 0.01	0.12 ± 0.21
VD	0.5 ± 0.01	0.03 ± 0.15	0.21 ± 0.28
JC	0.7 ± 0.14	0.05 ± 0.06	0.19 ± 0.17
RA	0.71 ± 0.14	0.2 ± 0.22	0.16 ± 0.15
AA	0.71 ± 0.14	0.19 ± 0.2	0.18 ± 0.13
N	0.5 ± 0.02	0.03 ± 0.15	0.23 ± 0.32
OE	0.5 ± 0.02	0.03 ± 0.15	0.19 ± 0.27

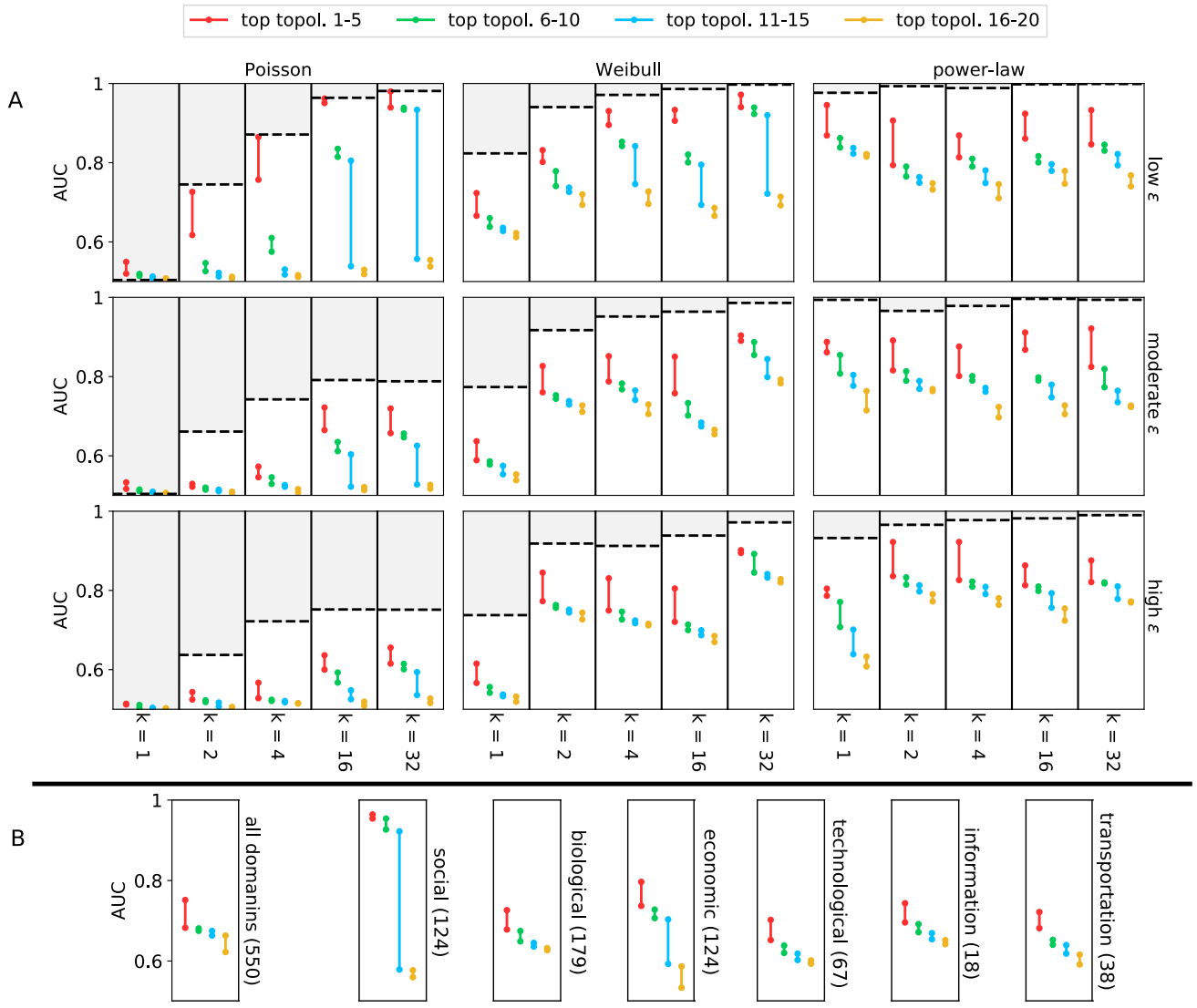


Fig. S8. (A) On synthetic networks, the mean link prediction performance (AUC) of the top 20 topological individual predictors across three forms of structural variability: (left to right, by subpanel) degree distribution variability, from low (Poisson) to high (power law); (top to bottom, by subpanel) fuzziness of community boundaries, ranging from low to high ($\epsilon = m_{\rm out}/m_{\rm in}$, the ratio of edges between the clusters to edges inside the clusters); and (left to right, within subpanel) the number of communities k. Across settings, the dashed line represents the theoretical maximum performance achievable by any link prediction algorithm (SI Appendix, section B). In each instance, the top topological features perform nearly optimal, and generally perform better when networks exhibit heavier-tailed degree distributions and more communities with distinct boundaries. For fuzzier communities the predictability deteriorates with increasing number of communities. (B) On real-world networks, the mean link prediction performance for the same predictors across all domains, and by individual domain. Top topological features exhibit superior performance, and they achieve nearly perfect accuracy on social networks. The performance, however, varies considerably across domains, with biological, technological, transportation, and information networks exhibiting the lowest link predictability. The order of the lines within a subpanel follows the order in the legend. The first top 5 topological features are listed in Tables S11– S12.

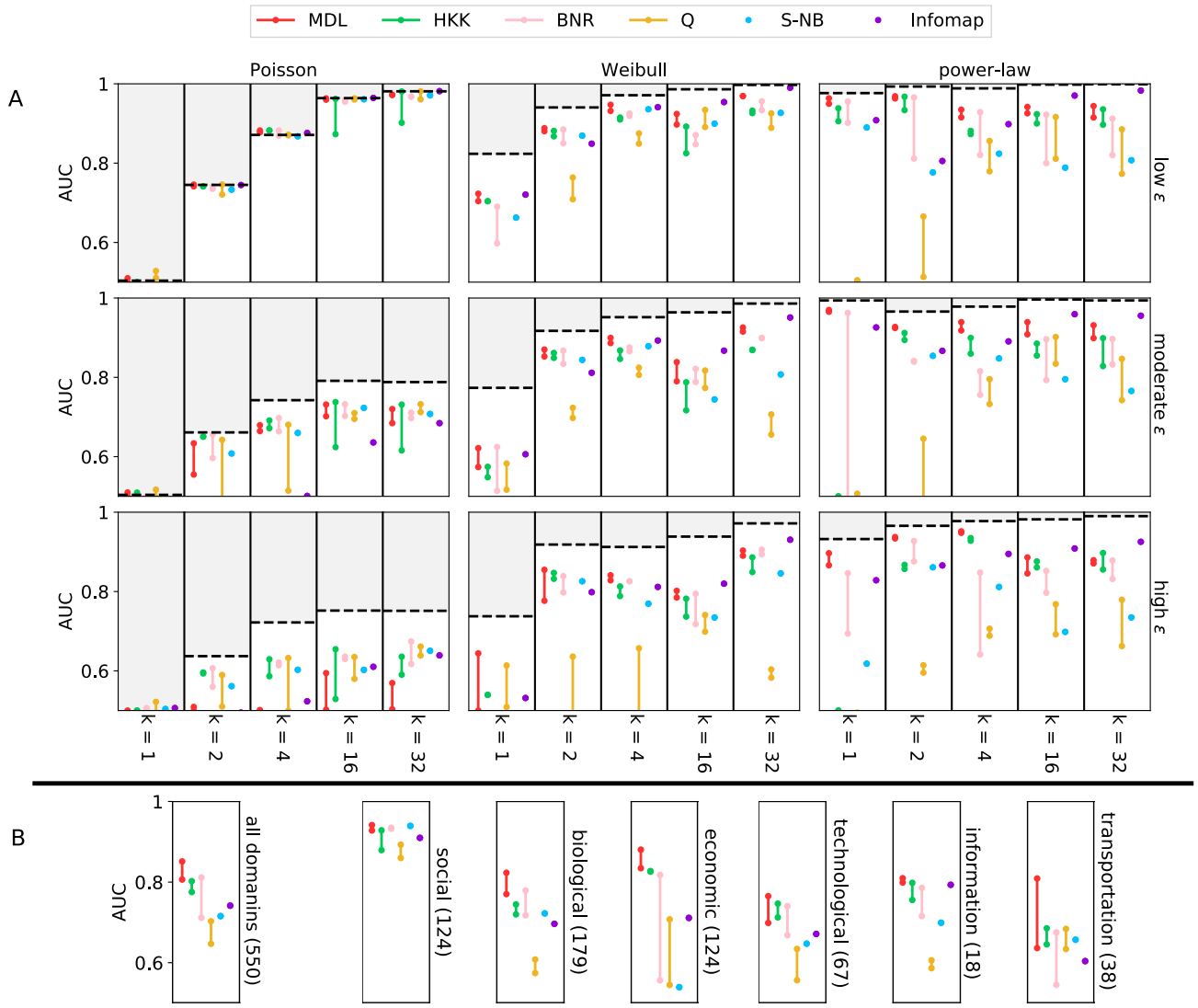


Fig. S9. (A) On synthetic networks, the mean link prediction performance (AUC) of model-based individual predictors across three forms of structural variability: (left to right, by subpanel) degree distribution variability, from low (Poisson) to high (power law); (top to bottom, by subpanel) fuzziness of community boundaries, ranging from low to high ($\epsilon = m_{\text{out}}/m_{\text{in}}$, the ratio of edges between the clusters to edges inside the clusters); and (left to right, within subpanel) the number of communities k. Across settings, the dashed line represents the theoretical maximum performance achievable by any link prediction algorithm (SI Appendix, section B). In each instance, the performances of model-based individual predictors vary considerably. For Poisson degree distribution almost all predictors perform equally well. For heavier-tailed degree distributions and more communities with distinct boundaries models based on SBM as expected and surprisingly Infomap perform very well. More interestingly Infomap performs slightly better than SBM when we have more fuzzier communities. (B) On real-world networks, the mean link prediction performance for the same predictors across all domains, and by individual domain. Among model-based predictors, on average, MDL family, S-NB, BNR family, and HKK family perform equally well on social networks, however, MDL family, exhibit superior performance on other domains. The performance, however, varies considerably across domains, with technological networks exhibiting the lowest link predictability. The order of the lines within a subpanel follows the order in the legend.

Table S8. Mean performance gap on holdout test set for each method in synthetic data.

Algorithm	Average gap ($\langle \Delta \text{AUC} \rangle$)
Q	0.187
Q-MR	0.198
Q-MP	0.191
B-NR (SBM)	0.085
B-NR (DC-SBM)	0.123
cICL-HKK	0.09
B-HKK	0.12
Infomap	0.083
MDL (SBM)	0.075
MDL (DC-SBM)	0.071
S-NB	0.138
mean indiv. model.	0.124
mean indiv. topol.	0.251
mean indiv. topol. & model	0.224
emb-DW	0.2
emb-vgae	0.172
all topol.	0.052
all model-based	0.034
all embed.	0.086
all topol. & model	0.038
all topol. & embed.	0.054
all model & embed.	0.039
all topol., model & embed.	0.037

Table S9. The AUC gap of the best 10 predictors on holdout test set with the upper-bound AUC for synthetic data.

