Journal of Complex Networks (2014) **2**, 19–37
doi:10.1093/comnet/cnt012
Advance Access publication on 7 October 2013

Community detection: effective evaluation on large social networks

Conrad Lee[†] and Pádraig Cunningham
Clique Research Cluster, University College Dublin, 8 Belfield Office Park, Clonskeagh, Dublin 4, Ireland
[†]Corresponding author. Email: conradlee@gmail.com

Edited by: Aaron Clauset

[Received on 4 February 2013; accepted on 6 August 2013]

While many recently proposed methods aim to detect network communities in large datasets, such as those generated by social media and telecommunications services, most evaluation (i.e. benchmarking) of this research is based on small, hand-curated datasets. We argue that these two types of networks differ so significantly that, by evaluating algorithms solely on the smaller networks, we know little about how well they perform on the larger datasets. Recent work addresses this problem by introducing social network datasets annotated with meta-data that is believed to approximately indicate a 'ground truth' set of network communities. While such efforts are a step in the right direction, we find this meta-data problematic for two reasons. First, in practice, the groups contained in such meta-data may only be a subset of a network's communities. Second, while it is often reasonable to assume that meta-data is related to network communities in some way, we must be cautious about assuming that these groups correspond closely to network communities. Here, we consider these difficulties and propose an evaluation scheme based on a classification task that is tailored to deal with them.

Keywords: social networks; community detection; evaluation; benchmarking.

1. Introduction

Community structure is thought to play a key role in the formation and function of many systems, and so it comes as no surprise that hundreds of papers are currently published on the topic every year. However, in the literature it is commonly observed that although many community detection methods exist, we do not know which ones work best on real network datasets containing more than a few dozen nodes [1–5].

Evaluation (also known as benchmarking) is the task of measuring how well an algorithm detects the desired set of communities in a network. Evaluation is often carried out on either synthetic networks which, by construction, have communities planted in them, or on small empirical networks like Zachary's Karate Club, which have clearly visible community structure. Evaluation based on larger empirical network data has proved challenging because it is often difficult to define *a priori* which communities are desired. This challenge has led to a lack of evaluation on large empirical networks that has been recognized as a 'a serious limit of the field' [1] and means that it may take several years for a serious flaw of a community detection algorithm to be identified [6].

The case of modularity maximization [7] illustrates the problems caused by our limited ability to evaluate detection methods on larger empirical networks. As the most popular class of community detection technique, modularity maximization has received the greatest amount of scrutiny by the scientific

community, which has found that modularity function Q suffers from a resolution limit [6,8,9] and exhibits extreme degeneracies [2]. In their detailed evaluation of modularity-based methods on synthetic and metabolic network data, Good *et al*. [2] find that these problems are so acute that, aside from small networks that contain only few modules, 'modularity maximization can only provide a rough sketch of some parts of a network's modular organization', and that 'the output of any modularity maximization procedure should be interpreted cautiously in scientific contexts'.

Years went by before these problems were recognized, and in the meantime modularity-based methods became widely used in practice. These limitations went unrecognized in part because, as we explain in Section 2, the small networks used for detailed evaluation differ from larger empirical networks in: (1) the uniformity of coverage in the dataset, (2) the extent to which social contexts overlap and (3) orders of magnitude of size. Thus, the performance of a method on smaller datasets such as Zachary's Karate Club provides little indication of the performance on larger datasets generated by social media sites. While synthetic network generators can provide benchmarks on larger networks, it is impossible to know how closely synthetic networks resemble empirically observed networks.

We therefore have a strong motivation to develop standard benchmarks based on large, empirical social networks, but devising such benchmarks is not straightforward. In Section 2, we point out that while synthetic networks and small empirical networks like Zachary's Karate Club have an exhaustive set of desired communities associated with them, in larger social media datasets we merely have access to looser proxies of such a set. This means that evaluation on large social media data is not simply a scaled-up version of evaluation on small datasets, but requires a new evaluation framework.

