Stacking Models for Nearly Optimal Link Prediction in Complex Networks

Amir Ghasemian[a,b,c], **Homa Hosseinmardi**[b], **Aram Galstyan**[b], **Edoardo M. Airoldi**[c,d], **and Aaron Clauset**[a,e,f]

[a]Department of Computer Science, University of Colorado, Boulder, CO 80309, USA; [b]Information Sciences Institute, University of Southern California, Marina del Rey, CA 90292, USA; [c]Department of Statistics, Harvard University, Cambridge, MA 02138, USA; [d]Department of Statistical Science, Fox School of Business, Temple University, Philadelphia, PA 19122, USA; [e]BioFrontiers Institute, University of Colorado, Boulder, CO 80309, USA; [f]Santa Fe Institute, Santa Fe, NM 87501, USA

Most real-world networks are incompletely observed. Algorithms that can accurately predict which links are missing can dramatically speedup the collection of network data and improve the validity of network models. Many algorithms now exist for predicting missing links, given a partially observed network, but it has remained unknown whether a single best predictor exists, how link predictability varies across methods and networks from different domains, and how close to optimality current methods are. We answer these questions by systematically evaluating 203 individual link predictor algorithms, representing three popular families of methods, applied to a large corpus of 548 structurally diverse networks from six scientific domains. We first show that individual algorithms exhibit a broad diversity of prediction errors, such that no one predictor or family is best, or worst, across all realistic inputs. We then exploit this diversity via meta-learning to construct a series of "stacked" models that combine predictors into a single algorithm. Applied to a broad range of synthetic networks, for which we may analytically calculate optimal performance, these stacked models achieve optimal or nearly optimal levels of accuracy. Applied to real-world networks, stacked models are also superior, but their accuracy varies strongly by domain, suggesting that link prediction may be fundamentally easier in social networks than in biological or technological networks. These results indicate that the state-of-the-art for link prediction comes from combining individual algorithms, which achieves nearly optimal predictions. We close with a brief discussion of limitations and opportunities for further improvement of these results.

networks | link prediction | meta-learning | stacking | near optimality

Networks provide a powerful abstraction for representing the structure of complex social, biological, and technological systems. However, data on most real-world networks is incomplete. For instance, social connections among people may be sampled, intentionally hidden, or simply unobservable (1, 2); interactions among genes, or cells, or species must be observed or inferred by expensive experiments (3, 4); and, connections mediated by a particular technology omit all off-platform interactions (2, 5). The presence of such "missing links" can, depending on the research question, dramatically alter scientific conclusions when analyzing a network's structure or modeling its dynamics.

Methods that accurately predict which observed pairs of unconnected nodes should, in fact, be connected have broad utility. For instance, they can improve the accuracy of predictions of future network structure and minimize the use of scarce experimental or network measurement resources (6, 7). Moreover, the task of link prediction itself has become a standard for evaluating and comparing models of network structure (8, 9), playing a role in networks that is similar to that of cross-validation in traditional statistical learning (10, 11). Hence, by helping to select more accurate network models (8), methods for link prediction can shed light on the organizing principles of complex systems of all kinds.

But, predicting missing links is a statistically hard problem. Most real-world networks are relatively sparse, and the number of unconnected pairs in an observed network—each a potential missing link—grows quadratically, like $O(n^2)$ for a network with n nodes when the number of connected pairs or edges m grows linearly, like $O(n)$. The probability of correctly choosing by chance a missing link is thus only $O(1/n)$—an impractically small chance even for moderate-sized systems (12). Despite this baseline difficulty, a plethora of link prediction methods exist (3, 13, 14), embodied by the three main families we study here: (i) topological methods (15, 16), which utilize network measures like node degrees, the number of common neighbors, and the length of a shortest path; (ii) model-based methods (8, 12), such as the stochastic block model, its variants, and other models of community structure; and (iii) embedding methods (17, 18), which project a network into a latent space and predict links based on the induced proximity of its nodes.

A striking feature of this array of methods is that all appear to work relatively well (8, 15, 17). However, systematic comparisons are lacking, particularly of methods drawn from different families, and most empirical evaluations are based on relatively small numbers of networks. As a result, the general accuracy of different methods remains unclear, and we do not know whether different methods, or families, are capturing the same underlying signatures of "missingness." For instance, is there a single best method or family for all circumstances? If not, then how does missing link predictability vary across methods and scientific domains, e.g., in social versus biological networks, or across network scales? And, how close to optimality are current methods?

Here, we answer these questions using a large corpus of 548 structurally and scientifically diverse real-world networks and 203 missing link predictors drawn from three large methodological families. First, we show that individual methods exploit different underlying signals of missingness, and, affirming the practical relevance of the No Free Lunch theorem (19, 20), no method performs best or worst on all realistic inputs. We then show that a meta-learning approach (21–23) can exploit this diversity of errors by "stacking" individual methods into a single algorithm (24), which we argue makes nearly optimal predictions of missing links. We support this claim with three lines of evidence: (i) evaluations on synthetic data with known structure and optimal performance, (ii) tests using real-world networks across scientific domains and network scales, and (iii) tests of sufficiency and saturation using subsets of methods. Across these tests, model stacking is nearly always the best method on held-out links, and nearly-optimal performance can be constructed using model-based methods, topological methods, or a mixture of the two. Furthermore, we find that

missing links are generally easiest to predict in social networks, where most methods perform well, and hardest in biological and technological networks. We conclude by discussing limitations and opportunities for further improvement of these results.

Methods and Materials

As a general setting, we imagine an unobserved simple network G with a set of E pairwise connections among a set of V nodes, with sizes m and n, respectively. Of these, a subset $E' \subset E$ of connections is observed, chosen by some function f. Our task is to accurately guess, based only on the pattern of observed edges E', which unconnected pairs $X = V \times V - E'$ are in fact among the missing links $Y = E - E'$. A link prediction method defines a *score* function over these unconnected pairs $i, j \in X$ so that better-scoring pairs are more likely to be missing links (15). In a supervised setting, the particular function that combines input predictors to produce a score is learned from the data.We evaluate the accuracy of such predictions using the standard AUC statistic, which provides a context-agnostic measure of a method's ability to distinguish a missing link $i, j \in Y$ (a true positive) from a non-edge $X - Y$ (a true negative) (12). Other accuracy measures may provide insight about a predictor's performance in specific settings, e.g., precision and recall at certain thresholds. We leave their investigation for future work.

The most common approach to predict missing links constructs a score function from network statistics of each unconnected node pair (15). We study 42 of these topological predictors, which include predictions based on node degrees, common neighbors, random walks, node and edge centralities, among others (see SI Appendix, Table S1). Models of large-scale network structure are also commonly used for link prediction. We study 11 of these model-based methods (8), which either estimate a parametric probability $\Pr(i \to j \,|\, \theta)$ that a node pair is connected (12), given a decomposition of a network into communities, or predict a link as missing if it would improve a measure of community structure (15) (see SI Appendix, Table S2). Close proximity of an unconnected pair, after embedding a network's nodes into a latent space, is a third common approach to link prediction. We study 150 of these embedding-based predictors, derived from two popular graph embedding algorithms and six notions of distance or similarity in the latent space. In total, we consider 203 features of node pairs, some of which are the output of existing link prediction algorithms, while others are numerical features derived from the network structure. For our purposes, each is considered a missing link "predictor." A lengthier description of these 203 methods, and the three methodological families they represent, is given in SI Appendix, section A.

Meta-learning techniques are a powerful class of machine learning algorithms that can learn from data how to combine individual predictors into a single, more accurate algorithm (22, 25). Stacked generalization (24) combines predictors by learning a supervised model of input query characteristics and the errors that individual predictors make. In this way, model "stacking" treats a set of predictors as a panel of experts, and learns the kinds of questions each is most expert at answering correctly. Stacked models can thus be strictly more accurate than their component predictors (24), making them attractive for hard problems like link prediction (26), but only if those predictors make distinct errors and are sufficiently diverse in the signals they exploit.

We evaluate individual prediction methods, and their stacked generalizations, using two types of network data. The first is 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 ϵ, being low, medium, or high. These synthetic networks thus range from homogeneous to heterogeneous random graphs, from no modules to many modules, and from weakly to strongly modular structure (see SI Appendix, section B and Table S3). Moreover, because the data generating process for these networks is known, we exactly calculate the optimal accuracy that any link prediction method could achieve, as a reference point (see SI Appendix, section B).

The second is a large corpus of 548 real-world networks. This structurally diverse corpus includes social (23%), biological (33%), economic (22%), technological (12%), information (3%), and transportation (7%) networks (8), and spans three orders of magnitude in size (see SI Appendix, section C and Fig. S1). It is by far the largest and most diverse empirical benchmark of link prediction methods to date, and enables an assessment of how methods perform across scientific domains.

Finally, our evaluations 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. This choice presents a hard test, 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 work. Our results thus provide a general, application-agnostic assessment of link predictability and method performance. In cases of 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 missingness model f, along with all observed non-edges $V \times V - E'$ as negative examples (see SI Appendix, section D). Unless other specified, results reflect a choice of $\alpha = 0.8$, i.e., 20% of edges are unobserved (holdout set); other values produce qualitatively similar results.

Results

Prediction Error Diversity. If all link predictors exploit a common underlying signal of missingness, then one or a few predictors will consistently perform best across realistic inputs. Optimal link prediction could then be obtained by further leveraging this universal signal. In contrast, if different predictors exploit distinct signals, they will exhibit a diversity of errors in the form of heterogenous performance across inputs, In this case, there will be no best or worst method overall, and optimal link predictions can only be obtained by combining multiple methods. This dichotomy also holds at the level of predictor families, one of which could be best overall, e.g., topological methods, even if no one family member is best.

To distinguish these possibilities, we characterize the empirical distribution of errors by training a random forest classifier over the 203 link predictors applied to each of the 548 real-world networks and separately to all networks in each of the six scientific domains within our corpus (see SI Appendix section E). In this setting, the character of a predictor's errors

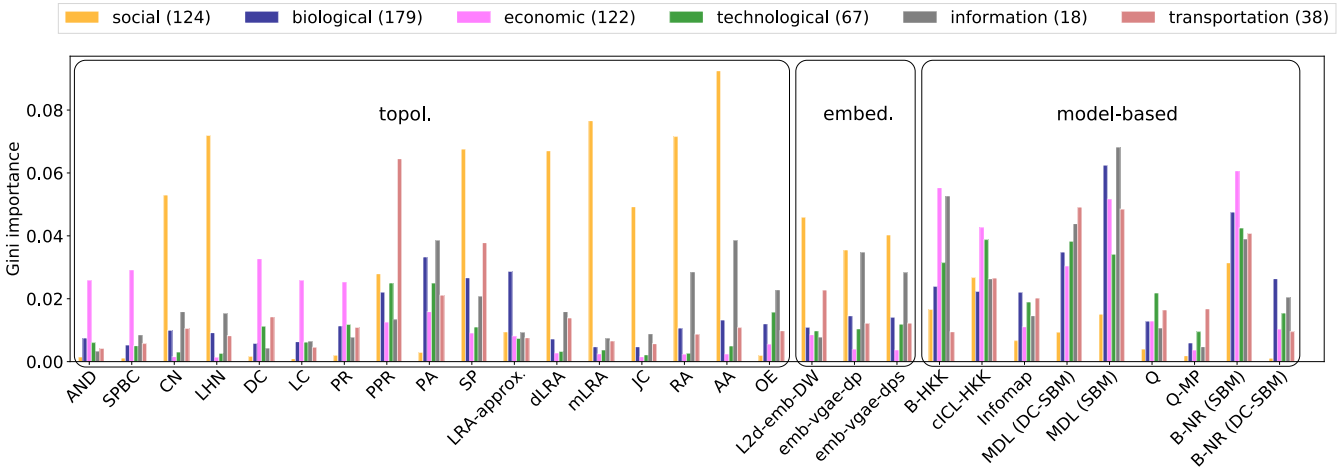

Fig. 1. The Gini importances for predicting missing links in networks within each of six scientific domains, for the 29 most important predictors, grouped by family, under a random forest classifier trained over all 203 predictors. Across domains, predictors exhibit widely different levels of importance, indicating a diversity of errors, such that no predictor is best overall. Here, topological predictors include shortest-path betweenness centrality (SPBC), common neighbors (CN), Leicht-Holme-Newman index (LHN), personalized page rank (PPR), shortest path (SP), the mean neighbor entries within a low rank approximation (mLRA), Jaccard coefficient (JC), and the Adamic-Adar index (AA); embedding predictors include the L2 distance between embedded vectors under emb-DW (L2d-emb-DW), and the dot product (emb-vgae-dp) of embedded vectors under emb-vgae; and, model-based predictors include Infomap (Infomap), stochastic block models with (MDL (DC-SBM), B-NR (DC-SBM)) and without degree corrections (MDL (SBM), B-NR (SBM)), and modularity (Q). (A complete list of abbreviations is given in SI Appendix, Section A.)

is captured by its learned Gini importance (mean decrease in impurity) (11) within the random forest: the higher the Gini importance, the more generally useful the predictor is for correctly identifying missing links on that network or that domain. If all methods exploit a common missingness signal (one method to rule them all), the same few predictors or predictor family will be assigned consistently greater importance across networks and domains. However, if there are multiple distinct signals (a diversity of errors), the learned importances will be highly heterogeneous across inputs, and no predictor or family will be best.