Rank	Algorithm	Average gap ($\langle \Delta \text{AUC} \rangle$)
1	MDL (DC-SBM)	0.071
2	MDL (SBM)	0.075
3	Infomap	0.083
4	B-NR (SBM)	0.085
5	cICL-HKK	0.09
6	PPR	0.114
7	B-HKK	0.12
8	B-NR (DC-SBM)	0.123
9	dLRA-approx.	0.13
10	S-NB	0.138

Table S10. Link prediction performance (mean$\pm$S.D.) on holdout test set, measured by AUC, precision, and recall, for link prediction algorithms applied to the 45 synthetic networks.

Algorithm	AUC	Precision	Recall
Q	0.69 ± 0.16	0.02 ± 0.03	0.31 ± 0.29
Q-MR	0.68 ± 0.17	0.02 ± 0.03	0.42 ± 0.33
Q-MP	0.68 ± 0.13	0.02 ± 0.02	0.23 ± 0.25
B-NR (SBM)	0.8 ± 0.14	0.1 ± 0.13	0.17 ± 0.15
B-NR (DC-SBM)	0.75 ± 0.15	0.16 ± 0.17	0.23 ± 0.18
cICL-HKK	0.78 ± 0.15	0.12 ± 0.13	0.2 ± 0.17
B-HKK	0.76 ± 0.14	0.08 ± 0.11	0.13 ± 0.11
Infomap	0.79 ± 0.17	0.19 ± 0.23	0.29 ± 0.2
MDL (SBM)	0.8 ± 0.16	0.16 ± 0.18	0.23 ± 0.17
MDL (DC-SBM)	0.81 ± 0.16	0.11 ± 0.16	0.19 ± 0.17
S-NB	0.74 ± 0.15	0.2 ± 0.18	0.21 ± 0.19
mean model-based	0.75 ± 0.16	0.11 ± 0.16	0.24 ± 0.22
mean indiv. topol.	0.63 ± 0.14	0.06 ± 0.13	0.17 ± 0.2
mean indiv. topol. & model	0.65 ± 0.15	0.07 ± 0.14	0.19 ± 0.21
emb-DW	0.68 ± 0.14	0.02 ± 0.03	0.3 ± 0.25
emb-vgae	0.7 ± 0.16	0.14 ± 0.14	0.2 ± 0.19
all topol.	0.82 ± 0.17	0.15 ± 0.15	0.24 ± 0.2
all model-based	0.84 ± 0.15	0.18 ± 0.16	0.28 ± 0.21
all embed.	0.79 ± 0.15	0.07 ± 0.07	0.19 ± 0.17
all topol. & model	0.84 ± 0.16	0.18 ± 0.17	0.25 ± 0.18
all topol. & embed.	0.82 ± 0.16	0.11 ± 0.1	0.22 ± 0.18
all model & embed.	0.84 ± 0.15	0.11 ± 0.09	0.25 ± 0.2
all topol., model & embed.	0.84 ± 0.15	0.15 ± 0.15	0.27 ± 0.23

Table S11. The detailed information of the top 5 topological predictors for synthetic data as presented in Fig. 2 in the manuscript.

Region	Model	Number of clusters k	Predictors
low ϵ	Poisson	1	[OE, DC_j, CC_i, $SPBC_i$, DC_i]
low ϵ	Poisson	2	[PPR, SP, dLRA-approx, mLRA-approx, LRA-approx]
low ϵ	Poisson	4	[PPR, SP, dLRA-approx, LRA-approx, mLRA-approx]
low ϵ	Poisson	16	[PPR, mLRA-approx, LRA-approx, dLRA-approx, SP]
low ϵ	Poisson	32	[PPR, SP, mLRA, LRA-approx, dLRA-approx]
low ϵ	Weibull	1	[PA, PR_i, KC_j, DC_j, PPR]
low ϵ	Weibull	2	[SP, PA, PPR, dLRA-approx, LRA-approx]
low ϵ	Weibull	4	[dLRA-approx, SP, PPR, mLRA-approx, LRA-approx]
low ϵ	Weibull	16	[dLRA-approx, PPR, SP, mLRA-approx, LRA-approx]
low ϵ	Weibull	32	[PPR, dLRA-approx, LRA-approx, CN, AA]
low ϵ	power law	1	[PA, LHN, dLRA, CN, RA]
low ϵ	power law	2	[PA, PR_i, AND_i, DC_i, LHN]
low ϵ	power law	4	[PA, PPR, PR_i, dLRA-approx, AA]
low ϵ	power law	16	[LRA-approx, dLRA-approx, PPR, mLRA-approx, PA]
low ϵ	power law	32	[PPR, dLRA-approx, LRA-approx, mLRA-approx, JC]
moderate ϵ	Poisson	1	[LC_i, PA, EC_i, SP, $SPBC_i$]
moderate ϵ	Poisson	2	[PPR, ND, AND_i, PA, dLRA-approx]
moderate ϵ	Poisson	4	[PPR, dLRA-approx, mLRA-approx, SP, LRA-approx]
moderate ϵ	Poisson	16	[mLRA-approx, dLRA-approx, LRA-approx, SP, PPR]
moderate ϵ	Poisson	32	[PPR, SP, LRA-approx, mLRA-approx, AA]
moderate ϵ	Poisson	1	[LC_i, PA, EC_i, SP, $SPBC_i$]
moderate ϵ	Weibull	2	[PA, SP, dLRA-approx, PPR, CN]
moderate ϵ	Weibull	4	[dLRA-approx, SP, PPR, mLRA-approx, PA]
moderate ϵ	Weibull	16	[PPR, SP, LRA-approx, dLRA-approx, mLRA-approx]
moderate ϵ	Weibull	32	[dLRA-approx, RA, dLRA, CN, PPR]
moderate ϵ	power law	1	[AND_i, PA, EC_i, AND_j, EC_j]
moderate ϵ	power law	2	[PA, dLRA-approx, AA, CN, RA]
moderate ϵ	power law	4	[PA, dLRA-approx, LHN, PPR, CN]
moderate ϵ	power law	16	[dLRA-approx, PPR, LRA-approx, SP, mLRA-approx]
moderate ϵ	power law	32	[PPR, CN, SP, PA, LHN]
high ϵ	Poisson	1	[LCC_j, ND, SP, dLRA, N]
high ϵ	Poisson	2	[EC_i, AA, N, SP, mLRA-approx]
high ϵ	Poisson	4	[LRA-approx, SP, PPR, PR_j, DC_j]
high ϵ	Poisson	16	[SP, PPR, mLRA-approx, CN, dLRA-approx]
high ϵ	Poisson	32	[PPR, mLRA-approx, SP, LRA-approx, RA]
high ϵ	Weibull	1	[SP, LRA-approx, KC_j, DC_j, KC_i]
high ϵ	Weibull	2	[PA, PPR, DC_j, dLRA-approx, LNT_j]
high ϵ	Weibull	4	[PA, dLRA-approx, PPR, DC_i, $SPBC_i$]
high ϵ	Weibull	16	[SP, PPR, PA, dLRA-approx, AA]
high ϵ	Weibull	32	[AA, RA, dLRA, dLRA-approx, CN]
high ϵ	power law	1	[DC_i, $SPBC_i$, LCC_i, PR_i, PA]
high ϵ	power law	2	[PA, dLRA-approx, AA, LHN, CN]
high ϵ	power law	4	[PA, AA, PPR, LHN, PR_j]
high ϵ	power law	16	[PPR, PA, dLRA-approx, CN, dLRA]
high ϵ	power law	32	[PPR, PA, dLRA, dLRA-approx, CN]

Table S12. The detailed information of the top 5 topological predictors for real data as presented in Fig. 2 in the manuscript.

Network set	Predictors
all domains	[SP, PPR, dLRA-approx, mLRA-approx, LRA-approx]
social	[PPR, CN, dLRA, SP, mLRA]
biological	[SP, PPR, PA, DC_i, dLRA-approx]
economic	[AND_i, PR_i, DC_i, DC_j, AND_j]
technological	[SP, PPR, PA, DC_i, DC_j]
information	[PA, PPR, SP, dLRA-approx, DC_i]
transportation	[SP, PPR, PA, DC_j, DC_i]

Table S13. Average AUC, precision, and recall performances of the link prediction algorithms over 124 social networks in our corpus. A random forest is used for supervised stacking of methods. Here, the predictors are adjusted for maximum F-measure using a model selection through a cross validation on training set. The results are reported on 20% holdout test set.

Algorithm	AUC	Precision	Recall
Q	0.89 ± 0.07	0.11 ± 0.07	0.42 ± 0.3
Q-MR	0.87 ± 0.07	0.16 ± 0.06	0.64 ± 0.23
Q-MP	0.86 ± 0.08	0.05 ± 0.05	0.35 ± 0.28
B-NR (SBM)	0.93 ± 0.06	0.34 ± 0.33	0.25 ± 0.13
B-NR (DC-SBM)	0.93 ± 0.07	0.29 ± 0.22	0.25 ± 0.12
clCL-HKK	0.93 ± 0.08	0.64 ± 0.31	0.37 ± 0.15
B-HKK	0.88 ± 0.07	0.24 ± 0.34	0.19 ± 0.16
Infomap	0.91 ± 0.04	0.26 ± 0.09	0.51 ± 0.19
MDL (SBM)	0.94 ± 0.07	0.48 ± 0.36	0.26 ± 0.15
MDL (DC-SBM)	0.93 ± 0.09	0.16 ± 0.11	0.25 ± 0.12
S-NB	0.94 ± 0.07	0.77 ± 0.22	0.43 ± 0.11
mean model-based	0.91 ± 0.08	0.32 ± 0.32	0.36 ± 0.23
mean indiv. topol.	0.65 ± 0.2	0.19 ± 0.31	0.3 ± 0.32
mean indiv. topol. & model	0.71 ± 0.21	0.22 ± 0.32	0.31 ± 0.31
emb-DW	0.95 ± 0.09	0.36 ± 0.14	0.46 ± 0.12
emb-vgae	0.95 ± 0.08	0.38 ± 0.24	0.33 ± 0.11
all topol.	0.98 ± 0.07	0.77 ± 0.21	0.74 ± 0.17
all model-based	0.97 ± 0.06	0.54 ± 0.27	0.47 ± 0.14
all embed.	0.96 ± 0.09	0.51 ± 0.19	0.5 ± 0.12
all topol. & model	0.98 ± 0.06	0.78 ± 0.21	0.73 ± 0.16
all topol. & embed.	0.97 ± 0.08	0.68 ± 0.22	0.65 ± 0.16
all model & embed.	0.97 ± 0.06	0.54 ± 0.2	0.53 ± 0.12
all topol., model & embed.	0.98 ± 0.07	0.69 ± 0.21	0.65 ± 0.15

Table S14. Average AUC, precision, and recall performances of the link prediction algorithms over 179 biological networks in our corpus. A random forest is used for supervised stacking of methods. Here, the predictors are adjusted for maximum F-measure using a model selection through a cross validation on training set. The results are reported on 20% holdout test set.

