# Towards a Systematic Evaluation of Generative Network Models

Thomas Bläsius, Tobias Friedrich, Maximilian Katzmann(✉), Anton Krohmer, and Jonathan Striebel

Hasso Plattner Institute, Potsdam, Germany
maximilian.katzmann@hpi.de

**Abstract.** Generative graph models play an important role in network science. Unlike real-world networks, they are accessible for mathematical analysis and the number of available networks is not limited. The explanatory power of results on generative models, however, heavily depends on how realistic they are. We present a framework that allows for a systematic evaluation of generative network models. It is based on the question whether real-world networks can be distinguished from generated graphs with respect to certain graph parameters.

As a proof of concept, we apply our framework to four popular random graph models (Erdős-Rényi, Barabási-Albert, Chung-Lu, and hyperbolic random graphs). Our experiments for example show that all four models are bad representations for Facebook's social networks, while Chung-Lu and hyperbolic random graphs are good representations for other networks, with different strengths and weaknesses.

**Keywords:** Generative graph models · Real-world comparison
Distinguishability of network classes

## 1 Introduction

Generative graph models play an important role in network science for a multitude of reasons. They can explain how certain properties observed in real-world networks naturally emerge when assuming simplified but reasonable creation mechanisms. The *small-world phenomenon* for example emerges from a small amount of randomness in the form of independently chosen edges [10,23], and analyzing how information spreads in random networks can help to explain *information cascades*, in which individuals act based on the behavior of other individuals instead of their own information, leading to a herd-like behavior [22]. Moreover, analyzing the expected run time of an algorithm on a realistic generative model has the potential to explain why certain algorithms perform well on real-world instances despite their bad worst-case performance [16]. Finally, randomly generated instances can serve as benchmark sets for algorithms.

The explanatory power of a generative model and its usefulness as benchmark heavily depends on how well the generated graphs mimic real-world networks.


A. Bonato et al. (Eds.): WAW 2018, LNCS 10836, pp. 99–114, 2018.
https://doi.org/10.1007/978-3-319-92871-5_8

A common way to provide evidence for the usefulness of a model, is to analyze it with respect to certain fundamental properties. The properties commonly perceived as most important are the *degree distribution* (which is typically heterogeneous with many vertices of low degree and few vertices of high degree), the *diameter* (maximum distance between nodes, which is typically small), and the *clustering coefficient* (providing a measure of locality, which is typically high).

Typical examples of arguments for or against a certain model are as follows. The Barabási-Albert model leads to a power-law degree distribution, which is realistic for certain classes of real-world networks [3,7]. Chung-Lu graphs have the small-world property often observed in real-world networks as their diameter is rather small (namely $\Theta(\log n)$) [8]. The clustering coefficient of Barabási-Albert graphs tends to 0 for $n \to \infty$ [11], while it is bounded away from 0 for hyperbolic random graphs [15,17], making the latter more realistic.

Though knowing the asymptotic behavior of these fundamental properties is an important contribution to understand a model, there are disadvantages when it comes to judging how realistic it is: the statements are only of qualitative nature (a parameter is "small" or "large") but it is unclear which values are actually realistic. This is particularly true when trying to compare the asymptotic growth in a model with the specific numbers of a few real-world networks.

A more direct comparison is achieved by comparing how different a generated network and its real-world counterpart are. Such an approach heavily depends on the used similarity measure [20]. While these measures have important applications, they typically compare only pairs of networks. Thus, we believe they are not suited to evaluate the usefulness of a generative model as this pairwise comparison heavily favors overfitting and discourages the models to generalize.

The goal of being as unbiased as possible while mimicking certain important properties of real-world networks is formally captured by the term *maximum entropy model*. Erdős-Rényi graphs are for example maximum entropy with respect to the number of vertices and edges, i.e., each graph with the desired number of vertices and edges is produced with the same probability. Using this perspective, the perfect model would be one that is maximum entropy with respect to as few properties as possible such that the generated networks are indistinguishable from real-world networks with respect to as many properties as possible.

Our goal with this paper is to develop a framework that enables a systematic experimental evaluation of how good generative graph models are. In particular, it is possible to answer questions of the following type.

- Barabási-Albert, Chung-Lu, and hyperbolic random graphs all have small diameter ($\Theta(\log n / \log\log n)$ [6], $\Theta(\log n)$ [8], and polylogarithmic [14]). Which of the three is more realistic?
- Which aspects of real-world networks are well represented by a given model and which are not? Note that answering this question is particularly interesting for maximum entropy models, as it provides a direction on how to make the model more realistic.

- Which types of real-world networks (e.g., social or infrastructural) are well represented by a given model?
- Given two seemingly similar models, do they actually generate graphs with similar properties?

*Contribution and Outline.* We developed a framework capable of answering these questions. The general approach is to select generative models, a collection of real-world networks, and a set of parameters. For each model, a set of graphs fitted to the real-world networks is generated. We then answer the question whether the chosen set of parameters is sufficient to distinguish between the different collections using machine learning. This general question allows us to formulate all the above mentioned specific questions by appropriately choosing the graph collections that should be distinguished and the set of parameters.

The different components of the framework can be easily adapted: New real-world networks, further generative models, and additional parameters can be included. Moreover, the used machine learning technique is interchangeable.