Across networks and domains, we find wide variation in both individual and family-wise predictor importances, such that no individual method and no family of methods is best, or worst, on all networks. On individual networks, predictor importances tend to be highly skewed, such that a relatively small subset of predictors account for the majority of prediction accuracy (SI Appendix, Table S4 and Fig. S2). However, the precise composition of this subset varies widely across both networks and families (SI Appendix, Tables S4–S5, and Figs. S3–S4), implying a broad diversity of errors and multiple distinct signals of missingness. At the same time, not all predictors perform well on realistic inputs, e.g., a subset of topological methods generally receive low importances, and most embedding-based predictors are typically mediocre. Nevertheless, each family contains some members that are ranked among the most important predictors for many, but not all, networks.

Across domains, predictor importances cluster in interesting ways, such that some individual and some families of predictors perform better on specific domains. For instance, examining the 10 most-important predictors by domain (29 unique predictors; Fig. 1), we find that topological methods, such as those based on common neighbors or localized random walks, perform well on social networks but less well on networks from other domains. In contrast, model-based methods perform relatively well across domains, but often perform less well on social networks than do topological measures and some embedding-based methods. Together, these results indicate that predictor methods exhibit a broad diversity of errors, which tend correlate somewhat with scientific domain.

This performance heterogeneity highlights the practical relevance to link prediction of the general No Free Lunch theorem (19), which proves that across all possible inputs, every machine learning method has the same average performance, and hence accuracy must be assessed on a per dataset basis. The observed diversity of errors indicates that none of the 203 individual predictors is a universally-best method for the subset of all inputs that are realistic. However, that diversity also implies that a nearly-optimal link prediction method for realistic inputs could be constructed by combining individual methods so that the best individual method is applied for each given input. Such a meta-learning algorithm cannot circumvent the No Free Lunch theorem, but it can achieve optimal performance on realistic inputs by effectively redistributing its worse-than-average performance onto unrealistic inputs, which are unlikely to be encountered in practice. In the following sections, we develop and investigate the near-optimal performance of such an algorithm.

Stacking on Networks with Known Structure. Model "stacking" is a meta-learning approach that learns to apply the best individual predictor according to the input's characteristics (24). Here, we assess the accuracy of model stacking both within and across families of prediction methods, which adds seven more prediction algorithms to our evaluation set.

Because the optimality of an algorithm's predictions can only be assessed when the underlying data generating process is known, we first characterize the accuracy of model stacking using synthetic networks with known structure, for which we calculate an exact upper bound on link prediction accuracy (see SI Appendix, section B). To provide a broad range of realistic variation in these tests, we use a structured random graph model, in which we systematically vary its degree distribution's

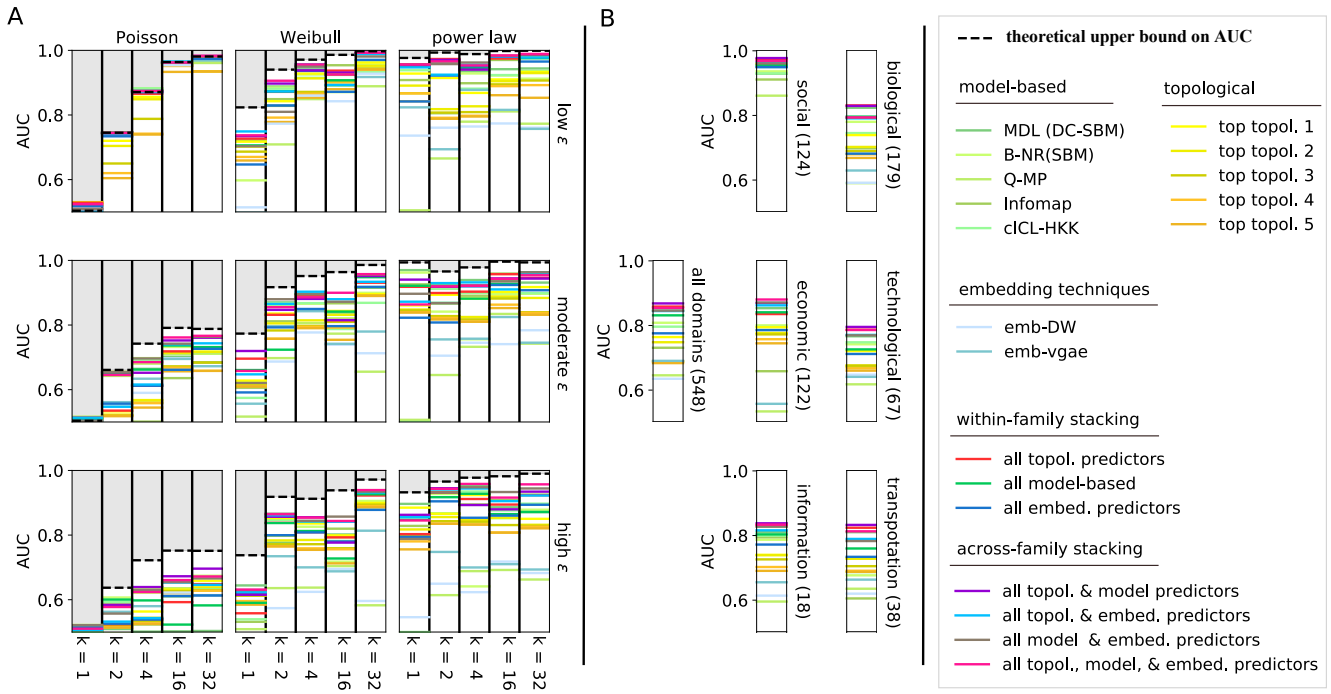

Fig. 2. (A) On synthetic networks, the mean link prediction performance (AUC) of selected individual predictors and all stacked algorithms across three forms of structural variability: (left to right, by subpanel) degree distribution variability, from low (Poisson) to high (power law); (top to bottom, by subpanel) fuzziness of community boundaries, ranging from low to high ($\epsilon = m_{\text{out}}/m_{\text{in}}$, the fraction of a node's edges that connect outside its community); 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, stacked models perform optimally or nearly optimally, and generally perform better when networks exhibit heavier-tailed degree distributions and more communities with distinct boundaries. Table S11 lists the top five topological predictors for each synthetic network setting, which vary considerably. (B) On real-world networks, the mean link prediction performance for the same predictors across all domains, and by individual domain. Both overall and within each domain, stacked models, particularly the across-family versions, exhibit superior performance, and they achieve nearly perfect accuracy on social networks. The performance, however, varies considerably across domains, with biological and technological networks exhibiting the lowest link predictability. Due to space limitations here, more complete results for individual topological and model-based predictors are shown in SI Appendix, Figs. S8 and S9, respectively.

variance, the number of communities k, and the fuzziness of those community boundaries ϵ.

Across these structural variables, the upper limit on link predictability varies considerably (Fig. 2A), from no better than chance in a simple random graph ($k = 1$; Poisson) to nearly perfect in networks with many distinct communities and a power-law degree distribution. Predictability is generally lower (no methods can do well) with fewer communities (low k) or with more fuzzy boundaries (high ϵ), but higher with increasing variance in the degree distribution (Weibull or power law). Most methods, whether stacked or not, perform relatively well when predictability is low. However, as potential predictability increases, methods exhibit considerable dispersion in their accuracy, particularly among topological and embedding-based methods.

Regardless of the synthetic network's structure, however, we find that stacking methods are typically among the most accurate prediction algorithms, and they often achieve optimal or nearly-optimal prediction accuracy (Fig. 2A). For instance, the best model stacking method exhibits a substantially smaller gap between practical and optimal performance (all topol., model & embed., $\Delta\text{AUC} = 0.04$; SI Appendix, Table S8) than the best individual predictor (MDL (DC-SBM), $\Delta\text{AUC} = 0.07$; SI Appendix, Table S9), and is far better than the average non-stacked topological and model-based methods ($\langle\Delta\text{AUC}\rangle = 0.23$; SI Appendix, Table S8). Moreover, in all structural settings, stacking across families tends to produce slightly more accurate predictions ($\langle\text{AUC}\rangle = 0.83$; SI Appendix, Table S10) than stacking within families ($\langle\text{AUC}\rangle = 0.80$), and only one stacked model (all embed.) is less accurate than the best individual predictor (marginally, with $\Delta\text{AUC} = 0.01$, and Table S10).

Stacking on Real-world Networks. To characterize the real-world accuracy of model stacking, we apply these methods, along with the individual predictors, to our corpus of 548 structurally diverse real-world networks. We analyze the results both within and across scientific domains, and as a function of network size.

Both across all networks, and within individual domains, model stacking methods produce the most accurate predictions of missing links (Fig. 2B and Table 1), and some individual predictors perform relatively well, particularly model-based ones. Applied to all networks, the best model-stacking method achieves slightly better average performance (all topol. & model, $\langle\text{AUC}\rangle = 0.87 \pm 0.10$) than the best individual method (MDL (DC-SBM), $\langle\text{AUC}\rangle = 0.84 \pm 0.10$), and far better performance than the average individual topological or model-based predictor ($\langle\text{AUC}\rangle = 0.63$; and see Tables 1 and S6). However, model stacking also achieves substantially better precision in its predictions (Table 1), which can be a desirable property in practice. We note that these stacking results were obtained by optimizing the standard F measure to choose the random forest's parameters. Alternatively, we may optimize the AUC

Table 1. Link prediction performance (mean±std. err.), measured by AUC, precision, and recall, for link prediction algorithms applied to the 548 structurally diverse networks in our corpus.

algorithm	AUC	precision	recall
Q	0.7 ± 0.14	0.14 ± 0.17	0.67 ± 0.15
Q-MR	0.67 ± 0.15	0.12 ± 0.17	0.63 ± 0.13
Q-MP	0.64 ± 0.15	0.09 ± 0.11	0.59 ± 0.17
B-NR (SBM)	0.81 ± 0.13	0.13 ± 0.12	0.65 ± 0.22
B-NR (DC-SBM)	0.7 ± 0.2	0.12 ± 0.12	0.61 ± 0.24
cICL-HKK	0.79 ± 0.13	0.14 ± 0.14	0.58 ± 0.25
B-HKK	0.77 ± 0.13	0.11 ± 0.1	0.51 ± 0.26
Infomap	0.73 ± 0.14	0.12 ± 0.12	0.68 ± 0.13
MDL (SBM)	0.79 ± 0.15	0.14 ± 0.13	0.57 ± 0.3
MDL (DC-SBM)	0.84 ± 0.1	0.13 ± 0.11	0.78 ± 0.12
S-NB	0.71 ± 0.19	0.12 ± 0.13	0.66 ± 0.17
mean model-based	0.74 ± 0.16	0.12 ± 0.13	0.63 ± 0.21
mean indiv. topol.	0.6 ± 0.13	0.09 ± 0.16	0.53 ± 0.35
mean indiv. topol. & model	0.63 ± 0.15	0.09 ± 0.16	0.55 ± 0.33
emb-DW	0.63 ± 0.23	0.17 ± 0.19	0.42 ± 0.35
emb-vgae	0.69 ± 0.19	0.05 ± 0.05	0.69 ± 0.21
all topol.	0.86 ± 0.11	0.42 ± 0.33	0.44 ± 0.32
all model-based	0.83 ± 0.12	0.39 ± 0.34	0.3 ± 0.29
all embed.	0.77 ± 0.16	0.32 ± 0.32	0.32 ± 0.31
all topol. & model	0.87 ± 0.1	0.48 ± 0.36	0.35 ± 0.35
all topol. & embed.	0.84 ± 0.13	0.4 ± 0.34	0.39 ± 0.33
all model & embed.	0.84 ± 0.13	0.36 ± 0.32	0.36 ± 0.31
all topol., model & embed.	0.85 ± 0.14	0.42 ± 0.34	0.39 ± 0.33

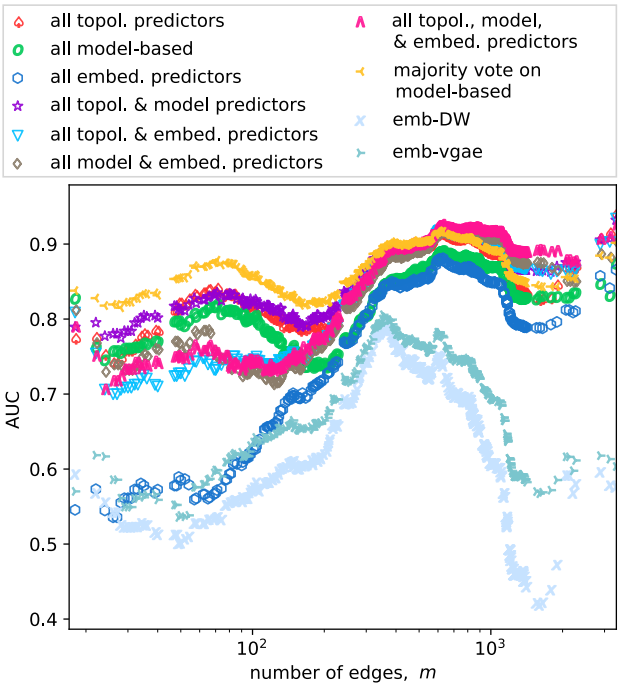

Fig. 3. Mean link prediction performance (AUC) as a function of network size (number of edges m) for stacked models and select individual predictors, applied to 548 real-world networks. Generally, stacking topological predictors, model-based predictors, or both yields superior performance, but especially on larger networks where link prediction is inherently more difficult.

itself, which produces similar results, but with slightly lower precisions in exchange for slightly higher AUC scores (see Table S18).