Algorithm	AUC	Precision	Recall
Q	0.61 ± 0.12	0.07 ± 0.13	0.29 ± 0.29
Q-MR	0.57 ± 0.11	0.1 ± 0.2	0.31 ± 0.3
Q-MP	0.59 ± 0.09	0.05 ± 0.08	0.25 ± 0.25
B-NR (SBM)	0.78 ± 0.13	0.16 ± 0.19	0.29 ± 0.24
B-NR (DC-SBM)	0.72 ± 0.17	0.22 ± 0.21	0.31 ± 0.23
cICL-HKK	0.74 ± 0.13	0.12 ± 0.14	0.32 ± 0.3
B-HKK	0.72 ± 0.14	0.11 ± 0.13	0.32 ± 0.28
Infomap	0.7 ± 0.12	0.09 ± 0.13	0.33 ± 0.25
MDL (SBM)	0.77 ± 0.14	0.18 ± 0.19	0.33 ± 0.28
MDL (DC-SBM)	0.82 ± 0.09	0.17 ± 0.2	0.28 ± 0.22
S-NB	0.72 ± 0.14	0.24 ± 0.23	0.27 ± 0.22
mean model-based	0.7 ± 0.15	0.14 ± 0.18	0.3 ± 0.26
mean indiv. topol.	0.6 ± 0.12	0.08 ± 0.13	0.29 ± 0.28
mean indiv. topol. & model	0.62 ± 0.13	0.09 ± 0.15	0.29 ± 0.28
emb-DW	0.59 ± 0.15	0.06 ± 0.12	0.36 ± 0.33
emb-vgae	0.62 ± 0.16	0.11 ± 0.15	0.31 ± 0.28
all topol.	0.85 ± 0.09	0.24 ± 0.23	0.31 ± 0.22
all model-based	0.84 ± 0.1	0.25 ± 0.23	0.31 ± 0.23
all embed.	0.72 ± 0.14	0.14 ± 0.2	0.27 ± 0.23
all topol. & model	0.86 ± 0.09	0.25 ± 0.22	0.32 ± 0.22
all topol. & embed.	0.83 ± 0.1	0.22 ± 0.22	0.3 ± 0.21
all model & embed.	0.83 ± 0.1	0.21 ± 0.21	0.31 ± 0.21
all topol., model & embed.	0.85 ± 0.1	0.23 ± 0.22	0.31 ± 0.21

Table S15. Average AUC, precision, and recall performances of the link prediction algorithms over 124 economic networks in our corpus. A random forest is used for supervised stacking of methods. Here, the predictors are adjusted for maximum F-measure using a model selection through a cross validation on training set. The results are reported on 20% holdout test set.

Algorithm	AUC	Precision	Recall
Q	0.71 ± 0.06	0.03 ± 0.13	0.14 ± 0.23
Q-MR	0.71 ± 0.06	0.01 ± 0.03	0.14 ± 0.21
Q-MP	0.54 ± 0.07	0.04 ± 0.15	0.09 ± 0.17
B-NR (SBM)	0.82 ± 0.08	0.06 ± 0.17	0.2 ± 0.3
B-NR (DC-SBM)	0.56 ± 0.1	0.09 ± 0.24	0.07 ± 0.13
clCL-HKK	0.83 ± 0.07	0.16 ± 0.33	0.12 ± 0.23
B-HKK	0.83 ± 0.06	0.05 ± 0.18	0.15 ± 0.22
Infomap	0.71 ± 0.05	0.03 ± 0.07	0.11 ± 0.15
MDL (SBM)	0.83 ± 0.05	0.09 ± 0.24	0.15 ± 0.23
MDL (DC-SBM)	0.88 ± 0.09	0.08 ± 0.18	0.12 ± 0.15
S-NB	0.54 ± 0.11	0.15 ± 0.32	0.2 ± 0.33
mean model-based	0.72 ± 0.14	0.07 ± 0.21	0.14 ± 0.23
mean indiv. topol.	0.59 ± 0.12	0.04 ± 0.15	0.14 ± 0.23
mean indiv. topol. & model	0.62 ± 0.14	0.04 ± 0.17	0.14 ± 0.23
emb-DW	0.38 ± 0.13	0.02 ± 0.1	0.18 ± 0.32
emb-vgae	0.56 ± 0.09	0.03 ± 0.11	0.17 ± 0.26
all topol.	0.89 ± 0.04	0.07 ± 0.14	0.17 ± 0.16
all model-based	0.9 ± 0.06	0.14 ± 0.28	0.15 ± 0.19
all embed.	0.78 ± 0.06	0.04 ± 0.08	0.13 ± 0.14
all topol. & model	0.91 ± 0.04	0.1 ± 0.23	0.16 ± 0.19
all topol. & embed.	0.89 ± 0.04	0.07 ± 0.19	0.19 ± 0.19
all model & embed.	0.91 ± 0.06	0.05 ± 0.12	0.21 ± 0.2
all topol., model & embed.	0.92 ± 0.05	0.06 ± 0.15	0.24 ± 0.23

Table S16. Average AUC, precision, and recall performances of the link prediction algorithms over 67 technological networks in our corpus. A random forest is used for supervised stacking of methods. Here, the predictors are adjusted for maximum F-measure using a model selection through a cross validation on training set. The results are reported on 20% holdout test set.

Algorithm	AUC	Precision	Recall
Q	0.63 ± 0.11	0.05 ± 0.13	0.13 ± 0.13
Q-MR	0.56 ± 0.11	0.03 ± 0.03	0.12 ± 0.16
Q-MP	0.62 ± 0.08	0.02 ± 0.02	0.18 ± 0.21
B-NR (SBM)	0.74 ± 0.11	0.08 ± 0.11	0.17 ± 0.16
B-NR (DC-SBM)	0.67 ± 0.12	0.13 ± 0.16	0.14 ± 0.15
clCL-HKK	0.75 ± 0.1	0.1 ± 0.16	0.14 ± 0.11
B-HKK	0.71 ± 0.11	0.08 ± 0.1	0.13 ± 0.14
Infomap	0.67 ± 0.13	0.04 ± 0.06	0.19 ± 0.17
MDL (SBM)	0.7 ± 0.15	0.13 ± 0.19	0.18 ± 0.15
MDL (DC-SBM)	0.77 ± 0.1	0.09 ± 0.11	0.14 ± 0.1
S-NB	0.65 ± 0.09	0.16 ± 0.19	0.13 ± 0.13
mean model-based	0.68 ± 0.13	0.08 ± 0.14	0.15 ± 0.15
mean indiv. topol.	0.58 ± 0.09	0.04 ± 0.11	0.14 ± 0.18
mean indiv. topol. & model	0.6 ± 0.11	0.05 ± 0.12	0.14 ± 0.18
emb-DW	0.65 ± 0.1	0.02 ± 0.03	0.16 ± 0.2
emb-vgae	0.64 ± 0.1	0.1 ± 0.13	0.16 ± 0.13
all topol.	0.8 ± 0.09	0.15 ± 0.16	0.14 ± 0.12
all model-based	0.78 ± 0.11	0.13 ± 0.19	0.16 ± 0.12
all embed.	0.73 ± 0.11	0.07 ± 0.13	0.13 ± 0.09
all topol. & model	0.82 ± 0.09	0.15 ± 0.17	0.15 ± 0.11
all topol. & embed.	0.78 ± 0.1	0.14 ± 0.18	0.14 ± 0.11
all model & embed.	0.78 ± 0.1	0.12 ± 0.19	0.15 ± 0.12
all topol., model & embed.	0.8 ± 0.09	0.14 ± 0.17	0.16 ± 0.11

Table S17. Average AUC, precision, and recall performances of the link prediction algorithms over 18 information networks as a subset of CommunityFitNet corpus. A random forest is used for supervised stacking of methods. Here, the predictors are adjusted for maximum F-measure using a model selection through a cross validation on training set. The results are reported on 20% holdout test set.

Algorithm	AUC	Precision	Recall
Q	0.61 ± 0.1	0.03 ± 0.02	0.26 ± 0.24
Q-MR	0.59 ± 0.1	0.05 ± 0.06	0.16 ± 0.13
Q-MP	0.59 ± 0.1	0.03 ± 0.03	0.17 ± 0.15
B-NR (SBM)	0.79 ± 0.14	0.18 ± 0.23	0.19 ± 0.1
B-NR (DC-SBM)	0.72 ± 0.14	0.19 ± 0.21	0.24 ± 0.2
clCL-HKK	0.8 ± 0.12	0.18 ± 0.22	0.27 ± 0.18
B-HKK	0.76 ± 0.13	0.17 ± 0.23	0.28 ± 0.2
Infomap	0.79 ± 0.08	0.08 ± 0.06	0.22 ± 0.17
MDL (SBM)	0.8 ± 0.13	0.2 ± 0.23	0.25 ± 0.17
MDL (DC-SBM)	0.81 ± 0.12	0.19 ± 0.24	0.21 ± 0.13
S-NB	0.7 ± 0.12	0.28 ± 0.28	0.19 ± 0.09
mean model-based	0.72 ± 0.15	0.14 ± 0.21	0.22 ± 0.17
mean indiv. topol.	0.62 ± 0.12	0.08 ± 0.15	0.2 ± 0.2
mean indiv. topol. & model	0.64 ± 0.13	0.1 ± 0.16	0.21 ± 0.19
emb-DW	0.62 ± 0.14	0.03 ± 0.02	0.26 ± 0.3
emb-vgae	0.64 ± 0.16	0.15 ± 0.12	0.17 ± 0.08
all topol.	0.84 ± 0.14	0.17 ± 0.09	0.28 ± 0.14
all model-based	0.84 ± 0.12	0.23 ± 0.24	0.28 ± 0.2
all embed.	0.78 ± 0.11	0.1 ± 0.06	0.2 ± 0.1
all topol. & model	0.85 ± 0.13	0.22 ± 0.21	0.27 ± 0.15
all topol. & embed.	0.84 ± 0.12	0.2 ± 0.21	0.24 ± 0.12
all model & embed.	0.84 ± 0.12	0.19 ± 0.21	0.24 ± 0.11
all topol., model & embed.	0.84 ± 0.14	0.21 ± 0.23	0.25 ± 0.11

Table S18. Average AUC, precision, and recall performances of the link prediction algorithms over 38 transportation networks as a subset of CommunityFitNet corpus. A random forest is used for supervised stacking of methods. Here, the predictors are adjusted for maximum F-measure using a model selection through a cross validation on training set. The results are reported on 20% holdout test set.

Algorithm	AUC	Precision	Recall
Q	0.68 ± 0.09	0.05 ± 0.16	0.14 ± 0.13
Q-MR	0.63 ± 0.08	0.03 ± 0.03	0.14 ± 0.11
Q-MP	0.63 ± 0.1	0.01 ± 0.01	0.15 ± 0.17
B-NR (SBM)	0.68 ± 0.14	0.11 ± 0.24	0.08 ± 0.09
B-NR (DC-SBM)	0.55 ± 0.23	0.13 ± 0.27	0.2 ± 0.28
cICL-HKK	0.69 ± 0.13	0.08 ± 0.18	0.16 ± 0.26
B-HKK	0.65 ± 0.13	0.07 ± 0.17	0.1 ± 0.12
Infomap	0.6 ± 0.13	0.06 ± 0.16	0.13 ± 0.1
MDL (SBM)	0.64 ± 0.15	0.05 ± 0.1	0.2 ± 0.3
MDL (DC-SBM)	0.81 ± 0.07	0.07 ± 0.17	0.1 ± 0.09
S-NB	0.66 ± 0.12	0.11 ± 0.19	0.1 ± 0.17
mean model-based	0.66 ± 0.15	0.07 ± 0.17	0.14 ± 0.19
mean indiv. topol.	0.59 ± 0.11	0.04 ± 0.15	0.14 ± 0.21
mean indiv. topol. & model	0.6 ± 0.12	0.05 ± 0.15	0.14 ± 0.21
emb-DW	0.62 ± 0.15	0.02 ± 0.02	0.17 ± 0.17
emb-vgae	0.67 ± 0.11	0.05 ± 0.09	0.17 ± 0.19
all topol.	0.86 ± 0.06	0.07 ± 0.08	0.14 ± 0.08
all model-based	0.82 ± 0.09	0.14 ± 0.24	0.11 ± 0.09
all embed.	0.75 ± 0.1	0.08 ± 0.16	0.12 ± 0.09
all topol. & model	0.86 ± 0.07	0.11 ± 0.18	0.12 ± 0.08
all topol. & embed.	0.81 ± 0.1	0.07 ± 0.07	0.12 ± 0.07
all model & embed.	0.8 ± 0.11	0.08 ± 0.13	0.12 ± 0.09
all topol., model & embed.	0.84 ± 0.09	0.08 ± 0.08	0.14 ± 0.1

Table S19. Average AUC performance of the link prediction supervised stacking methods over 550 networks as a subset of CommunityFitNet corpus. A random forest is used for supervised stacking of methods. Here, the predictors are adjusted for maximum AUC using a model selection through a cross validation on training set. The results are reported on 20% holdout test set.