To showcase our framework, we selected four models (Erdős-Rényi, Barabási-Albert, Chung-Lu, and hyperbolic random graphs) and evaluated them on 219 real-world networks based on ten different parameters. Our findings, interesting in their own right, are as follows.

- While all four models are bad representations for Facebook graphs, Chung-Lu and hyperbolic random graphs are reasonable models for other real-world networks.
- While the Chung-Lu model is better for features related to node degrees, hyperbolic random graphs excel when involving clustering or distance-related features.
- Though hyperbolic random graphs have a realistic average clustering, the variance in clustering is too low.
- In the Barabási-Albert model, the choice of the initial graph (clique or cycle) is only irrelevant if the average degree is small.

Our framework and the raw data produced in our experiments are available at https://github.com/jstriebel/nemo-eva.

*Related Work.* Attar and Aliakbary recently followed a similar approach of classifying networks based on certain graph parameters [1]. Their perspective is, however, significantly different: their goal is to decide for a given real-world network, which model is most suited to represent it. We note that this approach can also be used to evaluate which model is the most realistic for certain real-world networks by counting how many real-world networks are classified as which model. However, this only leads to a evaluation in comparison to the other models under consideration and it does not identify parameters with respect to which a model requires improvement.

## 2 Methodology

Our framework consists of three main steps. First, multiple collections of graphs are determined. Typically, we have one collection containing real-world networks and one collection for each generative model. In the second step, different graph parameters are computed for all graphs in a collection. For each graph, this yields a feature vector, which is used as its representation. Thus, the second step turns the collections of graphs into collections of feature vectors. The third step then determines, whether two collections can be distinguished based on the feature vectors, and if yes, which subsets of features can or cannot be used to distinguish between them. In the following, we describe the three steps in more detail.

### 2.1 Collections of Networks

In the first step, a collection of real-world networks and a selection of generative models is chosen. We denote the collection of real-world networks by $C = \{G_1, \dots, G_c\}$ with $c = |C|$. For each model $\textsc{m}$ and each graph $G_i \in C$, we use $\textsc{m}$ to generate an artificial graph $G_i^{\textsc{m}}$ trying to mimic the real-world network $G_i$. We denote the set of resulting networks by $C^{\textsc{m}} = \{G_1^{\textsc{m}}, \dots, G_c^{\textsc{m}}\}$.

*Fitting the Models.* We want the graph $G_i^{\textsc{m}}$ generated by $\textsc{m}$ to mimic the corresponding real-world network $G_i$. As this highly depends on the chosen model, it is not part of the framework. Most models, however, generate graphs based on a small set of input parameters and produce graphs that roughly match these parameters. In this case, we compute the relevant parameters for $G_i$ and generate $G_i^{\textsc{m}}$ using the resulting values. For input parameters of a model that are known to mainly influence one parameter of the generated graphs, without knowing an exact formula for this dependency, one can use a binary search to fit this parameter. In Sect. 3.2 we describe the fitting we used for Erdős-Rényi Graphs, Barabási-Albert, Chung-Lu, and hyperbolic random graphs.

### 2.2 Network Parameters

In the second step, each graph $G$ is turned into a feature vector by computing the values of different parameters. Formally, a *feature* $\varphi$ is a function that maps $G$ to a numerical value $\varphi(G)$. For a feature set $F = \{\varphi_1, \dots, \varphi_f\}$ of $f = |F|$ features, the *feature vector* of $G$ is $(\varphi_1(G), \dots, \varphi_f(G))$ and we denote it by $F(G)$. For a collection $C$ of graphs, $F(C)$ denotes the corresponding collection of feature vectors. We note that selecting a sufficiently expressive set of parameters is crucial: our framework is based on the assumption that structural properties distinguishing different networks types can be represented by the chosen features.

*Feature Cleaning.* To eliminate meaningless features, we apply three data cleaning techniques: numerical cleaning, variation cleaning, and correlation grouping. The *numerical cleaning* eliminates all features that are undefined or infinite for at least one of the networks. The *variation cleaning* eliminates features that have

little predictive value as they assume similar values on most networks. More precisely, features are eliminated based on their *normalized coefficient of variation*, which is defined as follows. For a given feature, let $X$ be the vector containing the $c$ values it assumes in different graphs. Then the feature's normalized coefficient of variation is defined as

$$\frac{\sigma(X)}{\mu(X)\sqrt{c-1}},$$

where $\sigma$ and $\mu$ denote the standard deviation and the arithmetic mean, respectively. We remove features with a normalized coefficient of variation below a threshold of 1%.

The *correlation grouping* groups highly correlated features, as having multiple very similar features does not add any predictive value. For each group of correlated features only the feature with the clearest semantics (given by a manually predefined order) of the group is used. The grouping is done by constructing a graph, using the features as nodes and connecting two features by an edge if they have an absolute Spearman's rank correlation coefficient above 99%. Each connected component in that graph is one group. Note that grouped features can have a smaller correlation than the threshold of 99%, as the correlation is not transitive, but being in the same connected component is.