In this paper, we introduce such a framework, one which depends on the availability of *meta-data* that is known to be a proxy for community structure. The type of meta-data we consider here consists of node attributes, e.g. the age of each node (our approach could also be extended to other types of meta-data, such as group assignments or edge attributes, but we do not consider such cases here). The evaluation framework we propose, which is described in Section 4, is based on a machine learning task. The basic assumption we make is that if a community detection algorithm is functioning well, then a classifier should be able to use the set of detected communities to infer missing values of a node attribute that is closely related to community structure. In other words, this evaluation scheme measures to what extent the detected communities, on the whole, can be used to infer meta-data.

A community detection algorithm can fail in many different ways; for example, it may only detect a subset of the desired communities (i.e. suffer from low *recall*), or it may detect all of the desired communities but also several spurious communities (suffer from low *precision*). An ideal evaluation scheme would be able to detect each of these deficiencies and even suggest the cause, such as that a method tends to merge or split communities. However, in Section 3 we point out that because meta-data is only a loose, incomplete proxy for community structure, any evaluation based on such meta-data will only be able to measure recall and not precision. Thus, if an algorithm detects many spurious communities in addition to finding all of the desired communities, this deficiency will not be detected by the evaluation framework we propose here.

We have created a reference implementation of this evaluation framework on the Facebook100 dataset [10,11], which contains 100 friendship networks that have meta-data which is well suited for the evaluation. In Section 5, we use this reference implementation to perform some example benchmarks, demonstrating how the evaluation framework can be used to reveal that two popular community detection algorithms (the Louvain method of modularity maximization and InfoMap) both fail to detect the smaller-scale community structure. We hope that this implementation, whose source code we have made public, will be a useful tool for other researchers who wish to evaluate new and existing algorithms.

2. Motivation: mined data differs from purpose-gathered data

From the 1940s to the 1990s, the datasets used to evaluate community detection algorithms were generally *purpose-gathered* networks for research by experts who had first-hand knowledge of the social system with which the dataset was associated. Examples include Moreno's early datasets dating back to the 1920s [12,13], the Southern Women dataset [14], Sampson's Monks [15] and Zachary's Karate Club [16]. Through their close observation, the researchers who collected this data were able to group the nodes into communities based on events such as crises or social gatherings. During this time, network datasets tended to be small (with fewer than 500 nodes, and often fewer than 50 nodes) and well-studied (in [17], Freeman synthesizes the findings of 21 methodological studies on the Southern Women's network alone).

A new era of work on community detection began in the late 1990s, caused in part by a new type of *mined* social network data that was extracted, for example, from mobile communication records or Facebook interactions [1]. Figure 1 displays an example of both a purpose-gathered and a mined network. While mined data still represents social networks, we posit that it differs from purpose-gathered data in important ways. Our motivation is not to claim that one type of data is superior to the other; rather, we wish to show that, for the purpose of evaluating the effectiveness of community detection methods, it makes sense to distinguish between the two. In particular, we contend that even if an algorithm works well on purpose-gathered datasets like Zachary's Karate club, we may nevertheless have little idea of how well it performs on mined datasets.

Here, we summarize three important differences between the two types of data related to (1) uniformity of coverage, (2) overlapping social contexts and (3) size:

1. With regard to *uniformity of coverage*, we observe that a purpose-gathered dataset's careful curation tends to ensure that sufficient information is gathered on each participant, or at least on most participants. In mined datasets, on the other hand, it is not uncommon for the majority of users to have very low activity levels and thus, arguably, be left out of the dataset entirely. Many mined datasets contain a large proportion of these low-degree nodes, leading Leskovec *et al*. [18] to characterize them in terms of whiskers and cores.