Algorithm	AUC	Precision	Recall
all topol.	0.89 ± 0.09	0.3 ± 0.33	0.35 ± 0.29
all model-based	0.88 ± 0.1	0.26 ± 0.29	0.29 ± 0.22
all embed.	0.79 ± 0.15	0.19 ± 0.25	0.27 ± 0.23
all topol. & model	0.9 ± 0.09	0.32 ± 0.34	0.35 ± 0.29
all topol. & embed.	0.87 ± 0.11	0.26 ± 0.31	0.33 ± 0.25
all model & embed.	0.87 ± 0.11	0.22 ± 0.25	0.31 ± 0.23
all topol., model & embed.	0.89 ± 0.1	0.27 ± 0.3	0.34 ± 0.26

Table S20. Average AUC, precision, and recall performances of the link prediction algorithms over 550 networks as a subset of Community-FitNet corpus. A XGBoost is used for supervised stacking of methods. Here, the predictors are adjusted for maximum F-measure using a model selection through a cross validation on training set. The results are reported on 20% holdout test set.

Algorithm	AUC	Precision	Recall
all topol.	0.87 ± 0.1	0.32 ± 0.35	0.35 ± 0.32
all model-based	0.85 ± 0.12	0.27 ± 0.3	0.27 ± 0.21
all embed.	0.79 ± 0.15	0.21 ± 0.27	0.26 ± 0.23
all topol. & model	0.87 ± 0.11	0.33 ± 0.35	0.35 ± 0.32
all topol. & embed.	0.85 ± 0.13	0.27 ± 0.34	0.36 ± 0.31
all model & embed.	0.85 ± 0.13	0.23 ± 0.28	0.3 ± 0.24
all topol., model & embed.	0.86 ± 0.12	0.29 ± 0.34	0.35 ± 0.32

Table S21. Average AUC, precision, and recall performances of the link prediction algorithms over 550 networks as a subset of Community-FitNet corpus. A XGBoost is used for supervised stacking of methods. Here, the predictors are adjusted for maximum AUC using a model selection through a cross validation on training set. The results are reported on 20% holdout test set.

Algorithm	AUC	Precision	Recall
all topol.	0.88 ± 0.1	0.31 ± 0.35	0.37 ± 0.31
all model-based	0.87 ± 0.11	0.26 ± 0.3	0.28 ± 0.21
all embed.	0.79 ± 0.15	0.19 ± 0.26	0.29 ± 0.24
all topol. & model	0.88 ± 0.1	0.31 ± 0.35	0.36 ± 0.31
all topol. & embed.	0.86 ± 0.12	0.28 ± 0.34	0.36 ± 0.31
all model & embed.	0.86 ± 0.12	0.24 ± 0.28	0.3 ± 0.24
all topol., model & embed.	0.87 ± 0.12	0.3 ± 0.35	0.37 ± 0.31

Table S22. Average AUC, precision, and recall performances of the link prediction algorithms over 550 networks as a subset of Community-FitNet corpus. An AdaBoost is used for supervised stacking of methods. Here, the predictors are adjusted for maximum F-measure using a model selection through a cross validation on training set. The results are reported on 20% holdout test set.

Algorithm	AUC	Precision	Recall
all topol.	0.83 ± 0.13	0.33 ± 0.37	0.33 ± 0.31
all model-based	0.81 ± 0.15	0.26 ± 0.3	0.26 ± 0.22
all embed.	0.75 ± 0.15	0.17 ± 0.24	0.24 ± 0.21
all topol. & model	0.83 ± 0.14	0.32 ± 0.37	0.34 ± 0.32
all topol. & embed.	0.82 ± 0.14	0.28 ± 0.37	0.33 ± 0.31
all model & embed.	0.8 ± 0.14	0.21 ± 0.27	0.25 ± 0.22
all topol., model & embed.	0.81 ± 0.14	0.28 ± 0.36	0.32 ± 0.31

Table S23. Average AUC, precision, and recall performances of the link prediction algorithms over 550 networks as a subset of Community-FitNet corpus. An AdaBoost is used for supervised stacking of methods. Here, the predictors are adjusted for maximum AUC using a model selection through a cross validation on training set. The results are reported on 20% holdout test set.

Algorithm	AUC	Precision	Recall
all topol.	0.86 ± 0.11	0.32 ± 0.37	0.34 ± 0.31
all model-based	0.83 ± 0.14	0.25 ± 0.3	0.27 ± 0.22
all embed.	0.76 ± 0.15	0.16 ± 0.23	0.25 ± 0.22
all topol. & model	0.85 ± 0.12	0.31 ± 0.36	0.33 ± 0.31
all topol. & embed.	0.83 ± 0.13	0.29 ± 0.37	0.34 ± 0.31
all model & embed.	0.81 ± 0.14	0.21 ± 0.28	0.26 ± 0.23
all topol., model & embed.	0.83 ± 0.13	0.29 ± 0.36	0.32 ± 0.31

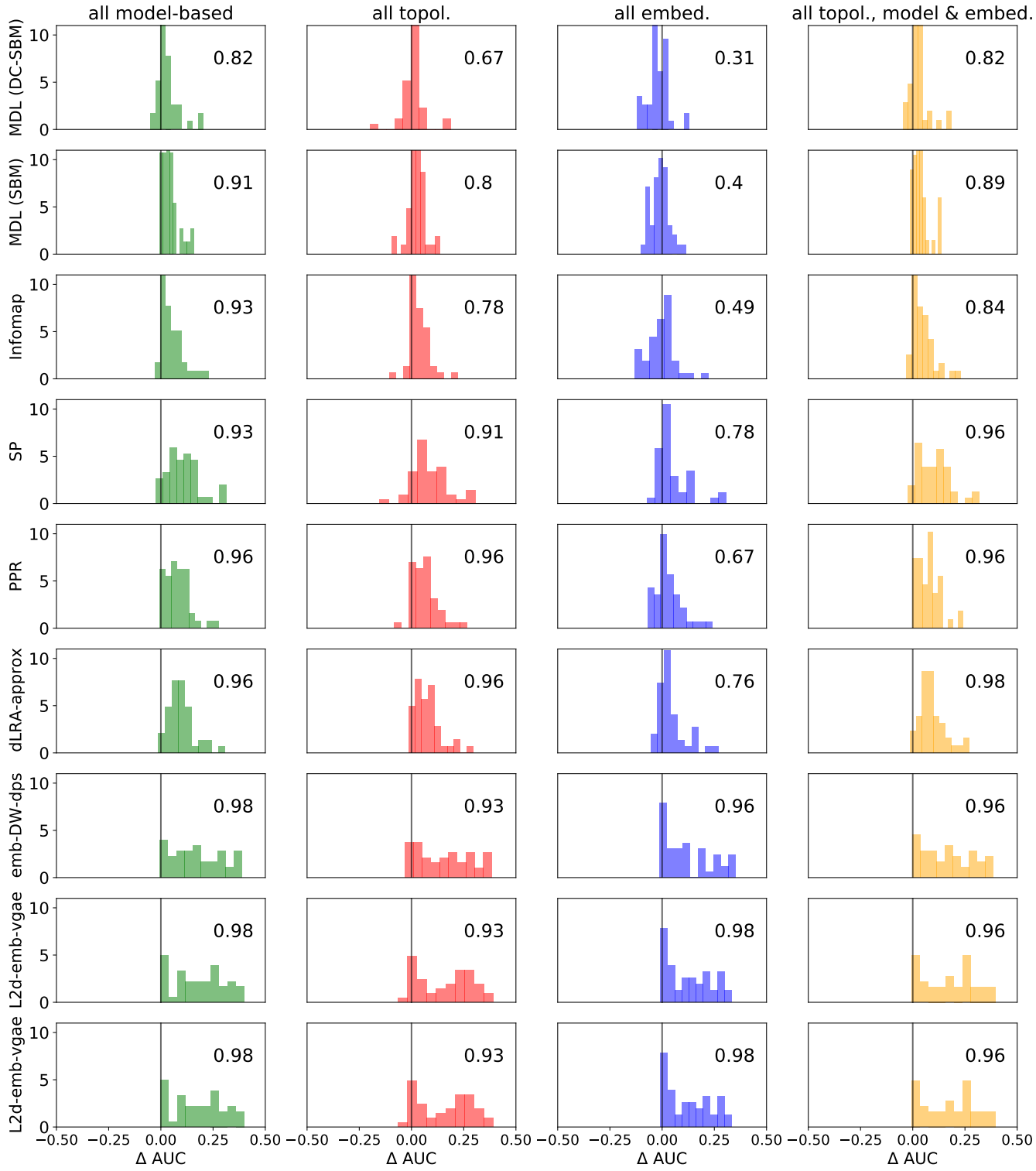


Fig. S10. The distribution of ΔAUC in comparison of stacking methods with the best individual predictors (3 model based, 3 topol., and 3 embedding) for synthetic data. The ΔAUC is the subtraction of AUC of one particular method (listed on the y-axis) from the AUC of the stacked models (listed in the column title). The reported fraction on each panel is the fraction of networks that has ΔAUC larger than zero which, i.e., the stacked model does better for them.