### 2.3 Distinguishing the Collections

In the third step, we want to determine which pairs of collections can be distinguished based on which features. To this end, we want to answer queries of the following type. The input is a subset $F$ of all features and two collections of graphs, typically the collection $C$ of real-world networks and the collection $C^{\text{M}}$ for one model M (it is also possible to compare the collections of two different models, but for the sake of readability, we assume $C$ and $C^{\text{M}}$ in the following). We then want to know how well $F(C)$ can be distinguished from $F(C^{\text{M}})$, i.e., whether it can be learned which feature vectors are members of which collections by observing the membership only for few samples.

*Classification Task.* The input for the classifier consists of a feature-matrix $X \in \mathbb{R}^{2c \times f}$ ($c = |C|$ and $f = |F|$) and a binary vector $Y \in \{0,1\}^{2c}$ that classifies the features as belonging to $C$ (denoted as 0) or as to $C^{\text{M}}$ (denoted as 1). They are defined as

$$X = \begin{pmatrix} F(G_1) \\ \vdots \\ F(G_c) \\ F(G_1^{\text{M}}) \\ \vdots \\ F(G_c^{\text{M}}) \end{pmatrix} \quad \text{and} \quad Y = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \begin{matrix} \left.\vphantom{\begin{matrix}0\\\vdots\\0\end{matrix}}\right\} c \\ \left.\vphantom{\begin{matrix}1\\\vdots\\1\end{matrix}}\right\} c \end{matrix}.$$

The task of distinguishing features $X$ according to the vector $Y$ corresponds directly to the classical machine-learning setting of binary classifications: Given

the feature-space $\mathcal{X} = \mathbb{R}^{2c \times f}$ for $2c$ observations of $c$ graphs and $c$ corresponding models with $f$ real-valued features, the target value space is defined as $\mathcal{Y} = \{0, 1\}^{2c}$, and the binary classification model $M_{\mathcal{X}}$ is a function of the form $M_{\mathcal{X}} \colon \mathcal{X} \to \mathcal{Y}$. The output for each prediction is independent from other predictions, therefore

$$M_{\mathbb{R}^{2c \times f}}((x_1 \cdots x_{2c})^T) = (M_{\mathbb{R}^f}(x_1) \cdots M_{\mathbb{R}^f}(x_{2c}))^T.$$

*Evaluation.* To evaluate the resulting predictions, the accuracy is measured. Given the actual target values $Y$ and the predicted values $\hat{Y} = M(X)$, the accuracy is the ratio of correctly classified examples [2]. Therefore,

$$\text{accuracy}(\hat{Y}, Y) = \frac{\sum_i [\hat{Y}_i = Y_i]}{|Y|},$$

where $[\cdot]$ is the Iverson bracket with $[p] = 1$ if $p$ is true and $[p] = 0$ otherwise.

*Supervision and Cross-Validation.* To do supervised learning, we need training data $X_{\text{train}}$ and $Y_{\text{train}}$. With this, a supervised learning strategy $S$ results in a trained model $M_{\mathcal{X}}$, therefore $S \colon \mathcal{X}_{\text{train}} \times \mathcal{Y}_{\text{train}} \to (M_{\mathcal{X}} \colon \mathcal{X} \to \mathcal{Y})$.

To make use of all the data as a target to predict, but simultaneously prevent to use the same data as a training and a testing example, we use cross-validation. Given some predictors $X$ and target values $Y$, the *$\ell$-fold cross-validation* splits the data in $\ell$ random, equally-sized subsets $X_1, \ldots, X_\ell, Y_1, \ldots, Y_\ell$. They are used to generate $\ell$ learned models, where for each model a single subset is used as the test dataset and all other subsets as the training data. To make the training unbiased we use *stratified cross-validation* which ensures that the number of examples is the same for both classes (i.e., each $X_i$ includes the same number of feature vectors from $F(C)$ as from $F(C^{\text{M}})$). The total accuracy of the cross-validation is then defined as the arithmetic mean of the accuracies of all models.

*Classification Model.* From the wide range of possible supervised machine learning classifiers we use support vector machines (SVMs) with the Gaussian radial basis function (rbf) kernel because they have a good predictive performance in general [13], are able to capture high order dependencies [4,19], and the parametrized regularization allows to tune the variance-bias trade-off [2,19].

To select the best parameters for the SVM and the rbf kernel, cross-validation over a grid of parameters is performed. The model with the best average accuracy in the testing sets is used as the final model. All features used in the SVM are normalized to have an arithmetic mean of zero and unit variance in the training data. The testing dataset is scaled using the same parameters. This ensures that also the scaling is done in an unbiased, cross-validated fashion.

## 3 Experiments

For the experiments we used 219 publicly available graphs from Network Repository [18]. For disconnected graphs, we used the largest connected component.