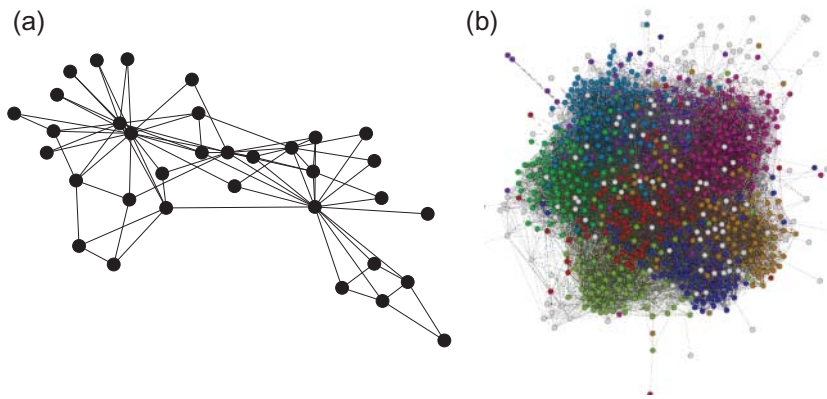

FIG. 1. On the left is a network which is typical of *hand-curated* datasets gathered by a researcher in the field. On the right is an example of a *mined* dataset. We show that the structure of the communities in these two types of networks differs in important ways. (a) Zachary's Karate Club [16]. (b) Facebook friendships at Caltech [11]. Colour figures can be viewed on the online version of this paper.

2. Concerning *overlapping social contexts*, we point out that the purpose-gathered datasets typically cover only one social context, such as activity in a club, at home or in the workplace. In contrast, mined data from services like Facebook combines interactions from several social contexts, such as personal and professional life, jumbling them together. By combining several contexts into one network, a typical node may belong to several communities, leading to a condition which has been called *pervasive overlap* and which has been shown to cause many community detection algorithms to perform poorly [19–23].
3. Finally, the *size* of mined datasets is often several orders of magnitude larger than purpose-gathered datasets. This is an important difference between the two types of data because the performance of many community detection methods, such as those based on modularity maximization, has been shown to decrease as the size of the network increases [6].

Any one of these differences may cause an algorithm that works well on purpose-gathered datasets to perform poorly on mined datasets. In one sense, researchers recognized that new types of data required new approaches: the emergence of mined datasets inspired many new methods for community detection. Indeed, the field of community detection enjoyed booming popularity as more physicists, computer scientists and social scientists developed these new methods, perhaps motivated by the potential discoveries that could be made in the mined data [24,25].

However, modern community detection methods have in many cases not been evaluated on modern social network data: rather than evaluating these new methods on the mined datasets for which they were designed, the new methods were often evaluated on the old, purpose-gathered datasets or synthetic benchmark networks [7,26–29].[1] Thus, we know that many of the new community detection methods work well on datasets like Zachary's Karate Club or the Southern Women's dataset, but we do not know how well they work on larger, digitally extracted datasets.

2.1 *Existing work on evaluation*

Our point in the preceding paragraphs is that evaluation on large mined *social* network datasets is lacking. However, much work has gone into evaluating these methods on *other types* of large, mined datasets, such as biological and synthetically created networks, and here we briefly review that work.

Community detection methods have been evaluated on diverse types of data. For example, the Gene Ontology and other annotation can be used to evaluate the modules found in protein–protein interaction networks [30] and product categorizations can be used to annotate the network of products co-purchased on Amazon.com [5]. See [19] for an example of thorough benchmarking of community detection methods on many types of large, mined datasets.

The introduction of synthetic network generators such as the planted partition model in [28,31] and its generalization, the Lancichinetti-Fortunato-Radicchi model (LFR) specified in [32,33], have also driven progress in evaluation. In these synthetic benchmarks, a network is artificially created and a known set of 'ground truth' communities (i.e. the set of communities one desires an algorithm to detect) is planted into it; see [20,34–36] for comparative evaluations based largely on synthetic benchmarks.

[1] In some of these papers, a larger social network was evaluated (such as the co-authorship network on arXiv), but these lacked the ground truth or meta-data necessary for a proper evaluation. These larger networks were typically employed only for comparing something other than how well the algorithm identifies all relevant community structure, such as which algorithm gets the highest modularity or runs quickest.

The advantage of this approach is that because the ground-truth set of communities is known, the evaluation procedure is clear-cut. The disadvantage of this approach is that synthetic networks may differ from empirical networks in important but unknown ways—if the field depends too heavily on synthetic benchmarks, then it may develop methods that work well on synthetic data but poorly on real-world data.

There is thus a clear need for benchmarks based on mined social network data. Two recent papers from Yang and Leskovec acknowledge this need and contribute large social network datasets with meta-data which they claim approximates a ground truth of network communities [4,5]. Similarly, a high-quality ensemble of 100 Facebook networks was released by Traud *et al.* [11], which includes node attributes that are considered to be closely related to community structure. If we are to make progress in detecting community structure in modern datasets, then it is imperative that we design benchmarks based on such data. We feel that the papers from Yang and Leskovec, while a step in the right direction, are problematic because they treat the meta-data as a ground truth, and do not sufficiently acknowledge the deficiencies from which the meta-data may suffer. We now proceed to describe some of these deficiencies and how they complicate evaluation.