Among the stacked models, the highest accuracy on real-world networks is achieved by stacking model-based and topological predictor families. Adding embedding-based predictors does not significantly improve accuracy, suggesting that the network embeddings do not capture more structural information than is represented by the model-based and topological families. This behavior aligns with our results on synthetic networks above, where the performances of stacking all predictors and stacking only model-based and topological predictors were nearly identical (SI Appendix, Tables S8 and S9).

Applied to individual scientific domains, we find considerable variation in missing link predictability, which we take to be approximated by the most-accurate stacked model (Fig. 2B). In particular, most predictors, both stacked and individual (SI Appendix, Figs. S8 and S9), perform well on social networks, and on these networks, model stacking achieves nearly perfect link prediction (up to AUC $= 0.98 \pm 0.06$; Table S12). In contrast, this upper limit is substantially lower in non-social domains, being lowest for biological and technological networks (AUC $= 0.83 \pm 0.10$; Tables S13 and S15), while marginally higher for economic and information networks (AUC $= 0.88 \pm 0.10$; SI Appendix, Tables S14 and S16).

Stacked models also exhibit superior performance on link prediction across real-world networks of different scales (number of edges m; Fig. 3), and generally exhibit more accurate predictions as network size increases, where link prediction is inherently harder. We note, however, that on small networks ($m < 200$), an alternative algorithm based on a simple majority-vote among model-based predictors slightly outperforms all stacking methods, but performs substantially worse than the best stacked model on larger networks ($m > 1000$). And, embedding-based methods perform poorly at most scales, suggesting a tendency to overfit, although stacking within that family produces higher accuracies on larger networks, but still lower than other stacked models.

Sufficiency and Optimality. In practice, the optimality of a meta-learning method can only be established indirectly, over a set of considered predictors applied to a sufficiently diverse range of empirical tests cases (19). We assess this indirect evidence for stacked link-prediction models through two numerical experiments.

In the first, we consider how performance varies as a function of the number of predictors stacked, either within or across families. Evidence for optimality here appears as an early saturation, in which performance achieves its maximum prior to the inclusion of all available individual predictors. This behavior would indicate that a subset of predictors is sufficient to capture the same information as the total set. To test for this early-saturation signature, we first train a random forest classifier on all predictors in each of our stacked models and calculate each predictor's within-model Gini importance. For each stacked model, we then build a new sequence of sub-models in which we stack only the k most important predictors at a time and assess its performance on the test corpus.

In each of the stacked models, performance exhibits a classic saturation pattern: it increases quickly as the 10 most-important predictors are included, and then stabilizes by around 30 predictors (Fig. 4 and SI Appendix, Fig. S5). Performance then degrades slightly beyond 30–50 included predictors,

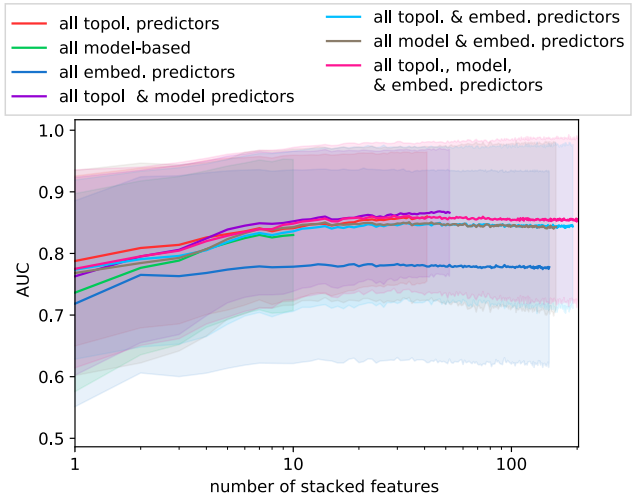

Fig. 4. Mean link prediction performance (AUC) as a function of the number of stacked features, for within- and across-family stacked models, applied to 548 real-world networks. The shaded regions show the standard error, and the early saturation behavior (at between 10 and 50 predictors) indicates that a small subset of predictors is sufficient to capture the same information as the total set.

suggesting a slight degree of overfitting in the full models. Notably, each within and across family model exhibits a similar saturation curve, except for the embedding-only model, which saturates early and at a lower level than other stacked models. This similar behavior suggests that these families of predictors are capturing similar missingness signals, despite their different underlying representations of the network structure. As in other experiments, the best saturation behavior is achieved by stacking model-based and topological predictors.

In the second, we evaluate whether individual predictors represent "weak" learners in the sense that their link-prediction performance is better than random. In general, we find that nearly all of the predictors satisfy this condition (SI Appendix, Figs. S6 and S7), implying that they can be combined according to the Adaboost theorem to construct an optimal algorithm (27). Replacing the random forest algorithm within our stacking approach with a standard boosting algorithm also produces nearly identical performance on our test corpus (see Tables S19–S22). The similar performance between the two methods suggests that relatively little additional performance is likely possible using other meta-learning approaches over the same set of predictors.

Discussion

Developing more accurate methods for predicting missing links in networks would help reduce the use of scarce resources in collecting network data, and would provide more powerful tools for evaluating and comparing network models of complex systems. The literature on such methods gives an unmistakable impression that most published algorithms produce reasonably accurate predictions. However, relatively few of these studies present systematic comparisons across different families of methods and they typically draw their test cases from a narrow set of empirical networks, e.g., social networks. As a result, it has remained unknown whether a single best predictor or family of predictors exists, how link predictability itself varies across different methods and scientific domains, or how close to optimality current methods may be.

Our broad analysis of individual link prediction algorithms, representing three large and popular families of such methods, applied to a large corpus of structurally diverse networks, shows definitively that common predictors in fact exhibit a broad diversity of errors across realistic inputs (Fig. 1 and SI Appendix, Fig. S2). Moreover, this diversity is such that no one predictor, and no family of predictors is overall best, or worst, in practice (SI Appendix, Table S4 and Fig. S3). The common practice of evaluating link prediction algorithms using a relatively narrow range of test cases is thus problematic. The far broader range of empirical networks and algorithms considered here shows that, generally speaking, good performance on a few test cases does not generalize across inputs. The diversity of errors we find serves to highlight the practical relevance of the No Free Lunch theorem (19) for predicting missing links in complex networks, and suggests that optimal performance on realistic inputs may only be achieved by combining methods, e.g., via meta-learning, to construct an ensemble whose domain of best performance matches the particular structural diversity of real-world networks.

Model stacking is a popular meta-learning approach, and our results indicate that it can produce highly accurate predictions of missing links by combining either topological predictors alone, model-based predictors alone, or both. Applied to structurally diverse synthetic networks, for which we may calculate optimal performance, stacking achieves optimal or near-optimal accuracy, and accuracy is generally closer to perfect when networks exhibit a highly variable degree distribution and/or many, structurally distinct communities (Fig. 2A).

Similarly, applied to empirical networks, stacking produces more accurate predictions than any individual predictor (Fig. 2B and Table 1), and these predictions appear to be nearly optimal, i.e., we find little evidence that further accuracy can be achieved using this set of predictors (Fig. 4), even under alternative meta-learning approaches. Of course, we cannot rule out the possibility that more accurate predictions overall could be obtained by incorporating, within the stacked models, specific new predictors or new families, if they provide better prediction coverage of some subset of input networks than do the currently considered predictors. Given the diverse set of predictors and families considered here, this possibility seems unlikely without fundamentally new ideas about how to represent the structure of networks, and therefore also signals of missingness.

Across networks drawn from different scientific domains, e.g., social vs. biological networks, we find substantial variation in link predictor performance, both for individual predictors and for stacked models. This heterogeneity suggests that the basic task of link prediction may be fundamentally harder in some domains of networks than others. Most algorithms produce highly accurate predictions in social networks, which are stereotypically rich in triangles (local clustering), exhibit broad degree distributions, and are composed of assortative communities, suggesting that link prediction in social networks may simply be easier (28) than in non-social network settings. In fact, stacked models achieve nearly perfect accuracy at distinguishing true positives (missing links) from true negatives (non-edges) in social networks (Fig. 2B and SI Appendix, Table S12). An alternative interpretation of this difference is that the existing families of predictors exhibit some degree

of selective inference, i.e., they work well on social networks because social network data is the most common inspiration and application for link prediction methods. Our results make it clear that developing more accurate individual predictors for non-social networks, e.g., biological and informational networks, is an important direction of future work. Progress along these lines will help clarify whether link prediction is fundamentally harder in non-social domains, and why.

Across our analyses, embedding-based methods, which are instances of representation learning on networks, generally perform more poorly than do either topological or model-based predictors. This behavior is similar to recent results in statistical forecasting, which found that neural network and other machine learning methods perform less well by themselves than when combined with other, conventional statistical methods (29, 30). A useful direction of future work on link prediction would specifically investigate tuning embedding-based methods to perform better on the task of link prediction.

Only strong theoretical guarantees, which currently seem out of reach, would allow us to say for certain whether the stacked models presented here actually achieve the upper bound on link prediction performance in complex networks. However, the evidence suggests that stacking achieves nearly optimal performance across a wide variety of realistic inputs. It is likely that efforts to develop new individual link prediction algorithms will continue, and these efforts will be especially beneficial in specific application domains, e.g., predicting missing links in genetic regulatory networks or in food webs. Evaluations of new predictors, however, should be carried out in the context of meta-learning, in order to assess whether they improve the overall prediction coverage embodied by the state-of-the-art stacked models applied to realistic inputs. Similarly, these evaluations should be conducted on a large and structurally diverse corpus of empirical networks, like the one considered here. More narrow evaluations are unlikely to produce reliable estimates of predictor generalization. Fortunately, stacked models can easily be extended to incorporate any new predictors, as they are developed, providing an incremental path toward fully optimal predictions.

ACKNOWLEDGMENTS. The authors thank David Wolpert, Brendan Tracey, and Cristopher Moore for helpful conversations, acknowledge the BioFrontiers Computing Core at the University of Colorado Boulder for providing High Performance Computing resources (NIH 1S10OD012300) supported by BioFrontiers IT, and thank the Information Sciences Institute at the University of Southern California for hosting AGh during this project. Financial support for this research was provided in part by Grant No. IIS-1452718 (AGh, AC) from the National Science Foundation. Data and code for replication purposes are provided at [https://github.com/Aghasemian/OptimalLinkPrediction].