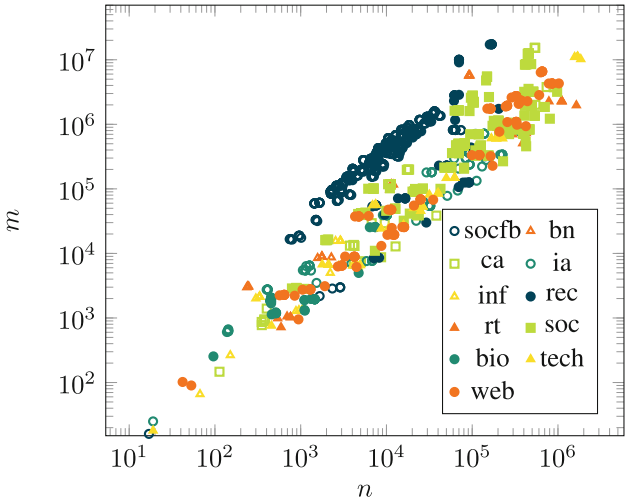


**Fig. 1.** The graphs used in our experiments.

The Network Repository divides graphs into different categories, as shown in Fig. 1. The graphs vary in their sizes and have up to 1.8 million nodes and 17 million edges. In the following, we define the used graph parameters, describe the considered generative models, and how we fitted them to the real-world networks.

### 3.1 Graph Properties

Table 1 lists all features we use. The properties we consider in our experiments can be divided into two categories: *single-value features* and *distributions over nodes*. We used NetworKit [21] to compute the features.

*Single-Value Features.* These are features that assign a single numerical value to a given graph. The most basic properties of this type are the *number of nodes* and the *number of edges*. Additionally, the *diameter* describes the maximum length of a shortest path between any two nodes in the graph and the *effective diameter* (a similar but more robust measure) represents an upper bound on the shortest path between 90% of all node pairs.

The generative models we consider are mostly meant to represent so-called scale-free networks whose degree distribution follows a power law, i.e., the fraction of vertices with at least $k$ neighbors roughly behaves like $k^{-\beta}$, where $\beta$ is the so-called *power-law exponent*.

*Distribution over Nodes.* These are features that assign a value to each node, leading to distributions over all nodes in the graph. For each of these distributions we consider the *arithmetic mean*, *median*, *first quartile* and *third quartile*, as well as the *standard deviation*.

The simplest measure of this type is the degree distribution, assigning each vertex its *degree*. The *local clustering coefficient* of a vertex $v$ is the probability that two randomly selected neighbors of $v$ are connected. The arithmetic mean of the local clustering coefficients is often referred to as "average local clustering coefficient" or simply "clustering coefficient" of the network and represents an important single-value feature. The $k$-core of a graph is obtained by successively removing all nodes with degree less than $k$. This leads to the measure of *core centrality*, where each node is assigned the largest $k$ such that it is contained in the $k$-core. The *betweenness centrality* measures for each vertex $v$ how many shortest paths between pairs of other nodes go though $v$, and the *closeness centrality* of a node denotes its average distance to every other node in the graph. Furthermore, the *Katz centrality* measures the importance of a node by its number of neighbors and the distance of all other nodes to these neighbors. Finally, the *PageRank centrality* is basically the limiting probability distribution of a random walk.

**Table 1.** The parameters we use, their abbreviations, and whether they are single-value or distribution parameters.

| Feature | Abbreviation | SV/Distr. |
|---|---|---|
| Number of nodes | $n$ | Single value |
| Number of edges | $m$ | Single value |
| Diameter | $d$ | Single value |
| Effective diameter | $d'$ | Single value |
| Power-law exponent | $\beta$ | Single value |
| Degree | deg | Distribution |
| Local clustering coefficient | $c$ | Distribution |
| Core centrality | core | Distribution |
| Betweenness centrality | betw | Distribution |
| Closeness centrality | close | Distribution |
| Katz centrality | Katz | Distribution |
| PageRank centrality | PR | Distribution |

### 3.2 Graph Models

As mentioned above, we are mostly interested in scale-free networks, i.e., highly heterogeneous networks with many low-degree and few high-degree nodes, whose degree distribution roughly follows a power law. We thus chose Barabási-Albert, Chung-Lu, and hyperbolic random graphs as models for our experiments. Moreover, we also consider the Erdős-Rényi Graphs model, as it is arguably the most basic random graph model possible. In the following we briefly describe how graphs are generated by the different models, discuss their basic properties, and report how we did the model fitting mentioned in Sect. 2.1.

*Erdős-Rényi Graphs* [12]. The Erdős-Rényi random graph model is the earliest and most studied one. A graph is generated by connecting each pair of $n$ vertices with probability $p$. Thus, one can control the number of vertices and expected number of edges, which is $p \cdot n(n-1)/2$. To fit the model to a given real-world network, we set $n$ to the number of vertices and the edge probability parameter to $p = 2m/(n(n-1))$ where $m$ denotes the number of edges in the network.

We do not expect the Erdős-Rényi Graphs model to generate very realistic graphs, i.e., we expect them to be easily distinguishable from the real-world networks.

*Barabási-Albert Graphs* [3]. This model (which is also called *preferential attachment*) generates a random graph by starting with a small graph of size $n_0$ (e.g. a cycle). Then, nodes are added one by one, each connected to $k$ already existing nodes with probability proportional to their degree, until there are $n$ nodes in the graph. The size of the initial graph is typically chosen as $n_0 = k$, which is the smallest value ensuring that the first node that is added in the generation process has enough neighbors to connect to. Note that $2k$ is the expected average degree of the resulting graph. Thus, to fit the model, we set $n$ to the number of vertices and derive $k$ from the average degree of the real-world network.