3. Imperfections in meta-data: incompleteness and nesting

We have so far motivated this paper with the observation that if we want to measure how well community detection algorithms perform on modern mined networks, then we must evaluate them on similar mined networks rather than on small, hand-curated datasets. We now discuss the problem of carrying out evaluation with mined data for which no perfect ground truth exists.

The famous opening lines to Tolstoy's *Anna Karenina* state that 'Happy families are all alike; every unhappy family is unhappy in its own way'. This is also true of the meta-data used to evaluate clustering algorithms, which can be deficient in many different ways. Depending on the exact nature of these deficiencies, one may need to use a completely different evaluation scheme in order to achieve effective evaluation. Here, we focus on the case in which we have an attribute which we assume is closely related to community structure, but which suffers from two deficiencies: incompleteness and nesting. Before describing these deficiencies in detail, it will be useful to establish some terminology.

3.1 *Terminology*

We will define the *true communities* as the ideal output of a community detection algorithm.[2] We will use the term *meta-data* as a catch-all for all of the node attributes. The meta-data associated with a network often contain many attributes, where each maps nodes to some class. For example, the gender attribute maps each node to a single class (i.e. a value), such as male or female. If we run a community detection algorithm on a network, it returns a list of communities which we will call the *found communities.* Note that both an attribute and a set of found communities map each node to zero or more classes, so it is possible to measure the similarity between the two.

[2] One might object to such an idea, and argue that there is no objectively correct set of true communities in a network—that depending on the purposes for which one wishes to use them, there may be several valid, yet distinct, sets of communities in the same network. This is a reasonable objection, but here we simply assume that there is one set of communities that is most generally useful for a wide range of purposes, and this is what is meant we refer to the true communities. While this assumption may seem dogmatic, it underpins the most of the literature on community detection, and it is not an assumption which we will debate here.

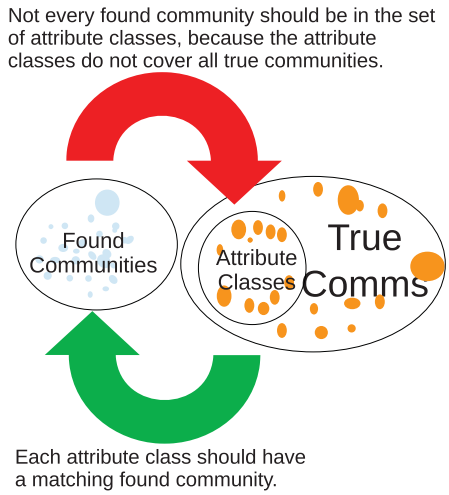

FIG. 2. When we evaluate the accuracy of a community detection algorithm, the attribute data we use may contain only a subset of the true communities. In this case, we cannot demand that each found community corresponds to an attribute class, and so we cannot measure the algorithm's precision. We can, however, demand that each of the attribute classes corresponds to one of the found communities, and so we can measure the algorithm's recall. Colour figures can be viewed on the online version of this paper.

When we describe the quality of a set of found communities, we will do so in terms of *precision* and *recall*. Precision measures the fraction of the found communities that exist in the true communities, whereas recall measures the fraction of true communities that exist in the found communities. For example, imagine a situation where there are 100 true communities, but a community detection algorithm finds only 10 of these (and no extraneous communities): in this case, the precision is 100% while the recall is 10%.

3.2 *Incompleteness and its consequences*

Let us imagine that we have a collegiate social network that we want to use to evaluate the quality of a community detection algorithm. Assume that in addition to the social network, we have data on the dormitory (henceforth, dorm) attribute, which maps each student to the dorm he or she lives in. For now let us assume that each dorm exactly corresponds to a true community (we will relax this assumption below). Let us also assume that we know that in addition to those based around dorms, there are network communities formed by other aspects of social life, such as hobbies and academic pursuits. However, in this imaginary scenario, we do not have data on these other attributes. The right-hand side of Fig. 2 depicts this situation. Thus, the dorm attribute is an *incomplete* enumeration of the true communities.