1. Kossinets G (2006) Effects of missing data in social networks. *Social Networks* 28(3):247–268.
2. Fire M, et al. (2013) Computationally efficient link prediction in a variety of social networks. *ACM Transactions on Intelligent Systems and Technology (TIST)* 5(1):10.
3. Lü L, Zhou T (2011) Link prediction in complex networks: A survey. *Physica A: statistical mechanics and its applications* 390(6):1150–1170.
4. Nagarajan M, et al. (2015) Predicting future scientific discoveries based on a networked analysis of the past literature in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. (ACM), pp. 2019–2028.
5. Kane GC, Alavi M, Labianca GJ, Borgatti S (2014) What's different about social media networks? a framework and research agenda. *MIS Quarterly* 38(1):274–304.
6. Burgess M, Adar E, Cafarella M (2016) Link-prediction enhanced consensus clustering for complex networks. *PLoS ONE* 11(5):e0153384.
7. Mirshahvalad A, Lindholm J, Derlen M, Rosvall M (2012) Significant communities in large sparse networks. *PloS one* 7(3):e33721.
8. Ghasemian A, Hosseinmardi H, Clauset A (2019) Evaluating overfit and underfit in models of network community structure. *IEEE Trans. Knowledge and Data Engineering (TKDE)*.
9. 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.
10. Arlot S, Celisse A, , et al. (2010) A survey of cross-validation procedures for model selection. *Statistics Surveys* 4:40–79.
11. Hastie T, Tibshirani R, Friedman J (2009) *The elements of statistical learning: data mining, inference, and prediction*. (New York, NY: Springer).
12. Clauset A, Moore C, Newman MEJ (2008) Hierarchical structure and the prediction of missing links in networks. *Nature* 453(7191):98.
13. Martínez V, Berzal F, Cubero JC (2017) A survey of link prediction in complex networks. *ACM Computing Surveys (CSUR)* 49(4):69.
14. Al Hasan M, Zaki MJ (2011) A survey of link prediction in social networks in *Social Network Data Analytics*. (Springer), pp. 243–275.
15. 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.
16. Zhou T, Lü L, Zhang YC (2009) Predicting missing links via local information. *The European Physical Journal B* 71(4):623–630.
17. 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.
18. Cai H, Zheng VW, Chang KCC (2018) A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering* 30(9):1616–1637.
19. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation* 1(1):67–82.
20. Peel L, Larremore DB, Clauset A (2017) The ground truth about metadata and community detection in networks. *Science Advances* 3(5):e1602548.
21. Schapire RE (1990) The strength of weak learnability. *Machine Learning* 5(2):197–227.
22. Breiman L (1996) Bagging predictors. *Machine Learning* 24(2):123–140.
23. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15(1):1929–1958.
24. Wolpert DH (1992) Stacked generalization. *Neural Networks* 5(2):241–259.
25. 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.
26. Koren Y (2009) The BellKor solution to the Netflix Grand Prize. *Netflix prize documentation 81* pp. 1–10.
27. Freund Y, Schapire RE (1997) A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences* 55:119–139.
28. Epasto A, Perozzi B (2019) Is a single embedding enough? Learning node representations that capture multiple social contexts in *The World Wide Web Conference*. (ACM), pp. 394–404.
29. Makridakis S, Spiliotis E, Assimakopoulos V (2018) The M4 competition: Results, findings, conclusion and way forward. *International Journal of Forecasting* 34(4):802–808.
30. Makridakis S, Spiliotis E, Assimakopoulos V (2018) Statistical and machine learning forecasting methods: Concerns and ways forward. *PLoS ONE* 13(3):e0194889.
31. Newman M (2019) *Networks*. (Oxford University Press).
32. 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.
33. 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.
34. Leicht EA, Holme P, Newman MEJ (2006) Vertex similarity in networks. *Physical Review E* 73(2):026120.
35. Newman MEJ, Girvan M (2004) Finding and evaluating community structure in networks. *Phys. Rev. E* 69(2):026113.
36. Newman MEJ (2016) Community detection in networks: Modularity optimization and maximum likelihood are equivalent. *arXiv:1606.02319*.
37. 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.
38. Newman MEJ, Reinert G (2016) Estimating the number of communities in a network. *Phys. Rev. Lett.* 117(7):078301.
39. Hayashi K, Konishi T, Kawamoto T (2016) A tractable fully Bayesian method for the stochastic block model. *arXiv:1602.02256*.
40. Rosvall M, Bergstrom CT (2008) Maps of random walks on complex networks reveal community structure. *Proc. Natl. Acad. Sci. USA* 105(4):1118–1123.
41. Peixoto TP (2013) Parsimonious module inference in large networks. *Phys. Rev. Lett.* 110(14):148701.
42. Krzakala F, et al. (2013) Spectral Redemption in Clustering Sparse Networks. *Proc. Natl. Acad. Sci.* 110(52):20935–20940.
43. 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.
44. Kipf TN, Welling M (2016) Variational graph auto-encoders. *preprint arXiv:1611.07308*.
45. Hamilton WL, Ying R, Leskovec J (2017) Representation learning on graphs: Methods and applications. *preprint arXiv:1709.05584*.
46. Dietterich T (2000) Ensemble methods in machine learning. *Multiple Classifier Systems* pp. 1–15.
47. Sewell M (2008) Ensemble learning. *RN* 11(02).
48. 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.
49. 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.
50. Karrer B, Newman MEJ (2011) Stochastic blockmodels and community structure in networks. *Physical review E* 83(1):016107.
51. 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.
52. Clauset A, Tucker E, Sainz M (2016) The Colorado Index of Complex Networks. (https://icon.colorado.edu/).
53. 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*.
54. Ahmed C, ElKorany A, Bahgat R (2016) A supervised learning approach to link prediction in twitter. *Social Network Analysis and Mining* 6(1):24.
55. 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.
56. Cover TM, Thomas JA (2012) *Elements of information theory.* (John Wiley & Sons).

Supporting Information

1. 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 (15), 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 like Fig. 1 in the main text.

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) (14, 15, 31, 32).

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), 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) (14, 15, 31, 32).

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) (14, 15, 31, 32).

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 (8), 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. (8) 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. (8) for additional details.

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-DeepWalk) (43)—a special case of node2vec (emb-node2vec) (17)—and the variational graph auto encoder (emb-vgae) (44).

Using emb-DeepWalk 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 (45). 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 (22, 25, 46, 47). By treating the output of individual prediction algorithms as features of the input instances themselves,

*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.

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" (24), 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 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. For a given network $G = (V, E)$, we sample the edges uniformly and 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 and take all non-edges 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 F measure optimization and the results are reported on the test

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	•			(31)
OE	number of observed edges	•			(31)
AD	average degree	•			(31)
VD	variance of degree distribution	•			(31)
ND	network diameter	•			(31)
DA	degree assortativity of graph	•			(33)
NT	network transitivity (clustering coefficient)	•			(31)
ACC	average (local) clustering coefficient	•			(31)
CN	common neighbors of i, j		•		(15)
SP	shortest path between i, j		•		(15)
LHN	Leicht-Holme-Newman index of neighbor sets of i, j		•		(34)
PPR	j-th entry of the personalized page rank of node i		•		(33)
PA	preferential attachment (degree product) of i, j		•		(15)
JC	Jaccard's coefficient of neighbor sets of i, j		•		(15)
AA	Adamic/Adar index of i, j		•		(15)
RA	resource allocation index of i, j		•		(33)
LRA	entry i, j in low rank approximation (LRA) via singular value decomposition (SVD)		•		(32)
dLRA	dot product of columns i and j in LRA via SVD for each pair of nodes i, j		•		(32)
mLRA	average of entries i and j's neighbors in low rank approximation		•		(32)
LRA-approx	an approximation of LRA		•		(32)
dLRA-approx	an approximation of dLRA		•		(32)
mLRA-approx	an approximation of mLRA		•		(32)
LCC_i, LCC_j	local clustering coefficients for i and j			•	(33)
AND_i, AND_j	average neighbor degrees for i and j			•	(33)
SPBC_i, SPBC_j	shortest-path betweenness centralities for i and j			•	(33)
CC_i, CC_j	closeness centralities for i and j			•	(33)
DC_i, DC_j	degree centralities for i and j			•	(33)
EC_i, EC_j	eigenvector centralities for i and j			•	(33)
KC_i, KC_j	Katz centralities for i and j			•	(33)
LNT_i, LNT_j	local number of triangles for i and j			•	(33)
PR_i, PR_j	Page rank values for i and j			•	(33)
LC_i, LC_j	load centralities for i and j			•	(33)

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.

Abbreviation	Description	Likelihood	Optimization	Ref.
Q	modularity, Newman-Girvan		•	(35)
Q-MR	modularity, Newman's multiresolution		•	(36)
Q-MP	modularity, message passing		•	(37)
B-NR (SBM)	Bayesian stochastic block model, Newman and Reinert	•		(38)
B-NR (DC-SBM)	Bayesian degree-corrected stochastic block model, Newman and Reinert	•		(38)
B-HKK (SBM)	Bayesian stochastic block model, Hayashi, Konishi and Kawamoto	•		(39)
cICL-HKK (SBM)	Corrected integrated classification likelihood, stochastic block model	•		(39)
Infomap	Map equation		•	(40)
MDL (SBM)	Minimum description length, stochastic block model	•		(41)
MDL (DC-SBM)	Minimum description length, degree-corrected stochastic block model	•		(41)
S-NB	Spectral with non-backtracking matrix	•		(42)

set. Results for, instead, optimizing using the AUC are given in Table S18 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 (48) and AdaBoost (49), for learning a single algorithm over the individual predictors. The results from these meta-learning algorithms are provided in Tables S19-S22 for different choices of model selection through AUC or F measure.

2. 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 fraction of a node's edges that connect outside its community, 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) (50), 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 $p_{\text{out}} = \tilde{\epsilon}\, p_{\text{in}}$,
- 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

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

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 2.

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	$n = 673, \beta = 1.5, \epsilon = 0.05$

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 and 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 $\lambda_{r,s}(d_i, d_j) = \frac{d_i}{d_r}\frac{d_j}{d_s}\omega_{r,s}$, 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 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. (51)), 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).

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

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) (51), where $\epsilon \to 0$), Eq. (3) can be rewritten as Eq. (6):

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

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

- First term:

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

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

- Second term:

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

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

- 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:

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

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 (51), 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:

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

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

- Second term:

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

- Third term:

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

- Fourth term:

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

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

3. Empirical corpus for link prediction evaluations

To evaluate and compare the different link prediction algorithms in a practical setting, we have selected 548 networks[‡] from the "CommunityFitNet corpus," a novel data set[§] containing 572 real-world networks drawn from the Index of Complex Networks (ICON) (52). This corpus spans a variety of network sizes and structures, with 22% social, 21% economic, 34% biological, 12% technological, 4% information, and 7% transportation graphs (Fig. S1).

4. 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

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

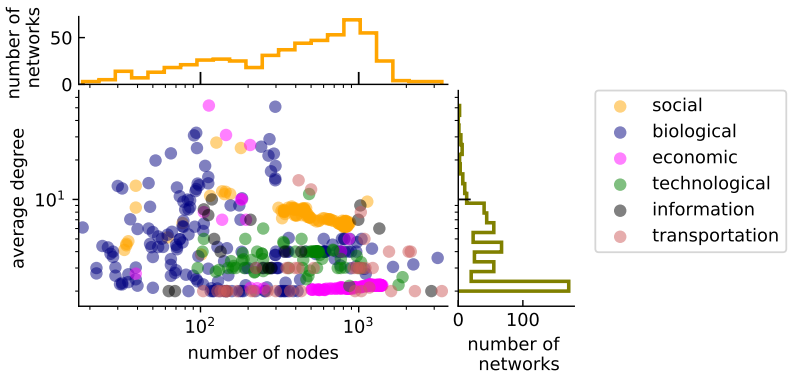

Fig. S1. Average degree versus number of nodes for a subset of CommunityFitNet corpus (8) consisting of 548 real-world networks drawn from the Index of Complex Networks (ICON) (52), including social, biological, economic, technological, information, and transportation graphs.

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 (53, 54), or if some missing links are known *a priori*, they may be used as training examples (55). 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.

5. 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 548 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 0.64 ± 0.14 (mean±stddev), 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 (56). 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.

We begin by computing the learned feature importance entropies for each domain in each family (Table S4). To calculate these, 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 domain via supervised learning, as described above.[‖] We denote the feature importances of all predictors in a family ℓ and 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 9% of predictors account for 90% of the importance in social networks. Other domains require far more predictors, e.g., 37% for biological

[¶] 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.

[‖] Unless otherwise noted, the reported results are based on a training a random forest. We also used AdaBoost and XGBoost and similar results have been observed (see Tables S19-S22).

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	5.03	9.36(19)	7.66
	biology	6.79	37.44(76)	
	economy	6.57	31.03(63)	
	technology	7	44.83(91)	
	information	6.41	27.59(56)	
	transportation	6.67	33.99(69)	
	feature-wise	6.71	34.98(71)	
Family	Domain	Entropy	Top x% (n)	Uniform
all topol. predictors (42)	social	3.85	21.43(9)	5.39
	biology	5.09	61.9(26)	
	economy	4.65	42.86(18)	
	technology	5.01	57.14(24)	
	information	4.81	47.62(20)	
	transportation	4.9	52.38(22)	
	feature-wise	5.08	61.9(26)	
Family	Domain	Entropy	Top x% (n)	Uniform
all model-based predictors (11)	social	3.14	63.64(7)	3.46
	biology	3.35	72.73(8)	
	economy	3.36	72.73(8)	
	technology	3.37	72.73(8)	
	information	2.94	54.55(6)	
	transportation	3.24	63.64(7)	
	feature-wise	3.31	72.73(8)	
Family	Domain	Entropy	Top x% (n)	Uniform
all embed. predictors (150)	social	4.65	9.33(14)	7.23
	biology	6.51	42.67(64)	
	economy	6.38	38.67(58)	
	technology	6.74	52(78)	
	information	5.19	14.67(22)	
	transportation	6.62	46.67(70)	
	feature-wise	6.23	34(51)	

and 45% 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 91 of 203 total predictors for technological networks, 26 out of 42 topol. predictors for biological networks, 8 of 11 model-based predictors for biological, economic, and technological networks, and 78 of 150 embed. predictors for technological networks. Also we see that across these predictor sets most of the uncertainty can be explained by at least 19 out of 203, 9 out of 42, 7 out of 11, and 14 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 learned 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.

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	1.27	1.58
	biology	1.57	
	economy	1.58	
	technology	1.55	
	information	1.58	
	transportation	1.56	

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 548 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 least important predictors (low importance-rank across networks) fall primarily in the topological family and a few model-based predictors. 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 large group of predictors (green cluster) on the left of the hierarchy correspond to the embedding-based predictors, whose distribution of importances is concentrated in the middle range (Fig. S4, inset panel 1). A second group (red 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 2). And a third group (cyan cluster) includes to predictors that receive high importance across nearly all 548 networks (Fig. S4, inset panels 7–9), as well as some with more bimodal importance (Fig. S4, inset panels 4–6).

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 548 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 1 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 548 real-world networks in our empirical corpus and the 45 generated synthetic data.