As this model generates scale-free graphs, we expect it to produce more realistic results than the Erdős-Rényi Graphs model. The main point commonly made against the Barabási-Albert model is its vanishing clustering coefficient [11], which indicates a lack of locality typically present in real-world networks.

*Chung-Lu Graphs* [8,9]. In the Chung-Lu model each node is assigned a weight and each pair of nodes is connected with a probability proportional to the product of their weights. In the resulting graph, each node has an expected degree equal to its weight. In our experiments, we fit the model by using the observed degree distribution in a real-world network as weights.

By construction, the Chung-Lu model mimics the degree distribution of a real-world network very well. As for the Barabási-Albert model, the most common point of criticism is its low clustering coefficient. Moreover, the Chung-Lu model seems more artificial than the Barabási-Albert model, as the latter mimics the evolution of a real-world network (in a simplified manner). The Chung-Lu model on the other hand matches the desired degree distribution much more accurately. It is thus interesting to know which of the two models leads to more realistic results with respect to features other than the degree distribution.

*Hyperbolic Random Graphs* [17]. In this model $n$ nodes are placed randomly in a disk within the hyperbolic plane. Then, each pair of vertices is connected if their hyperbolic distance is below a threshold, whose size depends on $n$ and the desired average degree. The resulting networks have a power-law degree distribution with the power-law exponent $\beta$ being an input parameter. Additionally, a parameter $T$ can be used to soften the threshold behaviour, allowing long-range edges with a certain probability. The geometry implies locality, which leads to a non-vanishing clustering coefficient [15]. Roughly speaking, the parameter $T$

controls how important this locality is (the more probable long-range edges are, the less important is the locality) and thus impacts the clustering coefficient.

When fitting the model parameters to a given real-world network, the largest connected component of the generated graph is typically smaller than the number of initially generated nodes $n$. To estimate $n$ we use a technique based on estimating the missing nodes of degree 0 [5]. As desired average degree, we simply use the average degree of the real-world network, and the power-law exponent $\beta$ is estimated based on the cumulative degree distribution. To fit the final parameter $T \in [0, 1)$, we perform a binary search on $T$, in each step comparing the clustering coefficients of the resulting graph and the real-world network.

**Table 2.** Failure rates on Facebook graphs. The table includes the same feature sets as Table 3, not showing all-0% rows. (∅: only average values for distributions)

| Feature Sets | ER | BA | CL | HRG |
|---|---|---|---|---|
| $n$, $m$ | 49% | 50% | 47% | 44% |
| $n$, $m$, $d$ | 0% | 0% | 1% | 0% |
| $n$, $m$, $d'$ | 1% | 1% | 14% | 16% |
| $n$, $m$, $c$ ∅ | 0% | 0% | 0% | 40% |
| $n$, $m$, betw ∅ | 4% | 5% | 12% | 41% |
| $n$, $m$, betw | 1% | 0% | 7% | 2% |
| $n$, $m$, close ∅ | 13% | 11% | 12% | 30% |
| $n$, $m$, close | 1% | 3% | 7% | 14% |
| $n$, $m$, PR | 0% | 1% | 21% | 1% |
| $n$, $m$, Katz ∅ | 0% | 1% | 20% | 2% |
| $n$, $m$, Katz | 0% | 0% | 13% | 0% |
| $n$, $m$, deg | 0% | 0% | 42% | 0% |
| $n$, $m$, core ∅ | 0% | 8% | 11% | 2% |
| $n$, $m$, core | 0% | 0% | 16% | 0% |
| $n$, $m$, deg, betw | 0% | 0% | 7% | 0% |
| $n$, $m$, close, deg | 0% | 0% | 20% | 0% |
| $n$, $m$, PR, deg | 0% | 0% | 4% | 0% |
| $n$, $m$, Katz, deg | 0% | 0% | 2% | 0% |
| $n$, $m$, core, deg | 0% | 0% | 2% | 0% |
| $c$, $\beta$, $d$ ∅ | 0% | 0% | 0% | 1% |
| $c$, $\beta$, $d'$ ∅ | 0% | 0% | 1% | 19% |
| $c$, $\beta$, $d'$ | 0% | 0% | 1% | 0% |
| betw, close, $d$ | 0% | 0% | 2% | 1% |
| betw, close, $d'$ | 0% | 1% | 4% | 1% |

We expect hyperbolic random graphs to be more realistic than the other models, due to their non-vanishing clustering coefficient. It is, however, unclear whether hyperbolic random graphs are also more realistic with respect to other features that are not explicitly enforced by the model.

### 3.3 Results

The results of our experiments can be summarized as follows. We note that the insights obtained by our method are meant to guide the direction of future research rather than being reliable scientific facts by themselves.

- None of the tested models is a good representation for Facebook's social networks. Further analysis has to show if this is due to Facebook's special structure or whether it is a general issue with graphs of high average degree.
- For other real-world networks, Chung-Lu and hyperbolic random graphs outperform Erdős-Rényi and Barabási-Albert graphs, which are easy to distinguish from real-world networks even for small parameter sets. Non-surprisingly, Chung-Lu graphs perform well with respect to the degree distribution but typically have a too low clustering. Hyperbolic random graphs not only improve with respect to clustering but also outperform Chung-Lu graphs for features related to graph distances.
- Though hyperbolic random graphs have a realistic average clustering coefficient (even for the Facebook graphs), the distribution of clustering coefficients is surprisingly unrealistic.
- In the Barabási-Albert model, the choice of the initial graph is only irrelevant when the average degree is small. For networks with average degree above 30, it is easy to distinguish between graphs initialized with cliques or cycles.