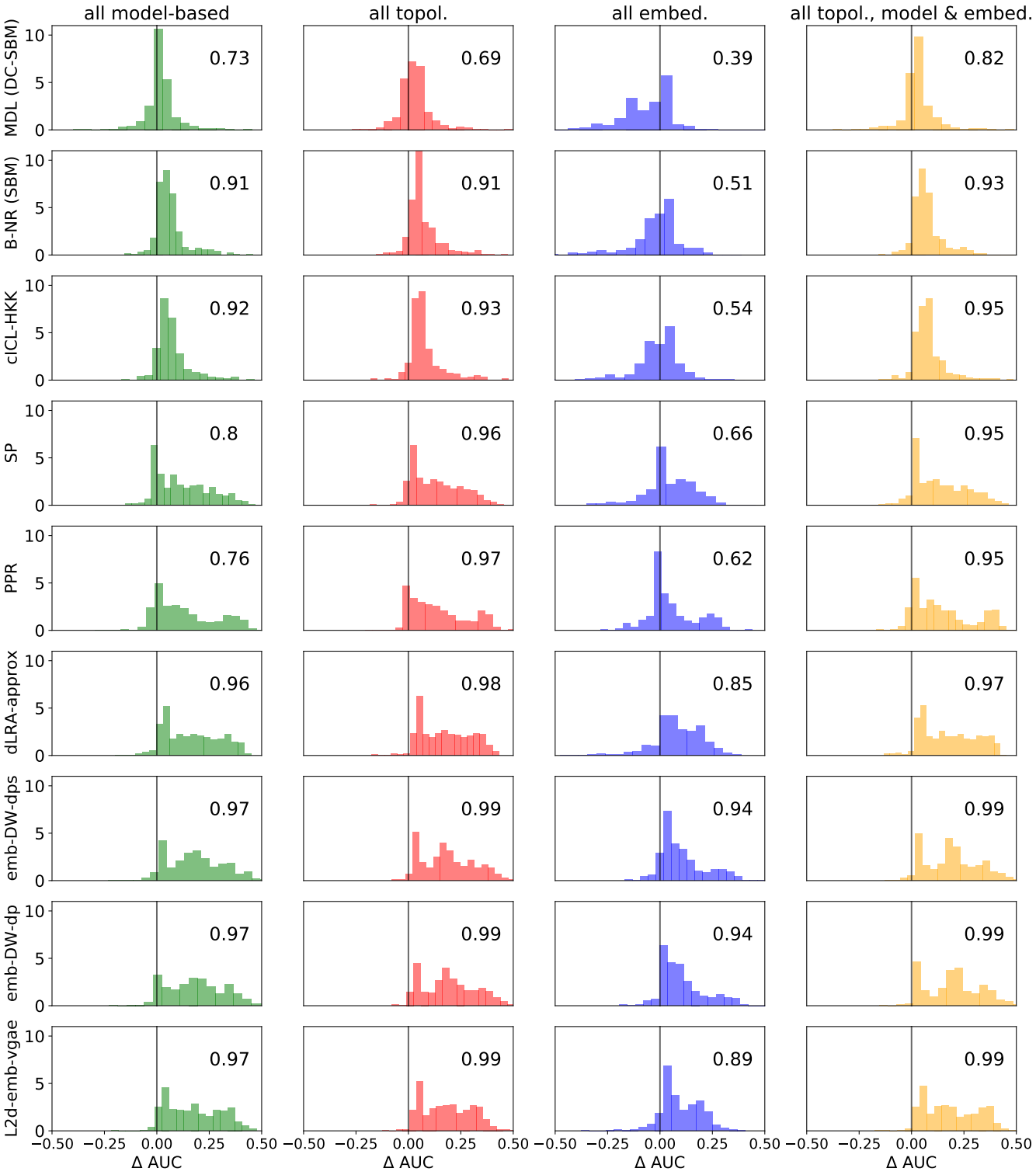


Fig. S11. The distribution of ΔAUC in comparison of stacking methods with the best individual predictors (3 model based, 3 topol., and 3 embedding) for real data. The ΔAUC is the subtraction of AUC of one particular method (listed on the y-axis) from the AUC of the stacked models (listed in the column title). The reported fraction on each panel is the fraction of networks that has ΔAUC larger than zero which, i.e., the stacked model does better for them.

F. Precision and Recall approximation

In this section, we provide equations to approximate precision and recall of link predictors from a uniform sample S. For computational consideration, we sample $10,000$ pairs with replacement of true positives and true negatives to compute the AUC. The sample S includes all unique entries from $10,000$ true positives and $10,000$ true negatives. We use pr and rc to represents precision and recall, respectively, and pr_s and rc_s to show the precision and recall computed for the sample S. Precision and recall can be computed for the population using Eqs. 16 and 17.

$$\text{pr} = \frac{TP}{TP + FP}\,, \tag{16}$$

$$\text{rc} = \frac{TP}{TP + FN}\,. \tag{17}$$

where, TP, FP, and FN are the number of true positives, the number of false positives, and the number of false negatives in the population, respectively.

However, for a sample S we need to consider some correction coefficients derived as follows. The true positive rate for the population and a uniform sample S noted as TPR and TPR_s, respectively, are almost equal. Therefore, we can write

$$TPR = \frac{TP}{P} \approx TPR_s = \frac{TP_s}{P_s},$$

where, P is the number of positives in the population, and P_s is the number of positives in the sample S. Then we have $TP \approx TP_s \times \alpha_s$, where, $\alpha_s = \frac{P}{P_s}$. Similarly, we have $FPR \approx FPR_s$ and following equation for FP,

$$FP \approx FP_s \times \beta_s,$$

where, $\beta_s = \frac{N}{N_s}$, N is the number of negative examples in the population, and N_s is the number of negative examples in the sample S. Similarly, for FN we have

$$FN \approx FN_s \times \alpha_s.$$

Using the above equations for TP, FP, and FN, we can approximate precision and recall from a sample S as Eqs. 18 and 19.

$$\text{pr} \approx \frac{TP_s\alpha_s}{TP_s\alpha_s + FP_s\beta_s}\,, \tag{18}$$

$$\begin{aligned}\text{rc} &\approx \frac{TP_s\alpha_s}{TP_s\alpha_s + FN_s\alpha_s}\,,\\ &\approx \frac{TP_s}{TP_s + FN_s}\,.\end{aligned} \tag{19}$$

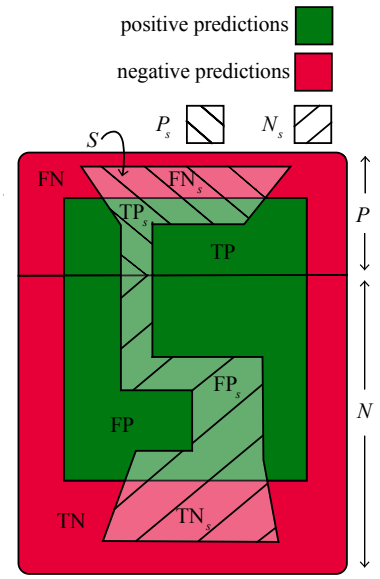


Fig. S12. A schematic representation of population with an imbalanced binary classification. For computational consideration, a sample S is drawn to compute the performance over different algorithms. The population is the whole space. The positive labels are located at the top of the figure (minority class), and the negative labels are located at the bottom of the figure (majority class). The green and red regions are the regions of positive and negative prediction by a binary classifier, respectively. The S-shape region is the region of samples drawn from the population. The shaded regions are the positive and negative prediction regions in the sample. For a binary classification, precision and recall can be approximated using the equations derived in section F.

G. Interpretability of the results coming from link prediction stacking algorithms

Based on the results of stacking methods over synthetic and real data, we found that the success of stacking methods depends on the difficulty of the problem. Through synthetic data, we have shown that model stacking is playing more positive role in settings with more heterogeneity and when clusters are more distinct. For sparser clusters with more homogeneity, the stacking approach is not better than the generative model itself, but clearly not worse (by controlling the overfitting). Therefore, stacking is a reliable method in different settings.

There are still some important questions to answer. For example, what are the crucial topological properties that guarantee the successful performance of stacking methods in a network? How we can choose a minimum topological feature set that can explain away the performance variablity of stacking methods on a given data? When stacking methods are not helpful in boosting up the performance, and what is the best practice in these settings? In fact, the No Free Lunch Theorem also applies to stacking algorithms. However, they can achieve higher performance on realistic inputs by effectively redistributing its worse-than-average performance onto unrealistic inputs, which are unlikely to be encountered in practice. We emphasize that our main goal in this manuscript is to develop and investigate the near-optimal performance of such an algorithm and learning this redistribution pattern would be a valuable direction of research in future. However, in order to add more interpretability to the results, we provided some illustrative examples with a discussion in the next few sections.

A. Minimal set of topological features equivalent with MDL (DC-SBM). As the first illustrative example, we would like to find a minimal set of topological features that their stacking performs equally well as MDL (DC-SBM), i.e., the best on average individual predictor. To this end, we choose a subset of networks, where MDL (DC-SBM) performs well ($\langle \text{AUC} \rangle > 0.8$). Then we build a sequence of sub-models in which we stack only the k^* ($k^* \leq 5$) most important predictors at a time and assess their performances on the test corpus, i.e., a very similar experiment with Fig. 4 in the manuscript. Only some of these networks can satisfy this constraint. The results coming from this experiment are presented in Fig. S13. We can see that most of the networks in this set are from social and biological domains.

Interestingly, in most of the social networks of our set, just one topological predictor can beat MDL (DC-SBM). The two best individual topological predictors are PPR and SP, in order. We can point out some of the reasons for good performance of topological features in social networks. In fact, there is not a specific generative model for real data, and although SBM is a good parametric model, topological features can better explain social networks in our experiments for three main reasons:

(i) many of topological features are originally generated to explain the interactions in social networks, and they are mostly originated from the characteristics of this domain; we can enumerate features like personalized page rank (PPR), LHN (Leicht-Holme-Newman index), common neighbor (CN), an extended version of it, resource allocation (RA) (40), and related quantities such as Adamic-Adar, Jaccard coefficient, which their Gini importances show their role in their good performances compared to more advanced algorithms such as SBM (see Fig. 1 in the main text),

(ii) in this paper, we only considered SBM with hard partitions, while mixed membership SBM (41) is a more realistic model to explain the interactions in real data; since through a hard partitioning we can explain just a subset of edges, but many of the edge-relations for each node in a social network belong to different partitions.

(iii) most of the social networks in our set have larger average degrees and stacking methods perform well in this region (see Fig. S14–S16).

From 135 biological networks where MDL (DC-SBM) performs well, in 61%, an equally well performance can be achieved by stacking at most 5 topological features. Among the top predictors, we can see features like PPR, PA, PR, and LRA-approx. In technological and transportation domains, among the networks that satisfy our constraint, an equally well performance can be achieved by stacking of a more diverse set of features. In a smaller fraction of networks, the top k^* topological features can perform equally well as MDL (DC-SBM) in economic and information networks that shows the shortage of good topological features in our set for these domains. These results suggest that developing more accurate topological predictors for non-social domains can be a good direction of research for future work.

In comparison of stacking methods with best individual model-based predictors (see Figs. S14–S15), we observe that stacking methods have marginally better performance (a margin of $\Delta\text{AUC} \geq 0.01$) over the networks with larger sizes and larger average degrees. To better visualize the dependency of performance on network size, looking at the AUCs versus average degrees and number of edges (Fig. S16), we can see that the individual predictors can outperform stacked models on smaller networks, where data for learning a stacked model is in short supply, while stacking methods outperform individual predictors significantly when the average degrees/number of edges increase.

B. Comparison of social networks with biological networks. In this section, we discuss the influencing factors in performance of stacking methods on the social networks versus the biological networks. To this end, we use an interpretable model (a simple linear regression model) and fit a regression of desired performance, AUC coming from all topol. & model stacking, on a list of important network features (here we chose AD, VD, ACC, NT, DA, SP, CN, AND, SPBC, and KC). Then, we choose the features with smaller p-values for further study and fit a locally weighted regression to the scatter plot of AUC versus each chosen feature. The fitted curves have been presented in Fig. S17. Note that due to the limited space, we have chosen only three of these crucial features to discuss. Some other topological features are correlated with the represented features and can be appropriate for our purpose. We leave a thorough exploration of these features and other domains for future work.