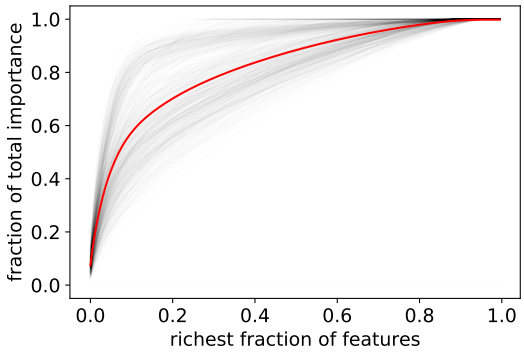

Fig. S2. Lorenz curves of the importance of the features. These curves illustrate that in a large portion of empirical networks, a very large fraction of learned "importance" belongs to a small fraction of predictors. The red solid line shows the average Lorenz curve over the 548 networks.

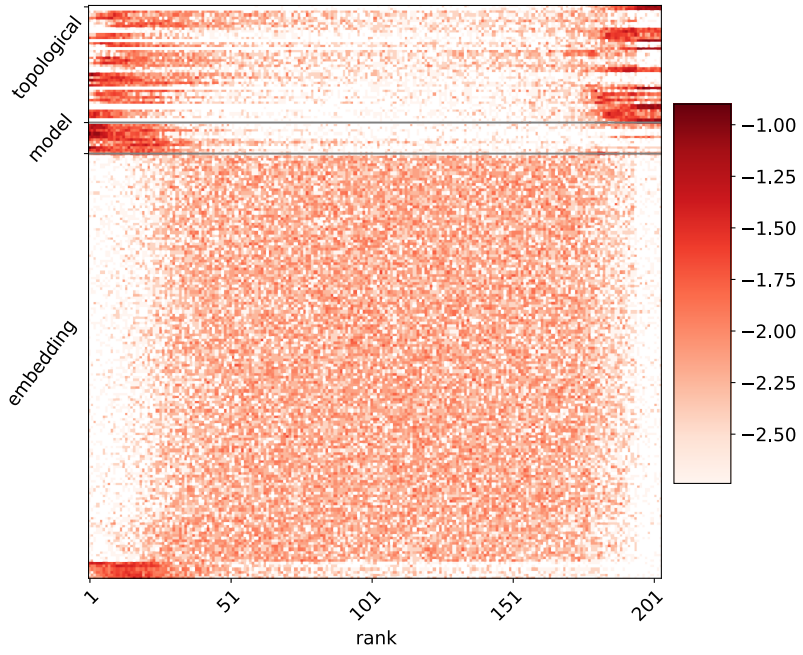

Fig. S3. Distribution of the importance ranks of each predictor across 548 networks. The most important features typically belong to model-based and topological predictor families. Among embedding predictor, the most important correspond to the distance measures among the embedded vectors. Almost all vector embedding predictors have middling levels of importance, although they are rarely the worst predictors. The distribution of the ranks is logarithmic (base 10).

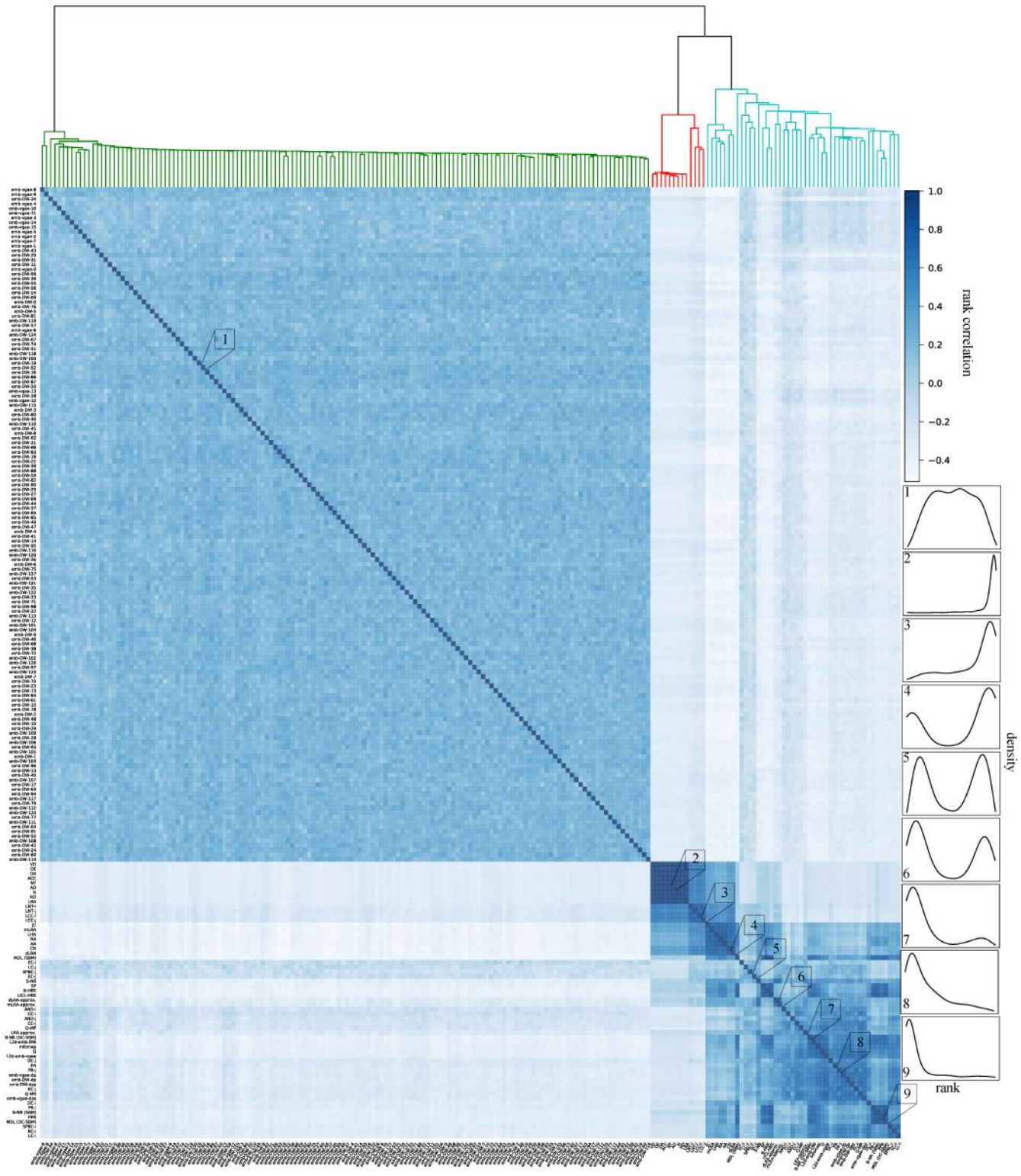

Fig. S4. A clustering of features based on the similarities of their rank in Gini importance. Clusters show similar importance distribution among 548 networks. Embedding predictors appear in middle ranks uniformly for different networks (inset 1). The red cluster shows the worst importance among different networks (insets 2–3). The cyan cluster shows better importance and the most important features are located to the right of this group (insets 4–9).

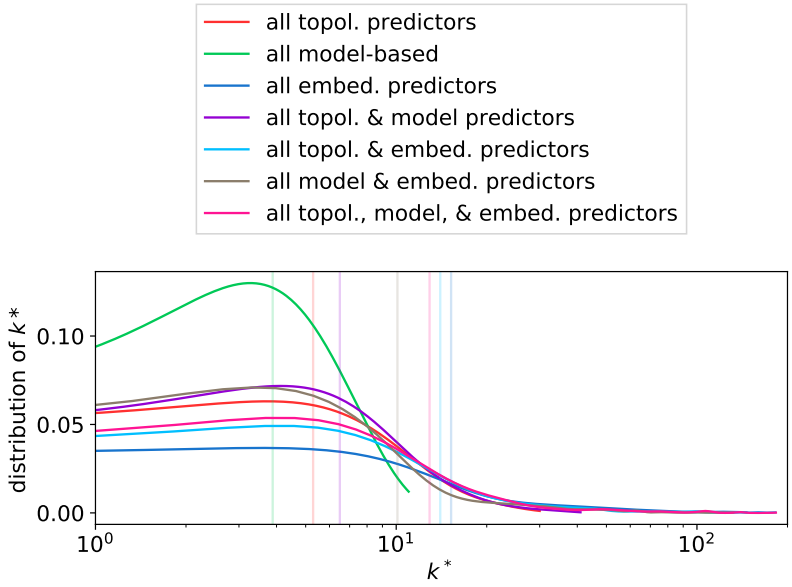

Fig. S5. Distribution of the minimum number of features k^* that is needed to achieve at least 95% of final AUC for each family of stacking methods.

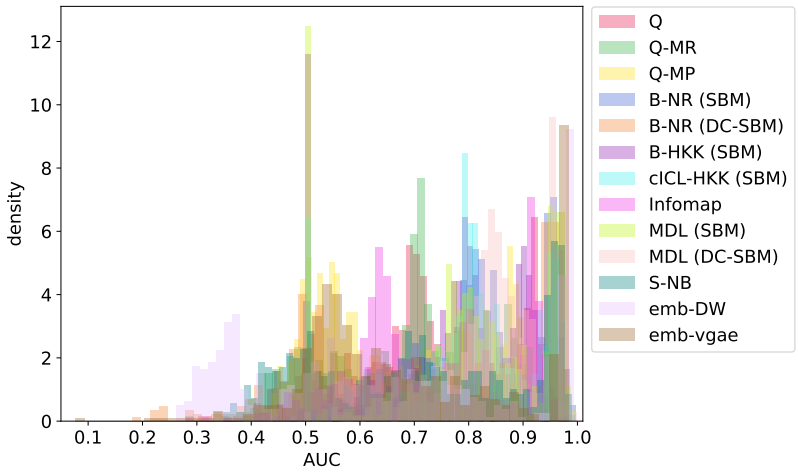

Fig. S6. Histogram of AUC performances on 548 empirical networks for all 11 model-based "weak learners" and two embedding link predictors used in model stacking.

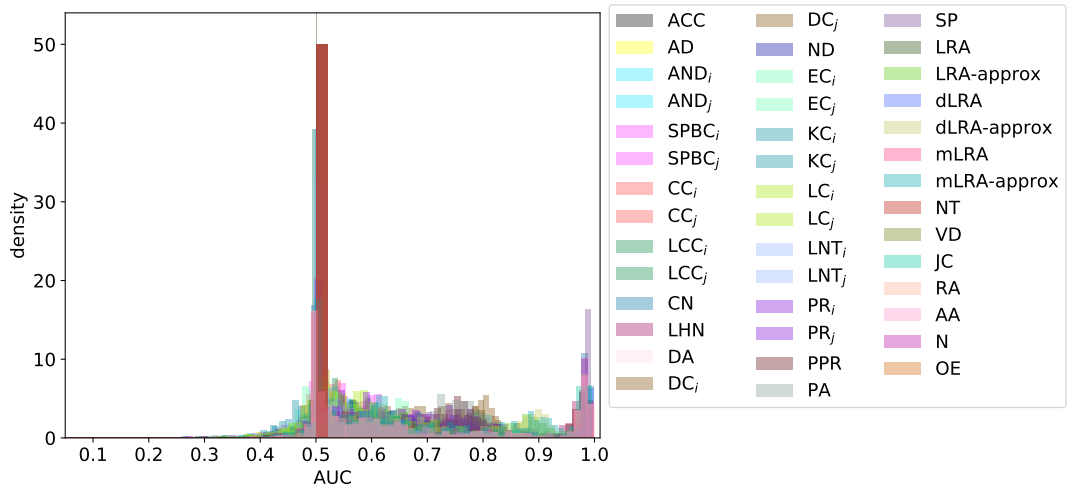

Fig. S7. Histogram of AUC performances on 548 empirical networks for all 42 individual topological "weak learners" used in model stacking.