Our findings are mainly based on Tables 2 and 3. They show the failure rates of the classifier trying to separate different real-world networks from generated graphs, given a subset of features. Note that a failure rate of 0% means the model can be easily distinguished while 50% means that the classifier cannot do better than guessing. To explain the resulting data, we selectively show scatter plots to visualize the relation between two parameters. The findings in this section nicely illustrate the strength of our approach: though the absolute numbers in Tables 2 and 3 have no significant meaning, their comparison can lead to interesting insights, which can be used as starting points for further investigations.

*Facebook Graphs.* Table 2 shows the failure rates when considering only Facebook graphs (label "socfb" in Fig. 1). Table 3 shows the same data, when excluding them. One can see that Facebook graphs are much easier to distinguish from the different models than other real-world networks: ignoring the high values for the parameters directly fitted ($n$ and $m$ for all models, the degree distribution for Chung-Lu and the average clustering coefficient for hyperbolic random graphs), only few parameters are well represented. Though some parameters lead to non-zero values (PageRank, average Katz centrality, and core centrality for Chung-Lu and average betweenness and closeness for hyperbolic random graphs), the failure rates are much lower than for their non-Facebook counterparts. The only exception is the average betweenness for hyperbolic random graphs.

It is interesting to note that, e.g., for the Katz centrality in Chung-Lu graphs, the failure rate drops from 20% to 2%, when additionally taking the degree distribution into account. Non-Facebook graphs behave different in this regard.

**Table 3.** Failure rates when excluding the Facebook graphs. (∅: only average values for distributions)

| Feature Sets | ER | BA | CL | HRG |
|---|---|---|---|---|
| all (uncorrelated) | 3% | 4% | 15% | 14% |
| $n$, $m$ | 50% | 50% | 45% | 46% |
| $n$, $m$, $d$ | 28% | 22% | 28% | 35% |
| $n$, $m$, $d'$ | 37% | 38% | 33% | 43% |
| $n$, $m$, $c$ ∅ | 9% | 9% | 23% | 39% |
| $n$, $m$, $c$ | 3% | 6% | 17% | 29% |
| $n$, $m$, betw ∅ | 47% | 49% | 38% | 42% |
| $n$, $m$, betw | 2% | 11% | 37% | 41% |
| $n$, $m$, close ∅ | 44% | 47% | 44% | 46% |
| $n$, $m$, close | 25% | 26% | 43% | 45% |
| $n$, $m$, PR | 11% | 16% | 43% | 35% |
| $n$, $m$, Katz ∅ | 27% | 27% | 46% | 39% |
| $n$, $m$, Katz | 9% | 15% | 42% | 31% |
| $n$, $m$, deg | 5% | 6% | 36% | 20% |
| $n$, $m$, core ∅ | 30% | 46% | 43% | 39% |
| $n$, $m$, core | 3% | 6% | 35% | 19% |
| $n$, $m$, $c$, $d$ | 4% | 6% | 17% | 28% |
| $n$, $m$, $c$, $d'$ | 4% | 5% | 14% | 28% |
| $n$, $m$, $c$, betw | 2% | 2% | 17% | 28% |
| $n$, $m$, close, $c$ | 4% | 5% | 17% | 28% |
| $n$, $m$, PR, $c$ | 2% | 5% | 18% | 27% |
| $n$, $m$, Katz, $c$ | 1% | 4% | 16% | 22% |
| $n$, $m$, core, $c$ | 1% | 2% | 17% | 18% |
| $n$, $m$, deg, $c$ | 0% | 2% | 17% | 19% |
| $n$, $m$, deg, betw | 4% | 6% | 32% | 21% |
| $n$, $m$, deg, close | 6% | 6% | 37% | 22% |
| $n$, $m$, deg, PR | 5% | 6% | 32% | 22% |
| $n$, $m$, deg, Katz | 5% | 7% | 37% | 21% |
| $n$, $m$, deg, core | 5% | 6% | 30% | 17% |
| $c$, $\beta$, $d$ ∅ | 3% | 5% | 22% | 43% |
| $c$, $\beta$, $d$ | 4% | 5% | 20% | 33% |
| $c$, $\beta$, $d'$ ∅ | 4% | 6% | 22% | 36% |
| $c$, $\beta$, $d'$ | 4% | 6% | 20% | 30% |
| betw, close, $d$ | 5% | 6% | 31% | 38% |
| betw, close, $d'$ | 4% | 6% | 28% | 33% |

We note that most Facebook graphs have an average degree above 40, while it is below 30 for most other networks in our data set (see Fig. 3 (left)). Thus, the observed discrepancy can have its origin in the special structure of Facebook networks or in the high average degree.