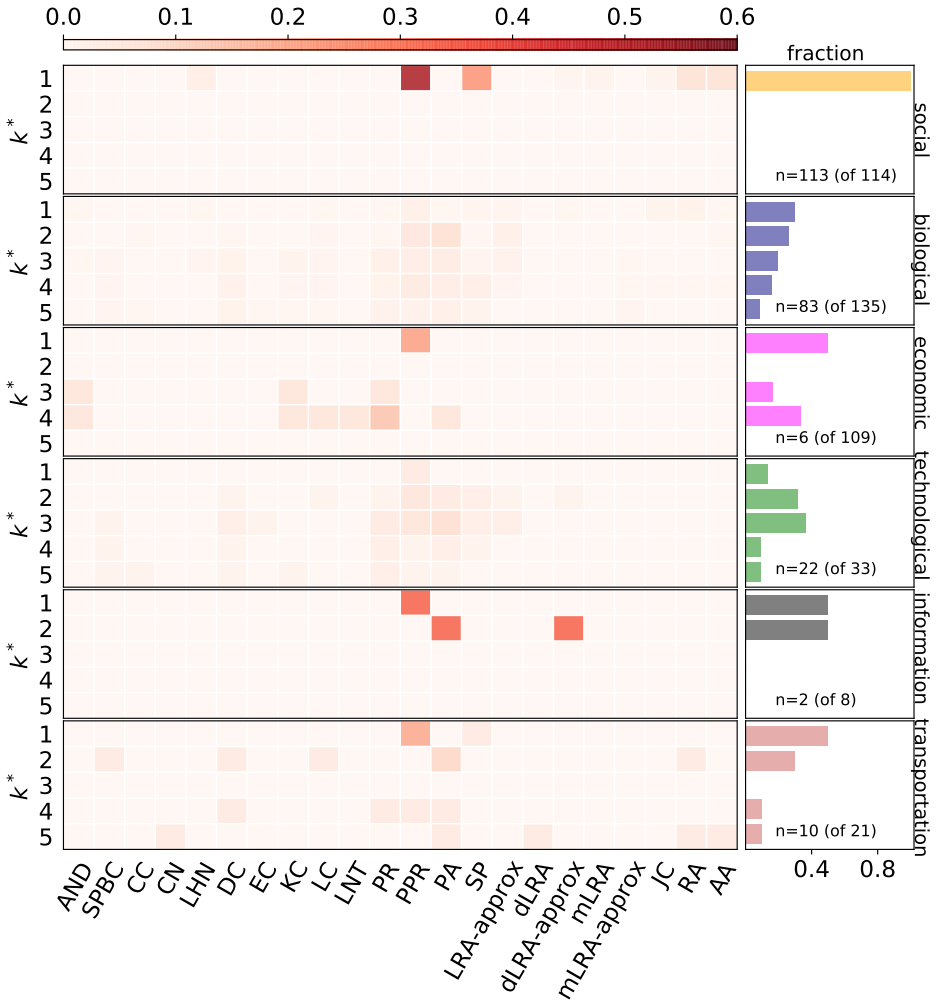


Fig. S13. The minimal set of k^* ($k^* \leq 5$) topological features for each domain that performs equally well as MDL (DC-SBM) on each network. The results are presented over a fraction of networks, where the MDL (DC-SBM) algorithm performs well on these networks ($\langle \mathrm{AUC} \rangle > 0.8$). On the left panels, at each row of each domain, we have the distribution of the minimal set for corresponding $k*$ when the weights are uniformly distributed and normalized at each panel, and on the right panels we have the distribution of k^* for each domain. The reported numbers in the right panels are the number of networks that stacking of at most k^* of their topological features can beat MDL (DC-SBM) out of the total number of networks with $\langle \mathrm{AUC} \rangle > 0.8$.

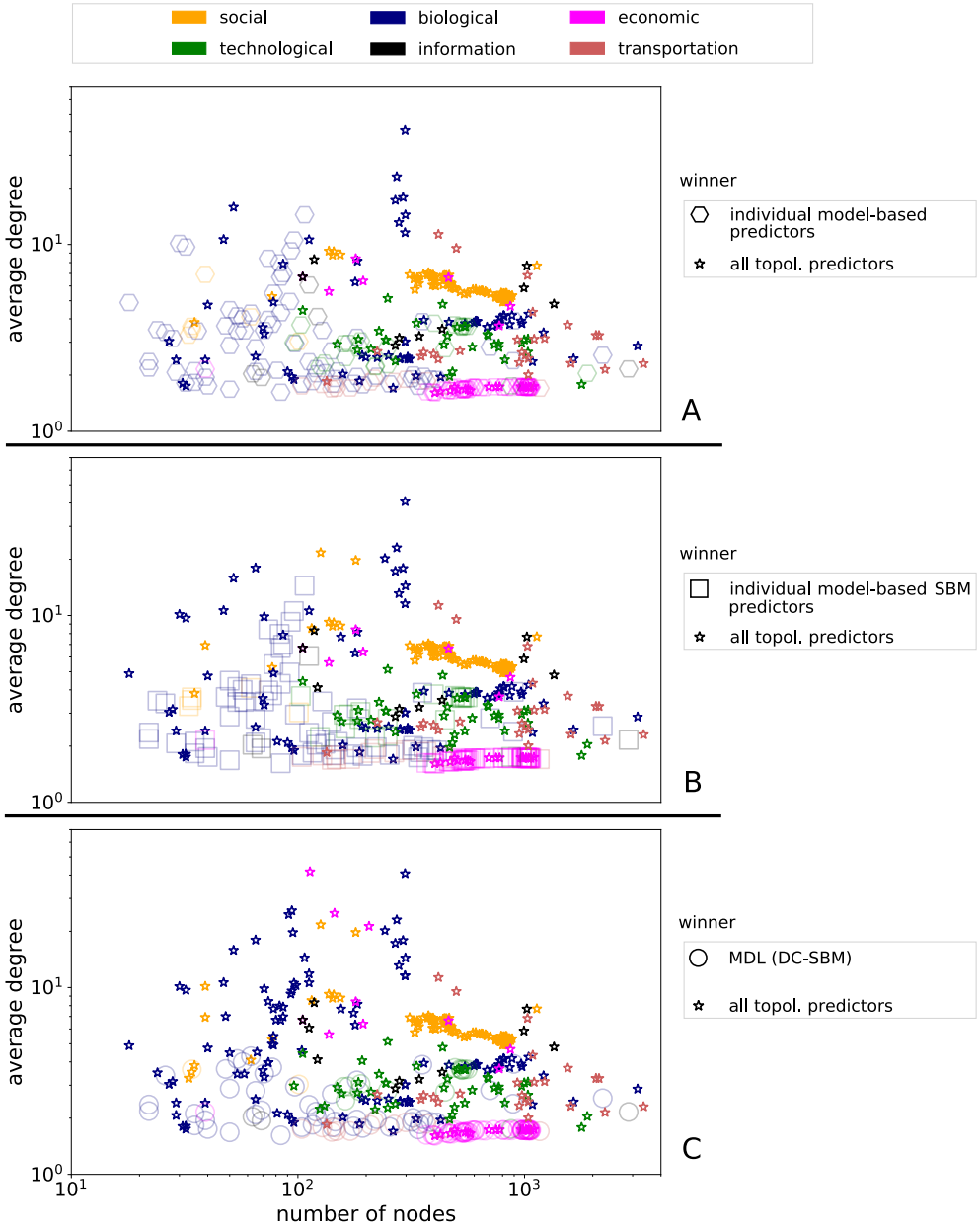
For social networks, we have chosen ACC, DA, and NT. We can see link prediction performance is positively correlated with these features. Based on these results we expect for locally dense networks (larger ACC), when similar degrees are connected together (larger DA), and when we have more triangles in the network (larger NT), predictability of missing links is improved. Note that, here we do not claim anything about causality, and we mean the predictability can be explained through these correlations. However, looking at the characteristics of the social networks, it can be perceived why increasing ACC, NT, and DA improves the predictability due to the triadic closure (42) for the first two, and homophily (43) for the latter in these networks. Interestingly, looking at Fig. S17, all topol. & model stacking method performs significantly well for social networks, when DA > 0.5 or NT > 0.73 (high performance with small standard deviation). This example again reveals that some social networks can be characterized with just one feature (Fig. S13).

For biological networks, link prediction is negatively correlated with SP and positively correlated with VD, and AD. On these networks, we expect for smaller average shortest path (SP < 0.35), more heterogeneous degree distributions (larger VD), and denser networks (larger AD) we have better performance. The correlation of AUC with AD confirms the importance of density of edges in better performance on these networks.

There can be several reasons that the link prediction in other domains are harder compare to social networks. To enumerate some:

(i) the corresponding topological features correlated with interactions in other domains are not known (e.g., in biological networks), are not tangible (e.g., in technological networks), and even can change over time (e.g., in economic networks) compared to social networks,

(ii) more researchers are working on social networks compare to other specific domains, and it is more explored in variety of



winner$_v$2.*pdf*

Fig. S14. (A) Comparison of link prediction performances (AUC) of individual model-based predictors versus stacking method of all topol. predictors. (B) Comparison of link prediction performances (AUC) of individual model-based SBM predictors (all model-based predictors from SBM family) versus stacking method of all topol. predictors. (C) Comparison of link prediction performances (AUC) of MDL (DC-SBM) versus stacking method of all topol. predictors. Here, any algorithm with a marginally better performance (a margin of $\Delta\text{AUC} \geq 0.01$) on a network, is considered to be "winner". Networks where neither predictor is a clear winner are omitted for clarity. Stacking of all topol. predictors shows better performance in comparison with any model-based predictors for larger networks (large N) with larger average degrees.

applications including link prediction,

(iii) networks in other domains are sparser (less information) compared to social networks which makes their link prediction harder (see Figs. S14–S16).

We think developing more specific domain-based topological features can help link prediction on other domains, and can be a valuable direction of research in future studies. As an example, in contrast with social networks that are more homogeneous with denser clusters, in biological networks increasing number of common neighbors decrease the probability of interaction between two proteins (44). From a protein-structure perspective, if two proteins have many common neighbors, they are more likely to have the same type of interaction interface, which reduces the probability of their own interaction. Therefore, Ref. (44) suggests the number of paths of length 3 plays a more important role in interactions among proteins, rather than paths of length 2.

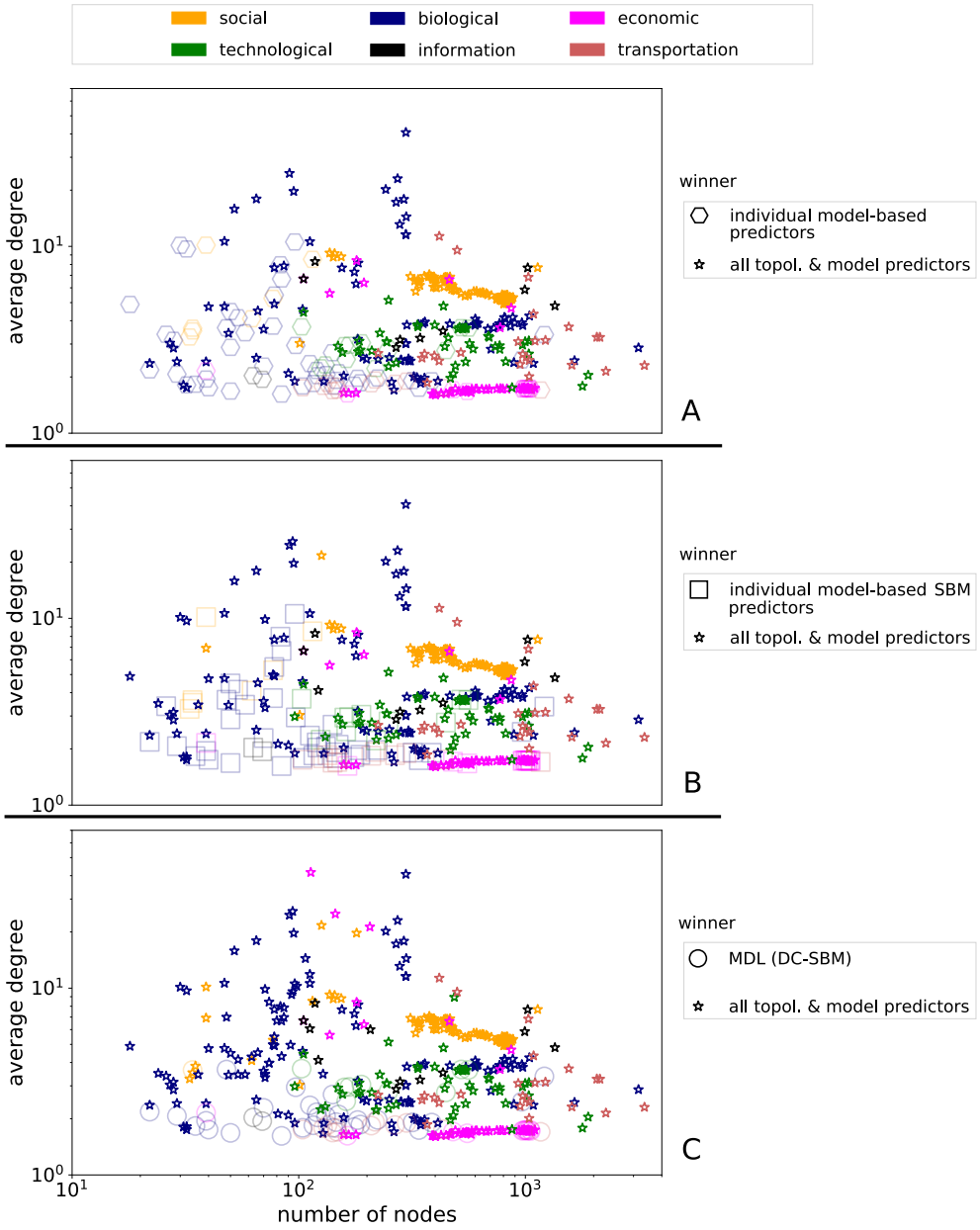


Fig. S15. (A) Comparison of link prediction performances (AUC) of individual model-based predictors versus stacking method of all topol. & model predictors. (B) Comparison of link prediction performances (AUC) of individual model-based SBM predictors (all model-based predictors from SBM family) versus stacking method of all topol. & model predictors. (C) Comparison of link prediction performances (AUC) of MDL (DC-SBM) versus stacking method of all topol. & model predictors. Here, any algorithm with a marginally better performance (a margin of $\Delta\text{AUC} \geq 0.01$) on a network, is considered to be "winner". Networks where neither predictor is a clear winner are omitted for clarity. Stacking of all topol. & model predictors shows better performance in comparison with any model-based predictors for larger networks (large N) and when networks have larger average degrees.