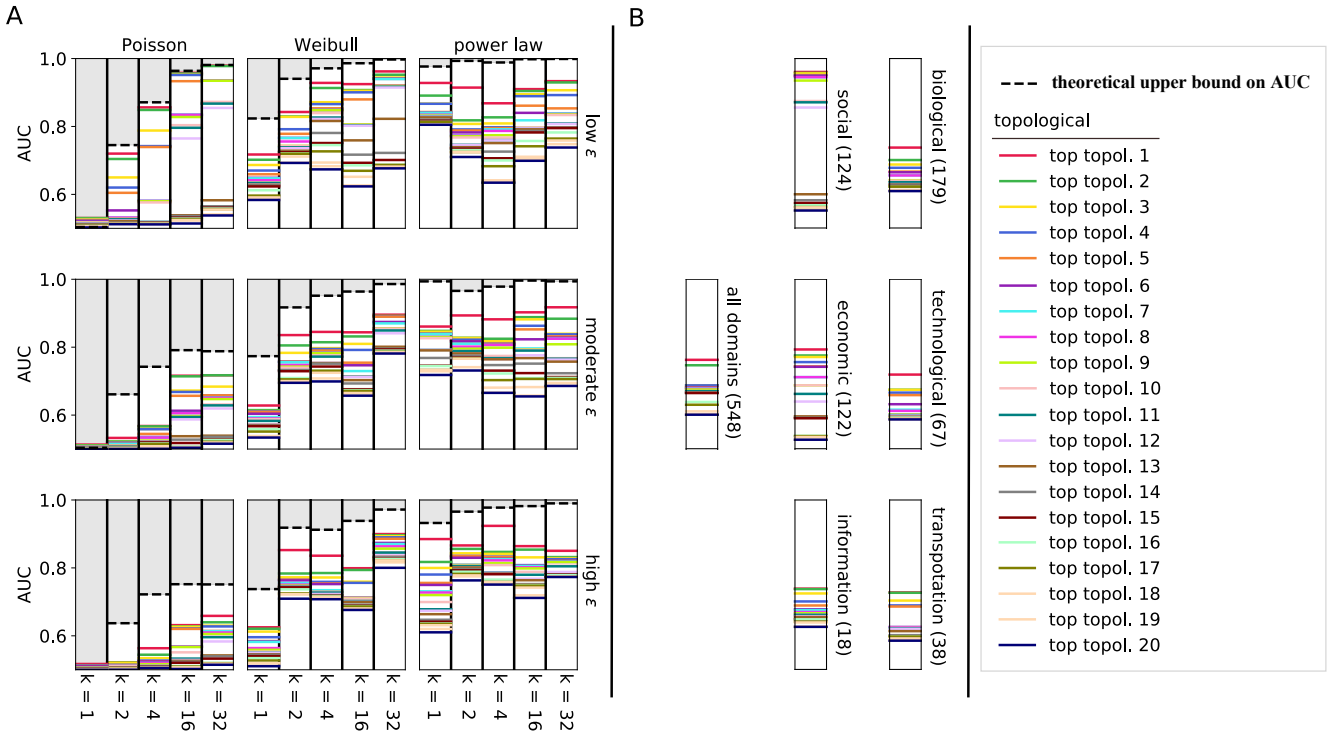

Fig. S8. (A) On synthetic networks, the mean link prediction performance (AUC) of topological individual predictors and all stacked algorithms across three forms of structural variability: (left to right, by subpanel) degree distribution variability, from low (Poisson) to high (power law); (top to bottom, by subpanel) fuzziness of community boundaries, ranging from low to high ($\epsilon = m_{\text{out}}/m_{\text{in}}$, the fraction of a node's edges that connect outside its community); 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, stacked models perform optimally or nearly optimally, and generally perform better when networks exhibit heavier-tailed degree distributions and more communities with distinct boundaries. (B) On real-world networks, the mean link prediction performance for the same predictors across all domains, and by individual domain. Both overall and within each domain, stacked models, particularly the across-family versions, 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.

Table S6. Link prediction performance (mean$\pm$std. err.), measured by AUC, precision, and recall, for individual topological predictors applied to the 548 structurally diverse networks in our corpus.

algorithm	AUC	precision	recall
ACC	0.5 ± 0.0	0.01 ± 0.02	0.48 ± 0.5
AD	0.5 ± 0.0	0.02 ± 0.02	0.51 ± 0.5
AND_i	0.6 ± 0.12	0.05 ± 0.04	0.46 ± 0.21
AND_j	0.61 ± 0.12	0.05 ± 0.05	0.48 ± 0.2
SPBC_i	0.58 ± 0.09	0.06 ± 0.06	0.44 ± 0.16
SPBC_j	0.55 ± 0.08	0.05 ± 0.06	0.41 ± 0.24
CC_i	0.56 ± 0.08	0.04 ± 0.03	0.5 ± 0.23
CC_j	0.6 ± 0.1	0.05 ± 0.03	0.54 ± 0.21
LCC_i	0.55 ± 0.07	0.05 ± 0.07	0.44 ± 0.36
LCC_j	0.53 ± 0.05	0.04 ± 0.06	0.45 ± 0.37
CN	0.68 ± 0.19	0.21 ± 0.27	0.7 ± 0.37
LHN	0.66 ± 0.18	0.25 ± 0.3	0.68 ± 0.37
DA	0.5 ± 0.0	0.01 ± 0.02	0.49 ± 0.5
DC_i	0.68 ± 0.11	0.06 ± 0.05	0.61 ± 0.2
DC_j	0.68 ± 0.1	0.06 ± 0.04	0.58 ± 0.18
ND	0.5 ± 0.0	0.02 ± 0.02	0.52 ± 0.5
EC_i	0.56 ± 0.09	0.05 ± 0.06	0.37 ± 0.17
EC_j	0.6 ± 0.08	0.05 ± 0.05	0.5 ± 0.21
KC_i	0.56 ± 0.09	0.05 ± 0.06	0.47 ± 0.19
KC_j	0.59 ± 0.1	0.05 ± 0.06	0.54 ± 0.22
LC_i	0.58 ± 0.09	0.06 ± 0.06	0.44 ± 0.16
LC_j	0.55 ± 0.07	0.05 ± 0.06	0.41 ± 0.24
LNT_i	0.55 ± 0.07	0.04 ± 0.05	0.51 ± 0.35
LNT_j	0.54 ± 0.07	0.04 ± 0.05	0.51 ± 0.36
PR_i	0.64 ± 0.1	0.06 ± 0.05	0.48 ± 0.18
PR_j	0.63 ± 0.11	0.06 ± 0.04	0.51 ± 0.18
PPR	0.75 ± 0.15	0.21 ± 0.26	0.57 ± 0.28
PA	0.69 ± 0.1	0.06 ± 0.05	0.61 ± 0.19
SP	0.76 ± 0.15	0.15 ± 0.18	0.73 ± 0.3
LRA	0.5 ± 0.0	0.01 ± 0.02	0.51 ± 0.5
LRA-approx	0.67 ± 0.15	0.17 ± 0.19	0.42 ± 0.3
dLRA	0.68 ± 0.19	0.2 ± 0.27	0.71 ± 0.36
dLRA-approx	0.69 ± 0.14	0.15 ± 0.19	0.56 ± 0.31
mLRA	0.67 ± 0.19	0.21 ± 0.28	0.68 ± 0.38
mLRA-approx	0.68 ± 0.14	0.14 ± 0.18	0.56 ± 0.3
NT	0.5 ± 0.0	0.01 ± 0.02	0.48 ± 0.5
VD	0.5 ± 0.0	0.01 ± 0.02	0.46 ± 0.5
JC	0.67 ± 0.19	0.23 ± 0.29	0.68 ± 0.38
RA	0.67 ± 0.19	0.24 ± 0.31	0.68 ± 0.38
AA	0.67 ± 0.19	0.24 ± 0.31	0.68 ± 0.38
N	0.5 ± 0.0	0.02 ± 0.02	0.52 ± 0.5
OE	0.5 ± 0.0	0.01 ± 0.02	0.45 ± 0.5

Table S7. Link prediction performance (mean±std. err.), measured by AUC, precision, and recall, for individual topological predictors applied to the 45 synthetic networks.

algorithm	AUC	precision	recall
ACC	0.5 ± 0.0	0.02 ± 0.02	0.44 ± 0.5
AD	0.5 ± 0.0	0.02 ± 0.02	0.49 ± 0.5
AND_i	0.57 ± 0.07	0.06 ± 0.03	0.41 ± 0.16
AND_j	0.56 ± 0.07	0.06 ± 0.03	0.4 ± 0.12
SPBC_i	0.61 ± 0.1	0.09 ± 0.06	0.44 ± 0.13
SPBC_j	0.61 ± 0.1	0.08 ± 0.06	0.46 ± 0.17
CC_i	0.61 ± 0.11	0.05 ± 0.03	0.59 ± 0.22
CC_j	0.6 ± 0.11	0.05 ± 0.03	0.63 ± 0.23
LCC_i	0.6 ± 0.1	0.09 ± 0.07	0.51 ± 0.2
LCC_j	0.62 ± 0.1	0.08 ± 0.05	0.48 ± 0.21
CN	0.71 ± 0.14	0.19 ± 0.15	0.54 ± 0.28
LHN	0.7 ± 0.14	0.22 ± 0.16	0.55 ± 0.29
DA	0.5 ± 0.0	0.02 ± 0.02	0.49 ± 0.5
DC_i	0.67 ± 0.13	0.08 ± 0.05	0.55 ± 0.16
DC_j	0.67 ± 0.12	0.08 ± 0.05	0.57 ± 0.16
ND	0.5 ± 0.0	0.02 ± 0.02	0.47 ± 0.5
EC_i	0.62 ± 0.11	0.08 ± 0.05	0.46 ± 0.14
EC_j	0.62 ± 0.12	0.08 ± 0.05	0.46 ± 0.16
KC_i	0.57 ± 0.08	0.07 ± 0.08	0.42 ± 0.11
KC_j	0.57 ± 0.09	0.06 ± 0.03	0.44 ± 0.16
LC_i	0.61 ± 0.1	0.09 ± 0.06	0.45 ± 0.13
LC_j	0.61 ± 0.1	0.08 ± 0.05	0.46 ± 0.14
LNT_i	0.63 ± 0.11	0.08 ± 0.07	0.57 ± 0.21
LNT_j	0.63 ± 0.11	0.07 ± 0.04	0.57 ± 0.2
PR_i	0.65 ± 0.12	0.09 ± 0.06	0.51 ± 0.14
PR_j	0.65 ± 0.12	0.09 ± 0.06	0.5 ± 0.15
PPR	0.74 ± 0.14	0.16 ± 0.14	0.54 ± 0.23
PA	0.72 ± 0.16	0.1 ± 0.07	0.62 ± 0.18
SP	0.75 ± 0.14	0.13 ± 0.13	0.72 ± 0.18
LRA	0.5 ± 0.0	0.01 ± 0.02	0.38 ± 0.48
LRA-approx	0.69 ± 0.14	0.15 ± 0.16	0.51 ± 0.21
dLRA	0.71 ± 0.14	0.18 ± 0.14	0.54 ± 0.27
dLRA-approx	0.73 ± 0.13	0.17 ± 0.12	0.51 ± 0.19
mLRA	0.68 ± 0.13	0.18 ± 0.15	0.51 ± 0.28
mLRA-approx	0.7 ± 0.12	0.12 ± 0.11	0.49 ± 0.19
NT	0.5 ± 0.0	0.02 ± 0.02	0.49 ± 0.5
VD	0.5 ± 0.0	0.02 ± 0.02	0.6 ± 0.49
JC	0.69 ± 0.14	0.21 ± 0.16	0.5 ± 0.29
RA	0.7 ± 0.14	0.21 ± 0.16	0.52 ± 0.28
AA	0.71 ± 0.14	0.21 ± 0.16	0.51 ± 0.28
N	0.5 ± 0.0	0.02 ± 0.02	0.62 ± 0.48
OE	0.5 ± 0.0	0.02 ± 0.02	0.49 ± 0.5

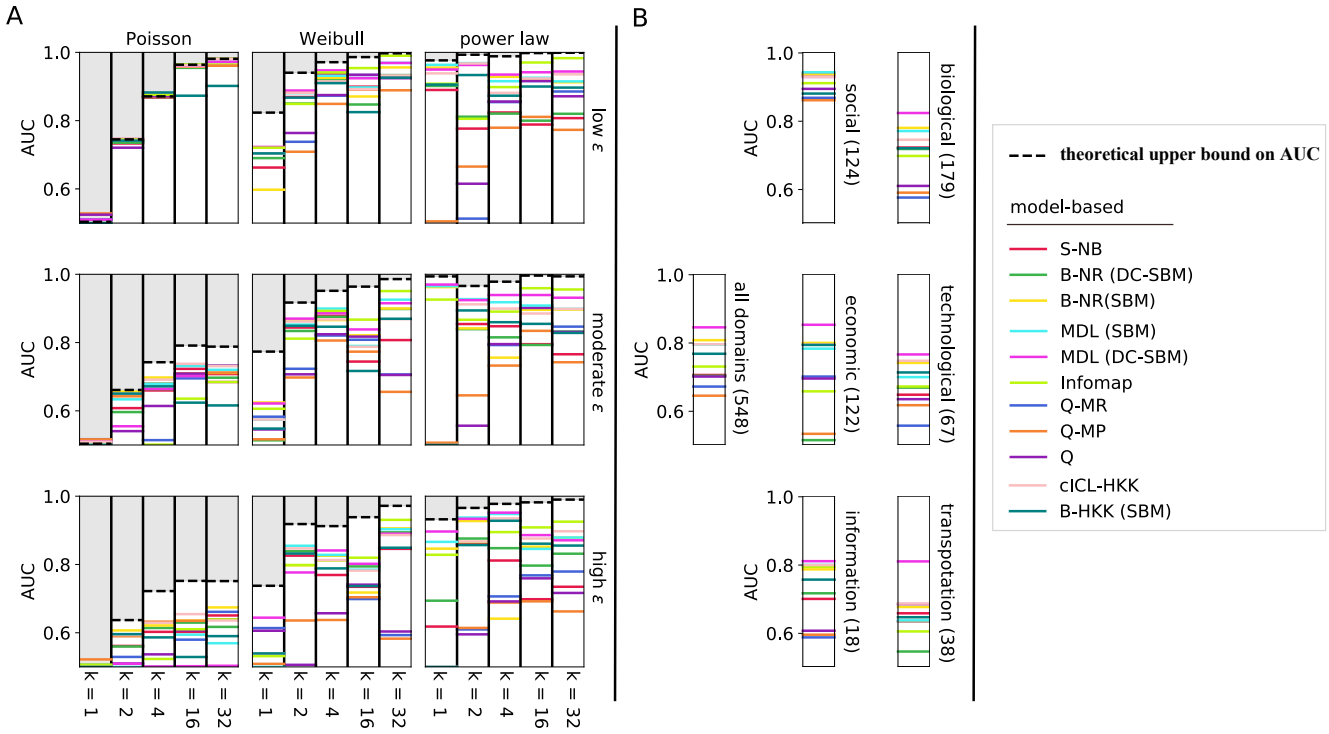

Fig. S9. (A) On synthetic networks, the mean link prediction performance (AUC) of model-based individual predictors and all stacked algorithms across three forms of structural variability: (left to right, by subpanel) degree distribution variability, from low (Poisson) to high (power law); (top to bottom, by subpanel) fuzziness of community boundaries, ranging from low to high ($\epsilon = m_{\text{out}}/m_{\text{in}}$, the fraction of a node's edges that connect outside its community); 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, stacked models perform optimally or nearly optimally, and generally perform better when networks exhibit heavier-tailed degree distributions and more communities with distinct boundaries. (B) On real-world networks, the mean link prediction performance for the same predictors across all domains, and by individual domain. Both overall and within each domain, stacked models, particularly the across-family versions, exhibit superior performance, and they achieve nearly perfect accuracy on social networks. The performance, however, varies considerably across domains, with technological networks exhibiting the lowest link predictability.