*Non-Facebook Graphs.* The first row of Table 3 shows failure rates when using all features, partitioned into correlated groups (see Sect. 2.2). The smallest

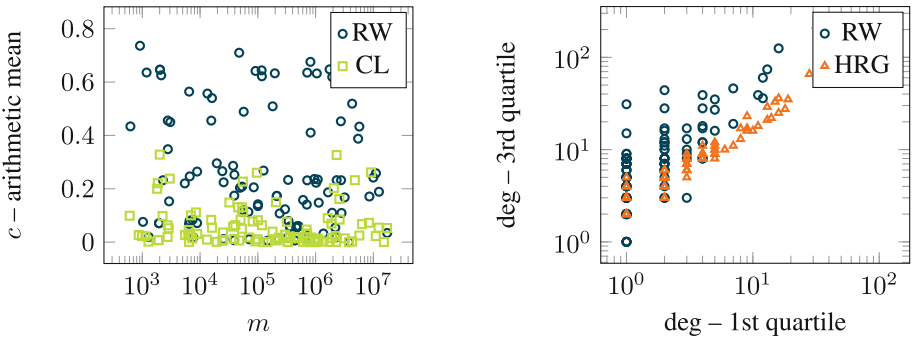


**Fig. 2.** Left: Average local clustering coefficients of real-world networks (not Facebook) and their Chung-Lu counterparts. Right: Degree distribution (1st/3rd quartile) of real-world networks (not Facebook) and their hyperbolic counterparts.

correlation within a group is 0.95. Erdős-Rényi and Barabási-Albert graphs are easy to distinguish from real-world networks. For Chung-Lu and hyperbolic random graphs, the classifier is wrong in about 15% of the cases.

Though the Erdős-Rényi and Barabási-Albert models perform reasonable with respect to the diameter, average betweenness, closeness, and average core centrality, they appear to be rather unrealistic in general. It is also interesting to note that the failure rate for Barabási-Albert graphs drops from 46% to 6% when considering the distribution of the core centrality instead of the average.

The main issue for Chung-Lu graphs is the clustering coefficient. For hyperbolic random graphs failure rates get worse when considering the degree distribution. Figure 2 (left) shows that for most considered networks the clustering coefficient of Chung-Lu graphs is too small. Figure 2 (right) shows that hyperbolic random graphs are not sufficiently heterogeneous: the degree of low-degree vertices is too high, while the degree of high-degree vertices is too low.

Concerning the other parameters, Chung-Lu graphs perform better than hyperbolic random graphs with respect to centrality measures that are closely related to the degree distribution (PageRank, Katz centrality, and core centrality). On the other hand, hyperbolic random graphs perform better with respect to measures related to distances (diameter, betweenness centrality, and closeness centrality). This is interesting as it supports the common claim that the metric of real-world networks is similar to the hyperbolic metric.

*Distribution of the Local Clustering Coefficient.* In this section, we focus on the local clustering coefficient of hyperbolic random graphs. Table 2 shows that the failure rate drops from 40% to 0% when considering the distribution instead of only the average. In Fig. 3 (left) it is easy to see that the standard deviation of the clustering coefficient is too small in hyperbolic random graphs, compared to Facebook networks. One can, however, also see, that hyperbolic random graphs can, in principal, achieve high variance. A possible explanation for the too small

standard deviation compared to Facebook graphs is that a high average degree decreases the variance in clustering for hyperbolic random graphs.

Even though the difference becomes more apparent for graphs with high-average degree, Fig. 3 (right) shows that hyperbolic random graphs tend to have too little variance in the clustering coefficient. We believe that this finding provides an interesting starting point for improving the model.

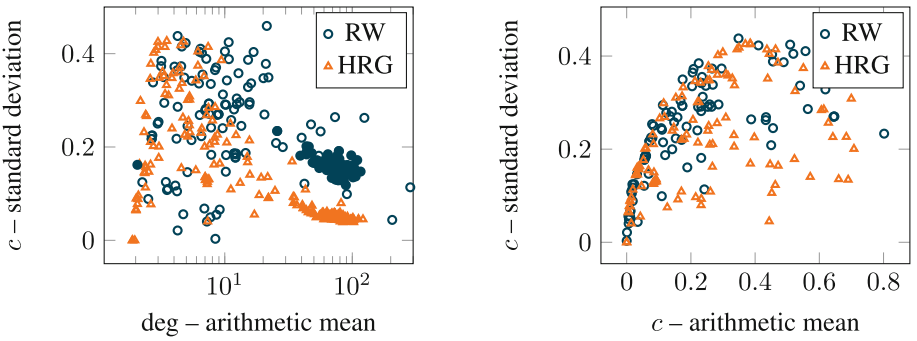


**Fig. 3.** Left: Standard deviation of clustering coefficients depending on the average degree. Facebook graphs are marked with filled shapes. Right: Standard deviation of clustering coefficients, depending on the average clustering coefficient. Facebook graphs are excluded.

*The Initial Graph in the Barabási-Albert Model.* Recall that the Barabási-Albert model generates graphs by starting with an initial graph and then successively adding vertices, each connected to the same number of already existing vertices. The size of the initial graph is typically chosen as small as possible, such that the first added vertex has enough neighbors to connect to. The original paper by Barabási and Albert [3] introducing the model does not specify how the initial graph has to be chosen and it is generally assumed to be a negligible choice.