Now imagine that we run a community detection algorithm on this network and are left with a set of found communities, represented by the circles contained within the 'Found Communities' set in the left half of Fig. 2 (shaded blue online). Given that the dorm attribute is incomplete, one might ask whether it is even possible to evaluate the quality of the found communities. In the terminology defined above, we are able to measure the recall of the community detection algorithm, but not its precision. That is, for each dorm, we know that there should be a matching found community—if there is not, we can say that the algorithm failed to detect a community it should have detected. However, if a found community has no matching dorm, then the situation is ambiguous. It could be that the found community is spurious and does not correspond to a true community, or it may be that while it is a true community, it is not a dorm.

Any evaluation scheme based on an attribute that is incomplete in this way, including the scheme we propose below, will have the following limitation: while it may be able to measure an algorithm's recall, it cannot measure its precision. Given that this problem has plagued the related field of clustering for decades [37], researchers wishing to evaluate community detection algorithms with mined data may have to simply accept this limitation.

3.3 *Network communities may be nested in attribute classes*

The example above included a collegiate social network as well as knowledge of each node's dorm, and for simplicity's sake we assumed that each dorm corresponded exactly to a network community—this is similar to Yang and Leskovec's assumption that user-defined groups correspond to network communities [4,5]. In practice, however, this may be a bad assumption. Even if we have an attribute such as dorm that we know to be closely related to the formation of network communities, it may be that there is not a one-to-one correspondence between the two. In particular, network communities may be *nested* within the attribute's classes. For example, it could be that within each dorm, each incoming freshman class forms its own network community. In this case, the network communities would correspond to the intersection of the dorm and year attributes.

Figure 3 indicates that this example of nesting is not only a hypothetical problem, but one which exists in real data, in this case the University of Chicago's Facebook friendship network from the Facebook100 dataset [11].[3] On the face of it, it seems reasonable to set up a benchmark in which the goal of the community finding algorithm is to detect the 'houses' (residential housing units), which are indicated with solid boxes along the diagonal (coloured blue online). According to the University of Chicago website, 'each house represents a tight-knit community of students, resident faculty masters and residential staff, who live, relax, study, dine together at House Tables, engage, socialize and learn from each other'.

However, the bottom panel of Fig. 3 suggests that while houses are indeed closely related to community structure, they *do not* correspond to network communities. In fact, there may be several times more network communities than houses, and in general each house is several times larger than a network community. In most cases, a single house appears to contain many separate, densely connected subgraphs (indicated by the dashed boxes along the diagonal, coloured brown online) which seem to be network communities. While we have highlighted only a few of these subgroups, if one zooms into the top panel of Fig. 3 and carefully examines the houses along the diagonal, one can observe that in general each house seems to contain many cohesive subgraphs.

4. A classification-based evaluation to cope with imperfect meta-data

We have just described two deficiencies which mined datasets may suffer from: incompleteness and nesting. We now propose that by utilizing an evaluation framework which incorporates a machine learning classifier, one can measure the recall of a community detection algorithm, as defined in Section 3.1.

[3] To create this figure, we have first taken the Facebook friendship network of the University of Chicago and arranged the adjacency matrix such that all of the nodes that belong to the same dormitory are placed in contiguous blocks, indicated by the solid boxes along the diagonal, coloured blue online. Within each block, the nodes are arranged according to the communities found on the subgraph induced by the nodes in the block. Furthermore, the blue blocks themselves are ordered according to the communities found in the 'meta-graph' formed by collapsing each of the blue blocks into a single meta-node and aggregating links between the meta-nodes. For both the subgraph and the meta-graph, the Louvain method of community detection was used [38]. We note that although the Louvain method is in part responsible for this ordering, it has received a large amount of assistance by our first partitioning the network by dorm, and that without this assistance, the method is not nearly as effective at detecting cohesive subgraphs.