Table S8. Mean performance gap for each method in synthetic data.

Algorithm	Average gap$\langle\Delta$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.	0.257
mean indiv. topol. & model	0.229
emb-DW	0.2
emb-vgae	0.172
all topol.	0.066
all model-based	0.069
all embed.	0.09
all topol. & model	0.049
all topol. & embed.	0.057
all model & embed.	0.05
all topol., model & embed.	0.044

Table S9. The AUC gap of the best 10 predictors 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	clCL-HKK	0.09
6	B-HKK	0.12
7	SP	0.121
8	B-NR (DC-SBM)	0.123
9	S-NB	0.138
10	PPR	0.139

Table S10. Link prediction performance (mean$\pm$std. err.), 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	0.11 ± 0.14	0.66 ± 0.15
Q-MR	0.68 ± 0.17	0.11 ± 0.14	0.66 ± 0.15
Q-MP	0.69 ± 0.13	0.11 ± 0.1	0.65 ± 0.15
B-NR (SBM)	0.79 ± 0.14	0.16 ± 0.12	0.67 ± 0.24
B-NR (DC-SBM)	0.75 ± 0.15	0.17 ± 0.12	0.7 ± 0.16
clCL-HKK	0.79 ± 0.15	0.17 ± 0.14	0.61 ± 0.27
B-HKK	0.76 ± 0.15	0.13 ± 0.1	0.56 ± 0.26
Infomap	0.79 ± 0.17	0.18 ± 0.17	0.75 ± 0.16
MDL (SBM)	0.8 ± 0.16	0.17 ± 0.13	0.65 ± 0.3
MDL (DC-SBM)	0.8 ± 0.16	0.15 ± 0.11	0.76 ± 0.16
S-NB	0.74 ± 0.15	0.14 ± 0.13	0.67 ± 0.15
mean model-based	0.75 ± 0.16	0.15 ± 0.13	0.67 ± 0.21
mean indiv. topol.	0.62 ± 0.13	0.09 ± 0.11	0.51 ± 0.3
mean indiv. topol. & model	0.65 ± 0.15	0.1 ± 0.11	0.54 ± 0.29
emb-DW	0.68 ± 0.14	0.15 ± 0.14	0.36 ± 0.28
emb-vgae	0.7 ± 0.16	0.06 ± 0.03	0.72 ± 0.17
all topol.	0.81 ± 0.16	0.4 ± 0.26	0.46 ± 0.21
all model-based	0.81 ± 0.15	0.51 ± 0.33	0.38 ± 0.28
all embed.	0.79 ± 0.15	0.35 ± 0.27	0.29 ± 0.26
all topol. & model	0.83 ± 0.14	0.49 ± 0.33	0.42 ± 0.25
all topol. & embed.	0.82 ± 0.15	0.41 ± 0.28	0.42 ± 0.23
all model & embed.	0.83 ± 0.15	0.47 ± 0.31	0.36 ± 0.26
all topol., model & embed.	0.83 ± 0.15	0.48 ± 0.3	0.4 ± 0.24

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	[mLRA-approx., PPR, PA, dLRA-approx., PR-j]
low ϵ	Poisson	2	[PPR, SP, dLRA-approx., LRA-approx., mLRA-approx.]
low ϵ	Poisson	4	[PPR, SP, LRA-approx., mLRA-approx., dLRA-approx.]
low ϵ	Poisson	16	[PPR, SP, dLRA-approx., mLRA-approx., LRA-approx.]
low ϵ	Poisson	32	[PPR, SP, RA, LHN, mLRA]
low ϵ	Weibull	1	[PR-i, PA, DC-j, EC-i, KC-j]
low ϵ	Weibull	2	[SP, PA, PPR, LRA-approx., DC-i]
low ϵ	Weibull	4	[SP, dLRA-approx., LRA-approx., mLRA-approx., PA]
low ϵ	Weibull	16	[SP, dLRA-approx., mLRA-approx., PPR, LRA-approx.]
low ϵ	Weibull	32	[PPR, SP, dLRA-approx., AA, LRA-approx.]
low ϵ	power law	1	[PA, LHN, CN, dLRA, dLRA-approx.]
low ϵ	power law	2	[PA, DC-i, PR-j, LHN, AND-i]
low ϵ	power law	4	[PA, PR-i, SP, DC-i, AA]
low ϵ	power law	16	[SP, LRA-approx., dLRA-approx., PPR, PA]
low ϵ	power law	32	[dLRA-approx., SP, PPR, LRA-approx., PA]
moderate ϵ	Poisson	1	[EC-i, SPBC-i, LNT-j, EC-j, AA]
moderate ϵ	Poisson	2	[LRA-approx., SP, AND-i, PPR, KC-i]
moderate ϵ	Poisson	4	[dLRA-approx., SP, mLRA-approx., LRA-approx., PPR]
moderate ϵ	Poisson	16	[mLRA-approx., dLRA-approx., LRA-approx., SP, PPR]
moderate ϵ	Poisson	32	[PPR, SP, LRA-approx., dLRA, CN]
moderate ϵ	Weibull	1	[PA, DC-i, dLRA-approx., SPBC-j, mLRA-approx.]
moderate ϵ	Weibull	2	[PA, SP, PPR, CN, dLRA]
moderate ϵ	Weibull	4	[SP, PA, mLRA-approx., dLRA-approx., PPR]
moderate ϵ	Weibull	16	[PPR, SP, dLRA-approx., LRA-approx., PA]
moderate ϵ	Weibull	32	[dLRA-approx., AA, dLRA, CN, PPR]
moderate ϵ	power law	1	[PA, EC-i, CC-i, JC, DC-j]
moderate ϵ	power law	2	[PA, AA, RA, LHN, CN]
moderate ϵ	power law	4	[PA, LHN, SP, AA, RA]
moderate ϵ	power law	16	[PPR, dLRA-approx., SP, mLRA-approx., LRA-approx.]
moderate ϵ	power law	32	[PPR, SP, CN, LHN, JC]
high ϵ	Poisson	1	[EC-i, DC-j, KC-j, LCC-j, RA]
high ϵ	Poisson	2	[PA, EC-i, DC-j, dLRA, KC-j]
high ϵ	Poisson	4	[LRA-approx., SP, PPR, DC-j, LNT-j]
high ϵ	Poisson	16	[SP, PPR, dLRA-approx., LRA-approx., mLRA-approx.]
high ϵ	Poisson	32	[PPR, SP, mLRA-approx., dLRA-approx., AA]
high ϵ	Weibull	1	[SP, dLRA-approx., mLRA-approx., PPR, DC-i]
high ϵ	Weibull	2	[PA, dLRA-approx., SP, DC-j, PR-i]
high ϵ	Weibull	4	[PA, dLRA-approx., SP, DC-i, DC-j]
high ϵ	Weibull	16	[SP, PPR, dLRA-approx., PA, AA]
high ϵ	Weibull	32	[RA, AA, PA, CN, dLRA]
high ϵ	power law	1	[PA, DC-i, PR-i, LCC-i, LNT-i]
high ϵ	power law	2	[PA, LHN, AA, dLRA, CN]
high ϵ	power law	4	[PA, SP, LHN, AA, RA]
high ϵ	power law	16	[PPR, PA, SP, AA, CN]
high ϵ	power law	32	[PA, SP, CN, dLRA, JC]

Table S12. Average AUC, precision, and recall performances of the link prediction algorithms over 124 social 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	0.89 ± 0.07	0.42 ± 0.13	0.85 ± 0.08
Q-MR	0.87 ± 0.07	0.38 ± 0.16	0.78 ± 0.07
Q-MP	0.86 ± 0.08	0.25 ± 0.07	0.83 ± 0.09
B-NR (SBM)	0.93 ± 0.06	0.3 ± 0.08	0.85 ± 0.12
B-NR (DC-SBM)	0.93 ± 0.07	0.28 ± 0.08	0.88 ± 0.08
cICL-HKK	0.93 ± 0.08	0.34 ± 0.1	0.85 ± 0.14
B-HKK	0.88 ± 0.07	0.17 ± 0.05	0.79 ± 0.17
Infomap	0.91 ± 0.04	0.29 ± 0.08	0.83 ± 0.05
MDL (SBM)	0.94 ± 0.07	0.31 ± 0.09	0.87 ± 0.16
MDL (DC-SBM)	0.93 ± 0.09	0.26 ± 0.09	0.89 ± 0.11
S-NB	0.94 ± 0.07	0.3 ± 0.1	0.87 ± 0.08
mean model-based	0.91 ± 0.08	0.3 ± 0.12	0.84 ± 0.12
mean indiv. topol.	0.64 ± 0.19	0.2 ± 0.27	0.56 ± 0.33
mean indiv. topol. & model	0.7 ± 0.21	0.22 ± 0.25	0.62 ± 0.32
emd-DW	0.95 ± 0.1	0.45 ± 0.16	0.92 ± 0.13
emb-vgae	0.95 ± 0.08	0.09 ± 0.02	0.96 ± 0.09
all topol.	0.97 ± 0.08	0.89 ± 0.21	0.88 ± 0.2
all model-based	0.95 ± 0.07	0.76 ± 0.2	0.68 ± 0.17
all embed.	0.95 ± 0.11	0.75 ± 0.23	0.74 ± 0.23
all topol. & model	0.98 ± 0.06	0.89 ± 0.22	0.88 ± 0.19
all topol. & embed.	0.96 ± 0.1	0.86 ± 0.22	0.83 ± 0.25
all model & embed.	0.96 ± 0.09	0.78 ± 0.21	0.74 ± 0.22
all topol., model & embed.	0.97 ± 0.09	0.86 ± 0.23	0.84 ± 0.23

Table S13. Average AUC, precision, and recall performances of the link prediction algorithms over 179 biological 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 ± 0.12	0.06 ± 0.09	0.58 ± 0.13
Q-MR	0.57 ± 0.11	0.05 ± 0.09	0.56 ± 0.12
Q-MP	0.59 ± 0.09	0.06 ± 0.07	0.52 ± 0.13
B-NR (SBM)	0.78 ± 0.13	0.09 ± 0.09	0.6 ± 0.21
B-NR (DC-SBM)	0.72 ± 0.17	0.1 ± 0.09	0.63 ± 0.21
cICL-HKK	0.74 ± 0.13	0.09 ± 0.09	0.47 ± 0.24
B-HKK	0.72 ± 0.14	0.11 ± 0.12	0.39 ± 0.26
Infomap	0.7 ± 0.12	0.07 ± 0.09	0.68 ± 0.11
MDL (SBM)	0.77 ± 0.14	0.11 ± 0.1	0.51 ± 0.29
MDL (DC-SBM)	0.82 ± 0.09	0.09 ± 0.07	0.75 ± 0.11
S-NB	0.72 ± 0.14	0.09 ± 0.1	0.64 ± 0.16
mean model-based	0.7 ± 0.15	0.08 ± 0.09	0.58 ± 0.21
mean indiv. topol.	0.59 ± 0.11	0.06 ± 0.08	0.51 ± 0.35
mean indiv. topol. & model	0.62 ± 0.13	0.06 ± 0.08	0.52 ± 0.32
emd-DW	0.59 ± 0.15	0.07 ± 0.08	0.39 ± 0.25
emb-vgae	0.63 ± 0.16	0.04 ± 0.06	0.62 ± 0.2
all topol.	0.83 ± 0.1	0.27 ± 0.23	0.34 ± 0.24
all model-based	0.79 ± 0.12	0.29 ± 0.29	0.24 ± 0.25
all embed.	0.68 ± 0.16	0.17 ± 0.25	0.12 ± 0.17
all topol. & model	0.83 ± 0.1	0.35 ± 0.31	0.23 ± 0.23
all topol. & embed.	0.79 ± 0.13	0.23 ± 0.27	0.18 ± 0.2
all model & embed.	0.79 ± 0.14	0.23 ± 0.26	0.18 ± 0.2
all topol., model & embed.	0.79 ± 0.15	0.25 ± 0.27	0.18 ± 0.2