H. Theoretical analysis of stacking

Generally, the model comparison problems can fall into three categories based on the inclusion of the true generative model and existence of a clear belief model or not (45, 46); $\mathcal{M}$-closed, when the true generative model is included in the collection of models, $\mathcal{M}$-complete, when the true generative model is not included, however, the learning algorithms include tractable proxies of the true model, and $\mathcal{M}$-open, when the true model is not in the list of models and there is no actual specified belief model, perhaps due to time consideration or lack of expertise.

Since the true generative models for real-world networks are not known, $\mathcal{M}$-complete and $\mathcal{M}$-open contexts are more realistic in practice. Recently, stacking problem under these settings has been studied for non-relational data (47). Interpreting the stacking as a model combination using weights to optimize a cross-validation criterion that learns the best generalization, and through showing that these weights asymptotically minimize a posterior expected loss, the authors formally provide a Bayesian justification for cross validation. They also questioned an old belief that for smaller error in stacking, one needs more dissimilar

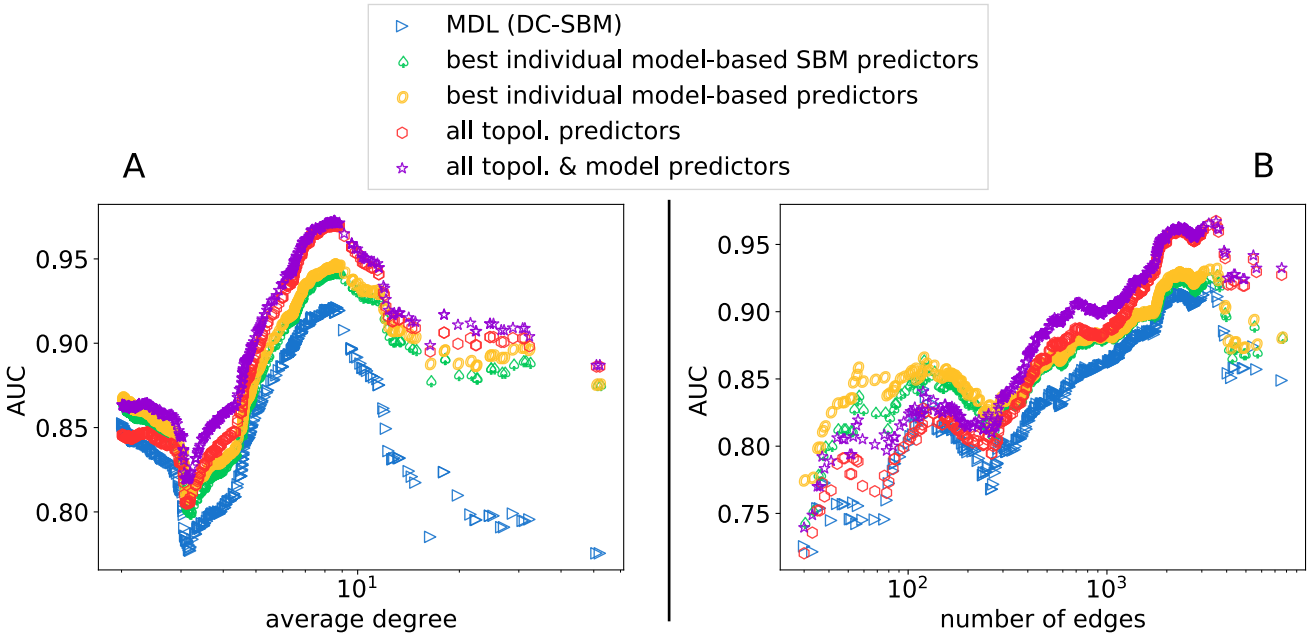


Fig. S16. Comparison of mean link prediction performance (AUC) for individual model-based predictors and best stacking methods as a function of (A) average degree, and (B) network size (number of edges m). The mean performance gap between individual predictors and stacking methods increases with network size.

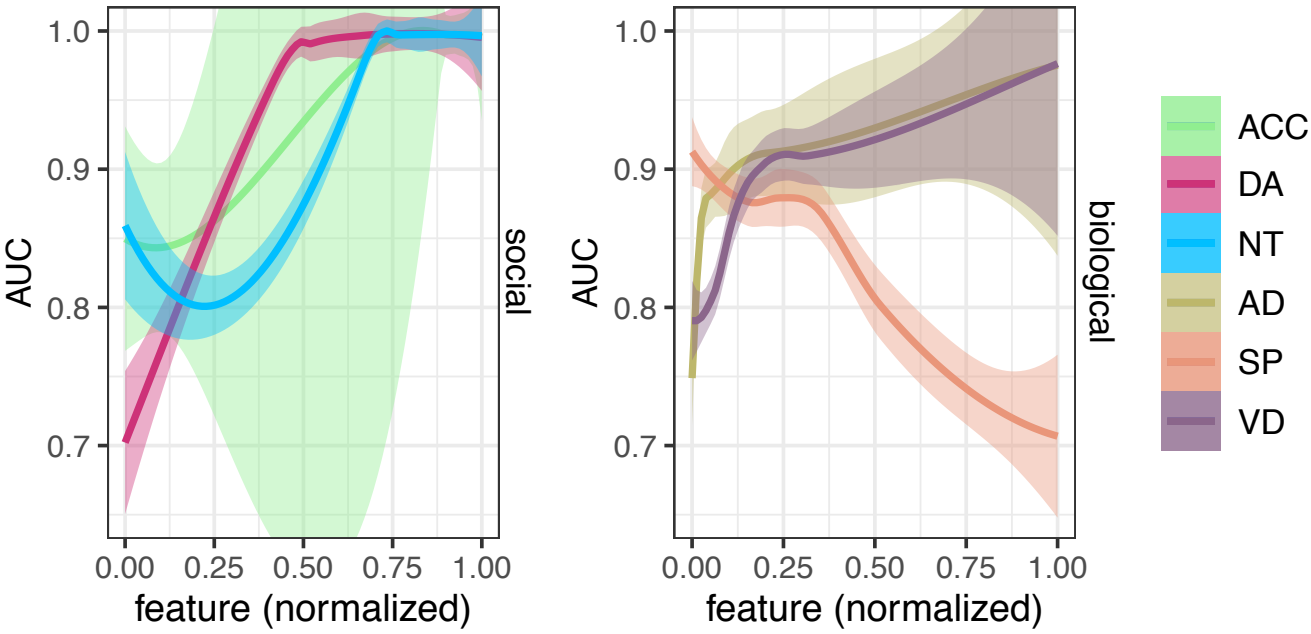


Fig. S17. The AUC performance of all topol. & model stacking method versus a variety of significant features chosen based on the p-values in a linear regression model. The curves present a fitted locally weighted regression to the scatter plots of AUC versus these crucial features.

predictors (48), and found that what matters in $\mathcal{M}$-complete context is the ingredients to be independent. In a more general setting in practice, i.e., in $\mathcal{M}$-open context, using different models but not necessarily independent gives us asymptotically optimal performance (47). Although these findings are for non-relational data, they are consistent with our results when we apply stacking on real-world networks.

A. Comparison of Bayesian model averaging and stacking. In Refs. (49, 50), the authors formalize the notion of a Bayesian model averaging (BMA) approach to link prediction by writing the posterior probability of missing links, given the observed network, as

Table S24. Link prediction performance (mean±S.D.) on holdout test set for link prediction algorithms applied to the 550 structurally diverse networks in our corpus. The results are presented to compare MDL (DC-SBM) averaging and stacking methods.

algorithm	AUC
MDL (DC-SBM)	0.85 ± 0.11
MDL (DC-SBM) averaging	0.87 ± 0.09
all topol. & model predictors	0.89 ± 0.09
all topol. & model predictors (including MDL (DC-SBM) averaging)	0.93 ± 0.09

$$P(\delta A \mid A^o) = \frac{P_{\delta A}(\delta A \mid A^o \cup \delta A) \sum_b P_G(A^o \cup \delta A \mid b) P(b)}{P(A^o)}$$
$$= \sum_b P_{\delta A}(\delta A \mid b, A^o) P(b \mid A^o)\,, \qquad [20]$$

where, A^o, δA, and b denote the matrix of observed interactions, the matrix of deleted interactions, and the model with partition b. Regardless of the theoretical relationship between BMA and model stacking in the setting of link prediction, it is clear that Eq. (20) is a perfectly reasonable approach, and the above papers do indeed show that it works relatively well, in practice.

However, theoretical work by Domingos (51) (for a discussion, see Ref. (52)), and Kim and Ghahramani (53) shows that BMA and model stacking are distinct algorithms, and they work in different ways. In particular, they show that model combination techniques like stacking work by enriching the space of hypotheses, not by approximating a Bayesian model average, and moreover that BMA is not optimal when the true data generating process is unknown. To quote Kim and Ghahramani (53):

> *[BMA] is only valid if we believe that the K classifiers capture mutually exclusive and exhaustive possibilities about how the data was generated. In fact, we might not believe at all that any of the K classifiers reflects the true data generation.*

Of course, BMA can be a very powerful technique, however, in our setting it behaves more like model selection than like model combination. For instance, consider the limiting case of a large amount of data, and hence a sharply peaked posterior, and denote the true data generating process as b^*. If b^* is among the "base" predictors, then the true model posterior $P(b^* \mid A^o)$ in Eq. 20 approaches 1, and the Bayesian model average approach yields optimal predictions. On other hand, when b^* is not among the base predictors, the model posterior concentrates on the closest model within the set, and that model achieves a posterior that approaches 1. In other words, in this setting, we should expect BMA to behave like model selection, rather than model combination. Hence, a BMA approach to link prediction should not be interpreted as a form of model stacking—they are different algorithms.

At the same time, model stacking does now have a Bayesian interpretation, as thoroughly discussed in a few recent papers (46, 47, 54, 55), in which stacking combines models so that the stacked weights asymptotically minimize a posterior expected loss. Yao et al. (46) show that the practical difference between BMA and model stacking is that the former is not optimized for the same predictive task as the latter, and that weights learned by BMA reflect the fit to the data, rather than evaluating the prediction accuracy. This behavior means that results of BMA for link prediction do not provide any information about how well or poorly a model stacking approach will perform.