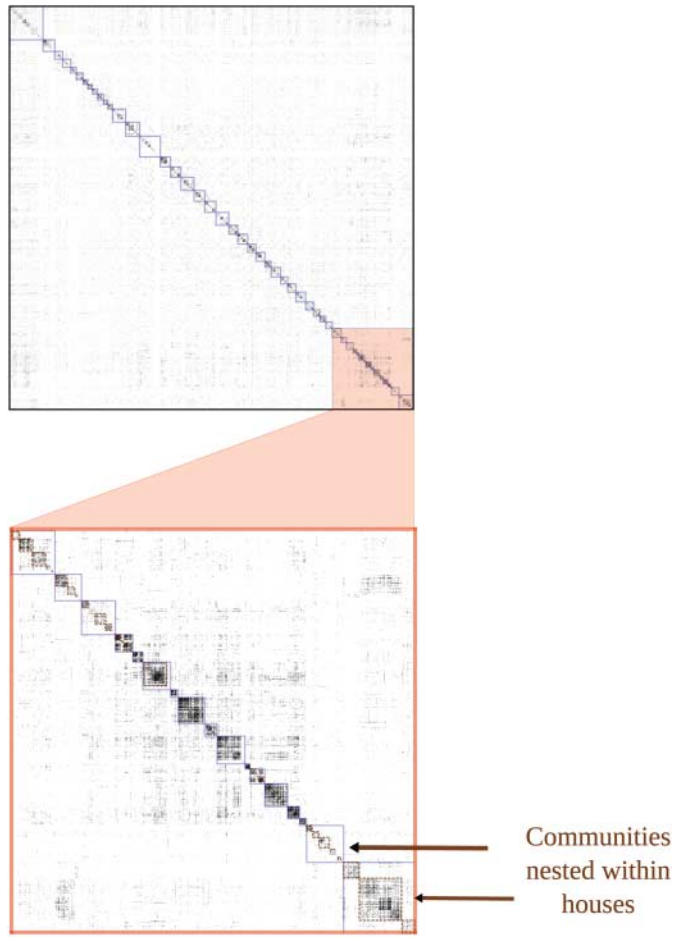

FIG. 3. Upper pane: the adjacency matrix of the University of Chicago, with house membership highlighted (in colour and zommable online). Lower pane: a zoomed-in region suggests that subcommunities exist within houses, and that macro-structure exists between houses. Colour figures can be viewed on the online version of this paper.

The basic idea behind this evaluation scheme is that, with a bit of simple logic, one can map network communities to attribute classes, even if communities are nested within attribute classes, and even if the attribute classes are an incomplete enumeration of network communities. Thus, if an algorithm detects a good set of communities, then a machine learning classifier should be able to discover this mapping.

An example is illustrated in Fig. 4. We represent the mapping from node to found community using the community assignment matrix $\mathbf{X}$ (sometimes called the indicator matrix), and we represent the attribute that we use for evaluation as a vector $\mathbf{y}$. The purpose of the classifier is, given both $\mathbf{X}$ and $\mathbf{y}$, to learn a bit of logic that allows it to map one row of $\mathbf{X}$ to the correct value in $\mathbf{y}$. If the classifier is able to learn this logic, then we can assume that at least some of the found communities are indeed closely related to the attribute, as we expect them to be.

Because we assume that the attribute we use for evaluation is incomplete, many of the true communities are not relevant for inferring the attribute value—hence, a good classifier should learn to ignore the

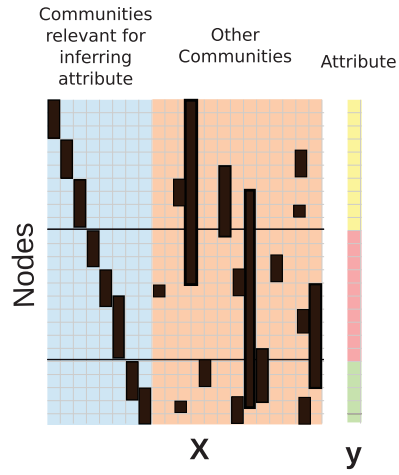

FIG. 4. A machine classifier can detect which network communities are relevant for inferring an attribute. Colour figures can be viewed on the online version of this paper.

'Other Communities' columns on the right hand side of matrix X in Fig. 4 (highlighted in pink online). Many classifiers are designed to deal with the problem of ignoring irrelevant features by performing *feature selection*. Also, due to the nesting issue described in Section 3.3, the classifier should not assume a one-to-one correspondence between the attribute classes and the communities, but rather should be able to flexibly learn more complex mappings between. In this case, the classifier should learn to map nodes belonging to any of the communities represented by the first three columns of **X** to the same attribute value. A classifier is capable of learning such relationships if it has an *expressive hypothesis space*.