Table S14. Average AUC, precision, and recall performances of the link prediction algorithms over 122 economic 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	0.69 ± 0.06	0.04 ± 0.02	0.69 ± 0.08
Q-MR	0.7 ± 0.06	0.05 ± 0.02	0.67 ± 0.06
Q-MP	0.53 ± 0.06	0.03 ± 0.02	0.51 ± 0.11
B-NR (SBM)	0.8 ± 0.05	0.07 ± 0.05	0.6 ± 0.16
B-NR (DC-SBM)	0.51 ± 0.1	0.04 ± 0.05	0.35 ± 0.13
cICL-HKK	0.79 ± 0.06	0.06 ± 0.04	0.45 ± 0.12
B-HKK	0.79 ± 0.06	0.06 ± 0.03	0.44 ± 0.11
Infomap	0.66 ± 0.05	0.05 ± 0.04	0.62 ± 0.06
MDL (SBM)	0.78 ± 0.05	0.07 ± 0.05	0.49 ± 0.14
MDL (DC-SBM)	0.85 ± 0.06	0.09 ± 0.04	0.79 ± 0.06
S-NB	0.49 ± 0.11	0.03 ± 0.05	0.55 ± 0.07
mean model-based	0.69 ± 0.14	0.05 ± 0.04	0.56 ± 0.16
mean indiv. topol.	0.58 ± 0.12	0.04 ± 0.06	0.6 ± 0.39
mean indiv. topol. & model	0.6 ± 0.13	0.04 ± 0.05	0.59 ± 0.35
emd-DW	0.37 ± 0.11	0.09 ± 0.06	0.12 ± 0.16
emb-vgae	0.56 ± 0.07	0.03 ± 0.02	0.55 ± 0.1
all topol.	0.83 ± 0.05	0.31 ± 0.08	0.28 ± 0.14
all model-based	0.84 ± 0.07	0.27 ± 0.26	0.14 ± 0.17
all embed.	0.78 ± 0.07	0.17 ± 0.1	0.34 ± 0.18
all topol. & model	0.87 ± 0.05	0.38 ± 0.25	0.12 ± 0.15
all topol. & embed.	0.86 ± 0.07	0.3 ± 0.1	0.41 ± 0.15
all model & embed.	0.87 ± 0.09	0.21 ± 0.12	0.42 ± 0.23
all topol., model & embed.	0.88 ± 0.1	0.31 ± 0.11	0.41 ± 0.18

Table S15. Average AUC, precision, and recall performances of the link prediction algorithms over 67 technological 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	0.63 ± 0.11	0.04 ± 0.03	0.58 ± 0.12
Q-MR	0.56 ± 0.11	0.03 ± 0.02	0.54 ± 0.09
Q-MP	0.62 ± 0.08	0.04 ± 0.03	0.57 ± 0.08
B-NR (SBM)	0.74 ± 0.11	0.06 ± 0.05	0.62 ± 0.2
B-NR (DC-SBM)	0.67 ± 0.12	0.06 ± 0.06	0.63 ± 0.13
cICL-HKK	0.75 ± 0.1	0.08 ± 0.08	0.59 ± 0.18
B-HKK	0.71 ± 0.11	0.08 ± 0.08	0.5 ± 0.2
Infomap	0.67 ± 0.13	0.05 ± 0.04	0.6 ± 0.12
MDL (SBM)	0.7 ± 0.15	0.07 ± 0.07	0.45 ± 0.32
MDL (DC-SBM)	0.77 ± 0.1	0.07 ± 0.07	0.68 ± 0.12
S-NB	0.65 ± 0.09	0.04 ± 0.04	0.56 ± 0.1
mean model-based	0.68 ± 0.13	0.06 ± 0.06	0.58 ± 0.17
mean indiv. topol.	0.58 ± 0.09	0.05 ± 0.07	0.48 ± 0.34
mean indiv. topol. & model	0.6 ± 0.11	0.05 ± 0.07	0.5 ± 0.31
emd-DW	0.65 ± 0.1	0.07 ± 0.1	0.26 ± 0.17
emb-vgae	0.64 ± 0.1	0.03 ± 0.02	0.63 ± 0.12
all topol.	0.79 ± 0.1	0.24 ± 0.19	0.27 ± 0.22
all model-based	0.72 ± 0.13	0.28 ± 0.33	0.13 ± 0.15
all embed.	0.71 ± 0.11	0.2 ± 0.21	0.13 ± 0.13
all topol. & model	0.79 ± 0.09	0.32 ± 0.31	0.18 ± 0.17
all topol. & embed.	0.77 ± 0.11	0.24 ± 0.23	0.17 ± 0.15
all model & embed.	0.77 ± 0.11	0.24 ± 0.23	0.16 ± 0.16
all topol., model & embed.	0.78 ± 0.1	0.27 ± 0.23	0.17 ± 0.15

Table S16. 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	0.06 ± 0.08	0.58 ± 0.13
Q-MR	0.59 ± 0.1	0.04 ± 0.05	0.57 ± 0.15
Q-MP	0.59 ± 0.1	0.06 ± 0.07	0.54 ± 0.11
B-NR (SBM)	0.79 ± 0.14	0.13 ± 0.2	0.58 ± 0.2
B-NR (DC-SBM)	0.72 ± 0.14	0.12 ± 0.19	0.61 ± 0.17
cICL-HKK	0.8 ± 0.12	0.15 ± 0.2	0.59 ± 0.24
B-HKK	0.76 ± 0.13	0.18 ± 0.19	0.46 ± 0.24
Infomap	0.79 ± 0.08	0.09 ± 0.1	0.74 ± 0.11
MDL (SBM)	0.8 ± 0.13	0.16 ± 0.2	0.57 ± 0.25
MDL (DC-SBM)	0.81 ± 0.12	0.13 ± 0.2	0.75 ± 0.13
S-NB	0.7 ± 0.12	0.08 ± 0.08	0.6 ± 0.14
mean model-based	0.72 ± 0.15	0.11 ± 0.16	0.6 ± 0.2
mean indiv. topol.	0.61 ± 0.12	0.07 ± 0.13	0.48 ± 0.31
mean indiv. topol. & model	0.63 ± 0.13	0.08 ± 0.14	0.51 ± 0.29
emd-DW	0.61 ± 0.15	0.08 ± 0.13	0.33 ± 0.21
emb-vgae	0.65 ± 0.15	0.04 ± 0.04	0.65 ± 0.19
all topol.	0.83 ± 0.12	0.32 ± 0.25	0.39 ± 0.25
all model-based	0.8 ± 0.11	0.38 ± 0.33	0.18 ± 0.18
all embed.	0.77 ± 0.12	0.3 ± 0.28	0.17 ± 0.27
all topol. & model	0.84 ± 0.11	0.39 ± 0.3	0.23 ± 0.23
all topol. & embed.	0.81 ± 0.15	0.32 ± 0.27	0.27 ± 0.26
all model & embed.	0.83 ± 0.12	0.34 ± 0.32	0.2 ± 0.22
all topol., model & embed.	0.83 ± 0.12	0.36 ± 0.28	0.26 ± 0.27

Table S17. 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	0.07 ± 0.07	0.6 ± 0.09
Q-MR	0.63 ± 0.08	0.05 ± 0.04	0.54 ± 0.08
Q-MP	0.63 ± 0.1	0.07 ± 0.07	0.56 ± 0.11
B-NR (SBM)	0.68 ± 0.14	0.09 ± 0.11	0.44 ± 0.31
B-NR (DC-SBM)	0.55 ± 0.23	0.09 ± 0.1	0.48 ± 0.25
cICL-HKK	0.69 ± 0.13	0.1 ± 0.14	0.52 ± 0.26
B-HKK	0.65 ± 0.13	0.09 ± 0.15	0.36 ± 0.28
Infomap	0.6 ± 0.13	0.08 ± 0.1	0.53 ± 0.12
MDL (SBM)	0.64 ± 0.15	0.08 ± 0.11	0.33 ± 0.35
MDL (DC-SBM)	0.81 ± 0.07	0.09 ± 0.1	0.72 ± 0.1
S-NB	0.66 ± 0.12	0.07 ± 0.08	0.61 ± 0.1
mean model-based	0.66 ± 0.15	0.08 ± 0.1	0.52 ± 0.24
mean indiv. topol.	0.58 ± 0.1	0.09 ± 0.15	0.48 ± 0.35
mean indiv. topol. & model	0.6 ± 0.12	0.09 ± 0.14	0.49 ± 0.33
emd-DW	0.62 ± 0.15	0.2 ± 0.21	0.29 ± 0.2
emb-vgae	0.66 ± 0.11	0.04 ± 0.04	0.67 ± 0.14
all topol.	0.82 ± 0.09	0.29 ± 0.28	0.34 ± 0.25
all model-based	0.76 ± 0.11	0.29 ± 0.28	0.22 ± 0.23
all embed.	0.73 ± 0.1	0.33 ± 0.28	0.18 ± 0.16
all topol. & model	0.83 ± 0.09	0.34 ± 0.33	0.25 ± 0.24
all topol. & embed.	0.79 ± 0.12	0.33 ± 0.28	0.24 ± 0.22
all model & embed.	0.78 ± 0.11	0.35 ± 0.27	0.22 ± 0.21
all topol., model & embed.	0.81 ± 0.11	0.35 ± 0.28	0.24 ± 0.21

Table S18. Average AUC performance of the link prediction supervised stacking methods over 548 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.	0.88 ± 0.1	0.32 ± 0.31	0.65 ± 0.27
all model-based	0.87 ± 0.11	0.25 ± 0.26	0.64 ± 0.28
all embed.	0.78 ± 0.17	0.27 ± 0.33	0.25 ± 0.35
all topol. & model	0.89 ± 0.09	0.33 ± 0.32	0.64 ± 0.28
all topol. & embed.	0.85 ± 0.15	0.35 ± 0.33	0.47 ± 0.35
all model & embed.	0.85 ± 0.14	0.31 ± 0.31	0.46 ± 0.34
all topol., model & embed.	0.87 ± 0.13	0.36 ± 0.32	0.51 ± 0.34

Table S19. Average AUC, precision, and recall performances of the link prediction algorithms over 548 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.	0.85 ± 0.11	0.45 ± 0.32	0.39 ± 0.33
all model-based	0.82 ± 0.13	0.31 ± 0.27	0.37 ± 0.31
all embed.	0.77 ± 0.16	0.32 ± 0.3	0.35 ± 0.33
all topol. & model	0.85 ± 0.12	0.45 ± 0.33	0.38 ± 0.34
all topol. & embed.	0.83 ± 0.14	0.41 ± 0.34	0.38 ± 0.34
all model & embed.	0.82 ± 0.14	0.34 ± 0.3	0.39 ± 0.33
all topol., model & embed.	0.84 ± 0.13	0.41 ± 0.34	0.38 ± 0.35

Table S20. Average AUC, precision, and recall performances of the link prediction algorithms over 548 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.	0.86 ± 0.11	0.38 ± 0.32	0.5 ± 0.35
all model-based	0.84 ± 0.12	0.24 ± 0.25	0.55 ± 0.34
all embed.	0.77 ± 0.16	0.31 ± 0.31	0.32 ± 0.36
all topol. & model	0.87 ± 0.11	0.38 ± 0.33	0.49 ± 0.36
all topol. & embed.	0.84 ± 0.14	0.43 ± 0.34	0.36 ± 0.37
all model & embed.	0.83 ± 0.13	0.31 ± 0.3	0.44 ± 0.36
all topol., model & embed.	0.84 ± 0.13	0.43 ± 0.35	0.36 ± 0.37

Table S21. Average AUC, precision, and recall performances of the link prediction algorithms over 548 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.	0.82 ± 0.13	0.4 ± 0.34	0.42 ± 0.33
all model-based	0.79 ± 0.14	0.31 ± 0.31	0.4 ± 0.31
all embed.	0.74 ± 0.16	0.27 ± 0.32	0.36 ± 0.3
all topol. & model	0.81 ± 0.13	0.38 ± 0.36	0.43 ± 0.34
all topol. & embed.	0.8 ± 0.14	0.33 ± 0.35	0.45 ± 0.32
all model & embed.	0.79 ± 0.14	0.29 ± 0.33	0.46 ± 0.32
all topol., model & embed.	0.81 ± 0.14	0.33 ± 0.35	0.44 ± 0.33

Table S22. Average AUC, precision, and recall performances of the link prediction algorithms over 548 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 ± 0.12	0.3 ± 0.3	0.62 ± 0.3
all model-based	0.83 ± 0.13	0.25 ± 0.29	0.57 ± 0.32
all embed.	0.76 ± 0.16	0.25 ± 0.33	0.41 ± 0.32
all topol. & model	0.85 ± 0.12	0.32 ± 0.35	0.58 ± 0.34
all topol. & embed.	0.82 ± 0.14	0.31 ± 0.36	0.51 ± 0.35
all model & embed.	0.8 ± 0.14	0.26 ± 0.31	0.5 ± 0.33
all topol., model & embed.	0.82 ± 0.13	0.29 ± 0.35	0.51 ± 0.36