Nevertheless, in this section we compare our model stacking approach to a BMA approach. To address this interesting point of comparison, we carried out a new numerical experiment using Bayesian modeling averaging under the MDL (DC-SBM) model over our entire corpus of 550 real-world networks. These results are shown in Fig. S18 and Table S24. We find that BMA does perform better than stacking for small networks, but performs worse on larger networks, with stacking being better on average for networks with $m > 300$ edges.

That said, one great advantage of model stacking is that its set of "base" predictors is easily extended to include new predictors. As an exercise, we have trained a new stacked model that includes the Bayesian model average predictor as a member of the ensemble, alongside our existing set of predictors. The results for this extended stacked model are shown in Fig. S18 and Table S24, and, we are pleased to note, it performs even better, $\Delta\text{AUC} = 0.04$, than our previous stacked model.

References

1. Liben-Nowell D, Kleinberg J (2007) The link-prediction problem for social networks. *Journal of the Association for Information Science and Technology* 58(7):1019–1031.
2. Newman M (2019) *Networks.* (Oxford University Press).
3. Al Hasan M, Zaki MJ (2011) A survey of link prediction in social networks in *Social Network Data Analytics.* (Springer), pp. 243–275.
4. Cukierski W, Hamner B, Yang B (2011) Graph-based features for supervised link prediction in *Neural Networks (IJCNN), The 2011 International Joint Conference on.* (IEEE), pp. 1237–1244.

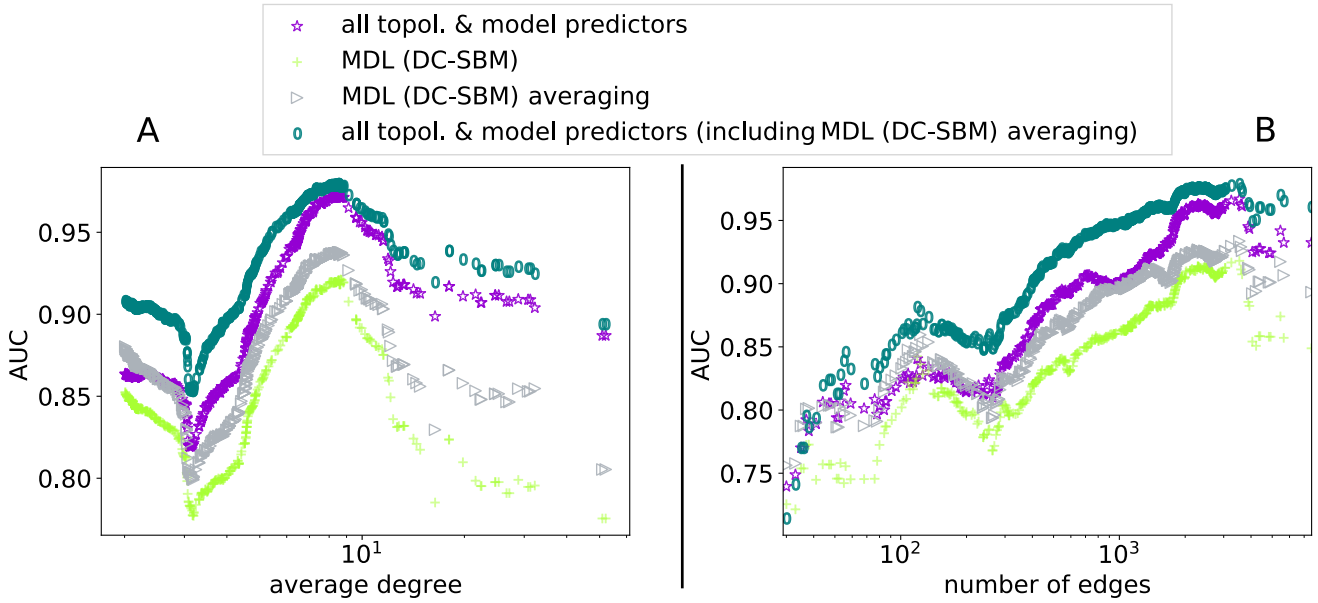


Fig. S18. Mean link prediction performance (AUC) as a function of (A) average degree, (B) number of edges, for MDL (DC-SBM), MDL (DC-SBM) averaging, all topol. & model predictors, all topol. & model predictors including MDL (DC-SBM) averaging, applied to 550 real-world networks. The MDL (DC-SBM) averaging performs better for smaller amount of data (average degrees and number of edges) in comparison with larger amount of data.

5. Hagberg A, Swart P, S Chult D (2008) Exploring network structure, dynamics, and function using networkx, (Los Alamos National Lab.(LANL), Los Alamos, NM (United States)), Technical report.
6. Leicht EA, Holme P, Newman MEJ (2006) Vertex similarity in networks. *Physical Review E* 73(2):026120.
7. Ghasemian A, Hosseinmardi H, Clauset A (2019) Evaluating overfit and underfit in models of network community structure. *IEEE Trans. Knowledge and Data Engineering (TKDE)*.
8. Newman MEJ, Girvan M (2004) Finding and evaluating community structure in networks. *Phys. Rev. E* 69(2):026113.
9. Jeub LGS, Bazzi M, Jutla IS, Mucha PJ (2011-2019) A generalized Louvain method for community detection implemented in MATLAB (https://github.com/GenLouvain/GenLouvain).
10. Newman MEJ (2016) Community detection in networks: Modularity optimization and maximum likelihood are equivalent. *arXiv:1606.02319*.
11. Zhang P, Moore C (2014) Scalable detection of statistically significant communities and hierarchies, using message passing for modularity. *Proc. Natl. Acad. Sci. USA* 111(51):18144–18149.
12. Newman MEJ, Reinert G (2016) Estimating the number of communities in a network. *Phys. Rev. Lett.* 117(7):078301.
13. Newman MEJ (2016) An implementation of Bayesian stochastic block model algorithm to infer number of communities and communities explained in Newman and Reinert, phys. rev. lett. 117(7), 078301 (2016) (http://www-personal.umich.edu/~mejn/communities/).
14. Hayashi K, Konishi T, Kawamoto T (2016) A tractable fully Bayesian method for the stochastic block model. *arXiv:1602.02256*.
15. Hayashi K, Konishi T, Kawamoto T (2016) An implementation of a tractable fully Bayesian method for the stochastic block model in python (https://github.com/hayasick/fabsbm).
16. Rosvall M, Bergstrom CT (2008) Maps of random walks on complex networks reveal community structure. *Proc. Natl. Acad. Sci. USA* 105(4):1118–1123.
17. Edler D, Eriksson A, Rosvall M (2015) The MapEquation software package (http://www.mapequation.org).
18. Peixoto TP (2013) Parsimonious module inference in large networks. *Phys. Rev. Lett.* 110(14):148701.
19. Peixoto TP (2014) The graph-tool python library. *figshare*.
20. Krzakala F, et al. (2013) Spectral Redemption in Clustering Sparse Networks. *Proc. Natl. Acad. Sci.* 110(52):20935–20940.
21. Perozzi B, Al-Rfou R, Skiena S (2014) Deepwalk: Online learning of social representations in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. (ACM), pp. 701–710.
22. Grover A, Leskovec J (2016) node2vec: Scalable feature learning for networks in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. (ACM), pp. 855–864.
23. Kipf TN, Welling M (2016) Variational graph auto-encoders. *preprint arXiv:1611.07308*.
24. Hamilton WL, Ying R, Leskovec J (2017) Representation learning on graphs: Methods and applications. *preprint arXiv:1709.05584*.
25. Breiman L (1996) Bagging predictors. *Machine Learning* 24(2):123–140.
26. Schapire RE (1999) A brief introduction to boosting in *Proceedings of the 16th International Joint Conference on Artificial intelligence, Volume 2*. (Morgan Kaufmann Publishers Inc.), pp. 1401–1406.

27. Dietterich T (2000) Ensemble methods in machine learning. *Multiple Classifier Systems* pp. 1–15.
28. Sewell M (2008) Ensemble learning. *RN* 11(02).
29. Wolpert DH (1992) Stacked generalization. *Neural Networks* 5(2):241–259.
30. Chen T, Guestrin C (2016) Xgboost: A scalable tree boosting system in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. pp. 785–794.
31. Freund Y, Schapire RE (1997) A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences* 55(1):119–139.
32. Karrer B, Newman ME (2011) Stochastic blockmodels and community structure in networks. *Physical review E* 83(1):016107.
33. Decelle A, Krzakala F, Moore C, Zdeborová L (2011) Asymptotic Analysis of the Stochastic Block Model for Modular Networks and Its Algorithmic Applications. *Phys. Rev. E* 84(6):066106.
34. Clauset A, Tucker E, Sainz M (2016) The Colorado Index of Complex Networks. (https://icon.colorado.edu/).
35. Al Hasan M, Chaoji V, Salem S, Zaki M (2006) Link prediction using supervised learning in *SDM06: workshop on link analysis, counter-terrorism and security*.
36. Ahmed C, ElKorany A, Bahgat R (2016) A supervised learning approach to link prediction in twitter. *Social Network Analysis and Mining* 6(1):24.
37. Lichtenwalter RN, Lussier JT, Chawla NV (2010) New perspectives and methods in link prediction in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. (ACM), pp. 243–252.
38. Cover TM, Thomas JA (2012) *Elements of information theory*. (John Wiley & Sons).
39. Pihur V, Datta S, Datta S (2007) Weighted rank aggregation of cluster validation measures: a monte carlo cross-entropy approach. *Bioinformatics* 23(13):1607–1615.
40. Zhou T, Lü L, Zhang YC (2009) Predicting missing links via local information. *The European Physical Journal B* 71(4):623–630.
41. Airoldi EM, Blei DM, Fienberg SE, Xing EP (2008) Mixed membership stochastic blockmodels. *Journal of Machine Learning Research* 9(Sep):1981–2014.
42. Easley D, Kleinberg J (2010) *Networks, crowds, and markets*. (Cambridge university press Cambridge).
43. McPherson M, Smith-Lovin L, Cook JM (2001) Birds of a feather: Homophily in social networks. *Annual review of sociology* 27(1):415–444.
44. Kovács IA, et al. (2019) Network-based prediction of protein interactions. *Nature communications* 10(1):1–8.
45. Bernardo JM, Smith AF (2009) *Bayesian theory*. (John Wiley & Sons) Vol. 405.
46. Yao Y, Vehtari A, Simpson D, Gelman A (2018) Using stacking to average bayesian predictive distributions. *Bayesian Analysis* 13(3):917–1007.
47. Le T, Clarke B (2017) A bayes interpretation of stacking for $\mathcal{M}$-complete and $\mathcal{M}$-open settings. *Bayesian Analysis* 12(3):807–829.
48. Breiman L (1996) Stacked regressions. *Machine learning* 24(1):49–64.
49. Guimerà R, Sales-Pardo M (2009) Missing and spurious interactions and the reconstruction of complex networks. *Proceedings of the National Academy of Sciences* 106(52):22073–22078.
50. Vallès-Català T, Peixoto TP, Sales-Pardo M, Guimerà R (2018) Consistencies and inconsistencies between model selection and link prediction in networks. *Physical Review E* 97(6):062316.
51. Domingos P (1997) Why does bagging work? a bayesian account and its implications in *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*. pp. 155–158.
52. Minka TP (2000) Bayesian model averaging is not model combination. pp. 1–2. MIT Media Lab note. Available electronically at https://tminka.github.io/papers/bma.html.
53. Kim HC, Ghahramani Z (2012) Bayesian classifier combination in *Prof. Artificial Intelligence and Statistics (AISTATS)*. pp. 619–627.
54. Yao Y (2019) Bayesian aggregation. *preprint arXiv:1912.11218*.
55. Clyde M, Iversen ES (2013) Bayesian model averaging in the $\mathcal{M}$-open framework. *Bayesian theory and applications* pp. 483–498.