Thus, the classifier we choose for our evaluation framework should both perform well at feature selection and have a hypothesis space expressive enough to account for the fact that some network communities may be subsets or supersets of the classes referred to by the attribute. There are many classifiers that fulfil these requirements, including ensemble techniques based on decision trees, such as random forests or stochastic gradient boosting. For those unfamiliar with supervised machine learning, we recommend [39].

We can now propose our meta-data-based evaluation procedure:

1. Select an empirical network and a node attribute thought to be related to community structure. The attribute classes do not need to correspond exactly to communities—it can suffer from incompleteness and nesting. Represent this attribute as a column vector **y**, with one row for each node.
2. Run the community detection algorithm on the network and create a community assignment matrix **X**, as in Fig. 4. Order the rows so that they correspond to **y**.
3. Randomly remove a fraction of the rows **X** along with their corresponding class labels from **y**.
4. Train a classifier on the remaining data in **X** and **y**.
5. Measure the accuracy of the classifier using the data that was set aside in 4.

Steps 4 and 6 involve 'holding out' some data from the classifier and using it to test the accuracy of the classifier. We note that this is the standard way of measuring a classifier's accuracy. By performing this

procedure many times and choosing different data to hold out each time, a technique known as *k-fold cross-validation*, one can arrive at a better estimate of the classifier's accuracy.

After following this procedure, the community detection algorithm is assigned a score: the accuracy with which the classifier can infer missing attribute values. The accuracy of a classifier can be measured in many ways, but we propose the simplest measure of accuracy, which is simply the percentage of predictions that are correct.

A crucial question is how we ought to interpret the scores: when is the accuracy high enough for us to say that an algorithm has performed well, and when is it low enough for us to say that one has performed poorly? The answer is that, in most cases, the absolute value of the accuracy is not very meaningful, and that the accuracy of a method becomes meaningful only when compared with the accuracy of other methods, or of the same method with different parameter settings. In a few exceptional cases where we know that the node attributes are highly correlated to obvious community structure, we may be able to say that an algorithm should be able to score near 100% accuracy, but such datasets will be the exception. The empirical example we present in the following section demonstrates how the relative value of the accuracy can be meaningfully interpreted. There we see that some community detection methods allow classifiers to attain substantially higher accuracy than others. We also see that the methods with lower performance can be improved by tweaking their parameters.

There are some important limitations and drawbacks to this classification-based evaluation. First and foremost, it can only measure recall, and not precision, as mentioned above. In other words, this procedure will in theory not punish an algorithm for detecting many bogus communities.[4] Additionally, this evaluation scheme is appropriate only for the case where a classifier can work out the relation between community structure and node attributes. This should work fine for the case where network communities are rather cleanly nested within attribute classes (as was the case for the example of the University of Chicago data in Section 3), but more complicated relationships may not be discovered by the classifier. Furthermore, if an attribute is not in fact related to the set of true communities at all (imagine eye colour or, in some cases, gender), then the results of the evaluation will be spurious and misleading. Thus, the evaluation proposed here is only appropriate where one has strong *a priori* knowledge that the desired community structure is related to the attribute whose value is being inferred.

Another downside is that setting up such classification tasks is rather complicated, and this makes replication challenging. One must ensure that the classifier performs well both at feature selection and at learning the relationships between network communities and the attribute used for evaluation. If the classifier does not perform well at these two tasks, then we can conclude little from the evaluation: if the resulting classification accuracy is low, then one will not know whether it was because the network communities are uninformative with respect to the attribute (and thus a bad set of communities), or whether the classifier was to blame. Further complexity creeps in when one evaluates the performance of a classifier, which typically involves k-fold cross-validation. In practice, yet another complication is added by computational complexity: depending on the classifier one uses, the size of the network and the number of communities found, it can be computationally expensive to train a classifier. These difficulties can be mitigated by sharing the source code to an easily extensible implementation of the evaluation scheme.

[4] Machine learning classifiers cannot perform feature selection perfectly, and thus are generally adversely affected by a large number of irrelevant features. Thus, in practice, if a method find a many bogus communities, the classifier's accuracy will drop.