We compared two variants of the Barabási-Albert model using cliques and cycles as initial graphs. For graphs with average degree at most 30, our classifier was not able to distinguish the two different variants (50% failure rate when using all features). If, however, the average degree is above 30, the two variants indeed lead to graphs with different properties. Using (in addition to $n$ and $m$) the degree distribution led to an 18% failure rate, which dropped to only 5% when using the distribution of clustering coefficients instead. Thus, for graphs with average degree above 30, the initial graph makes a difference. This general behaviour is not surprising as the initial graph becomes larger when increasing the average degree. It is, however, surprising that this happens for such comparatively small average degrees.

## 4 Conclusions

We have seen that the question whether or not a machine learning technique can successfully learn to distinguish between real-world and generated networks can

lead to interesting insights. We believe that this is particularly useful for guiding network science towards more realistic generative graph models by evaluating how good a model mimics the real world and by revealing its strengths and weaknesses.

**Acknowledgements.** This research has received funding from the German Research Foundation (DFG) under grant agreement no. FR 2988 (ADLON, HYP).

## References

1. Attar, N., Aliakbary, S.: Classification of complex networks based on similarity of topological network features. Chaos **27**(9), 1–7 (2017)
2. Baldi, P., Brunak, S., Chauvin, Y., Andersen, C.A.F., Nielsen, H.: Assessing the accuracy of prediction algorithms for classification: an overview. Bioinformatics **16**(5), 412–424 (2000)
3. Barabási, A.L., Albert, R.: Emergence of scaling in random networks. Science **286**(5439), 509–512 (1999)
4. Bennett, K.P., Campbell, C.: Support vector machines: hype or hallelujah? SIGKDD Explor. **2**(2), 1–13 (2000)
5. Bläsius, T., Friedrich, T., Krohmer, A., Laue, S.: Efficient embedding of scale-free graphs in the hyperbolic plane. In: 24th ESA, pp. 16:1–16:18 (2016)
6. Bollobás, B., Riordan, O.: The diameter of a scale-free random graph. Combinatorica **24**(1), 5–34 (2004)
7. Bollobás, B., Riordan, O., Spencer, J., Tusnády, G.: The degree sequence of a scale-free random graph process. Random Struct. Algor. **18**(3), 279–290 (2001)
8. Chung, F., Lu, L.: The average distances in random graphs with given expected degrees. Proc. Natl. Acad. Sci. **99**(25), 15879–15882 (2002)
9. Chung, F., Lu, L.: Connected components in random graphs with given expected degree sequences. Ann. Comb. **6**(2), 125–145 (2002)
10. Easley, D., Kleinberg, J.: The small-world phenomenon. In: Networks, Crowds, and Markets: Reasoning About a Highly Connected World, Chap. 20, pp. 611–644. Cambridge University Press (2010)
11. Eggemann, N., Noble, S.D.: The clustering coefficient of a scale-free random graph. Discrete Appl. Math. **159**(10), 953–965 (2011)
12. Erdős, P., Rényi, A.: On random graphs I. Publ. Math. **6**, 290–297 (1959)
13. Fernández-Delgado, M., Cernadas, E., Barro, S., Amorim, D., Amorim Fernández-Delgado, D.: Do we need hundreds of classifiers to solve real world classification problems? J. Mach. Learn. Res. **15**, 3133–3181 (2014)
14. Friedrich, T., Krohmer, A.: On the diameter of hyperbolic random graphs. In: Halldórsson, M.M., Iwama, K., Kobayashi, N., Speckmann, B. (eds.) ICALP 2015. LNCS, vol. 9135, pp. 614–625. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-47666-6_49
15. Gugelmann, L., Panagiotou, K., Peter, U.: Random hyperbolic graphs: degree sequence and clustering. In: Czumaj, A., Mehlhorn, K., Pitts, A., Wattenhofer, R. (eds.) ICALP 2012. LNCS, vol. 7392, pp. 573–585. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31585-5_51
16. Karp, R.M.: The probabilistic analysis of combinatorial optimization algorithms. In: Proceedings of the International Congress of Mathematicians, pp. 1601–1609 (1983)

17. Krioukov, D., Papadopoulos, F., Kitsak, M., Vahdat, A., Boguñá, M.: Hyperbolic geometry of complex networks. Phys. Rev. E **82**(3), 036106 (2010)
18. Rossi, R.A., Ahmed, N.K.: The network data repository with interactive graph analytics and visualization. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (2015). http://networkrepository.com
19. Schölkopf, B., Burges, C.J.C., Smola, A.J. (eds.): Advances in Kernel Methods: Support Vector Learning. MIT Press, Cambridge (1999)
20. Soundarajan, S., Eliassi-Rad, T., Gallagher, B.: A guide to selecting a network similarity method. In: SDM, pp. 1037–1045 (2014)
21. Staudt, C.L., Sazonovs, A., Meyerhenke, H.: NetworKit: a tool suite for large-scale complex network analysis. Netw. Sci. **4**(4), 508–530 (2016)
22. Watts, D.J.: A simple model of global cascades on random networks. Proc. Natl. Acad. Sci. **99**(9), 5766–5771 (2002)
23. Watts, D.J., Strogatz, S.H.: Collective dynamics of "small-world" networks. Nature **393**, 440–442 (1998)