5. An empirical example using the Facebook100 dataset

We now provide a concrete example of the classification-based evaluation framework proposed in Section 4. Here, we provide a high-level description of the evaluation and results, leaving some of the details for the appendix. Those seeking even more detailed information, or those seeking to replicate the evaluation with other community detection methods, should refer to the public source code repository. This repository is provided to the research community at large with the aim of allowing it to easily benchmark any algorithm in a replicable manner.[5]

We begin by describing the data used for the evaluation. In Section 3.3, we illustrated an example with a Facebook network representing acquaintanceship at the University of Chicago. This network came from a larger dataset, the Facebook100 dataset from Traud *et al.* [10,11], which includes Facebook data on 100 collegiate networks. The Facebook100 networks range in size from 769 nodes and 17k edges to 36k nodes and 1.6 m edges, and have many desirable characteristics that make the dataset a good candidate for becoming a 'gold standard' for evaluating community detection methods: (1) they come directly from Facebook and are not biased by sampling; (2) they cover universities, which, being 'formally defined groups with well-defined labels', are reasonably coherent social systems, and therefore avoid issues related to the boundary specification problem [40]; (3) at the time of collection, Facebook was popular at universities and so coverage of the social systems is high and, crucially, (4) the datasets include two node attributes which are thought to be related to community structure: dorm and year of graduation.

We now turn our attention to the algorithms evaluated. We benchmark four community detection algorithms: the Louvain method of modularity maximization [38], the InfoMap method of map equation maximization [41], the Link Community (LC) method [19] and the Greedy Clique Expansion (GCE) algorithm [21]. We choose the Louvain method and InfoMap because they are perhaps the two currently most popular methods of community detection. We include the LC method and GCE because they both claim to handle the case of overlapping communities particularly well, and we have reason to believe that in the Facebook data most nodes could belong to multiple communities. For further details on the implementations and parameter settings used, see the appendix.

Our evaluation is limited to the forty networks with the fewest nodes because, performing the evaluation on many of the larger networks, the classifier took prohibitively large amounts of CPU time. In Table 1, we summarize how well each method performs across all 40 networks on both the dorm and year attributes with a histogram, as well as the mean value of these histograms.

We now explain the meaning of these histograms; let us do so by considering the example of the top-left histogram, which shows how well the GCE method performed on the dorm attribute. To create this histogram, we began by running the GCE algorithm on each of the 40 networks and then, for each network, followed the five-step procedure outlined in Section 4. Note that in step 3 of this procedure, we held out a random sample containing 10% of the data. For each network, we were left with an estimate of the accuracy with which a classifier could infer the dorm attribute. It could be the case that if we had held out a different 10% sample of the data, this estimate would differ, so for good measure we performed this procedure three times on each network, holding out a different 10% sample each time.[6] For each of the 40 networks, we were left with three estimates of the accuracy. The histogram includes all 120 of these estimates, and indicates how well the community detection method allows the classifier

[5] The repository can be found at https://github.com/conradlee/network-community-benchmark.

[6] We thus carried out three of the folds of 10-fold cross validation. We did not carry out the other seven due to the computational expense.

TABLE 1 *Each community detection algorithm is run on* 40 *Facebook networks. Then, for each combination of network and algorithm, a classifier is trained to infer the dorm attribute based on the found communities, and the accuracy of this classifier is recorded. The first column of histograms displays the distribution of this accuracy across all* 40 *networks. The mean value of the histogram is reported in the next column. The second column of histograms reports the same distribution for the year attribute. The range of each histogram on the x-axis is from a* 0% *classifier accuracy to* 100% *classifier accuracy. Some summary statistics for the UChicago network is presented in the final two columns—see the main body of text for details.*

	Dorm accuracies		Year accuracies		UChicago stats	
Method	Histogram	Mean	Histogram	Mean	Median smallest	# Comms
GCE		47.0		65.0	43	266
InfoMap		32.7		57.2	171	114
Louvain		25.4		60.0	1016	27
LC		27.9		38.5	4	10
GCE combined		53.4		75.3	22	890
Louvain combined		50.8		73.6	59	278
LC combined		44.7		61.3	5	15731

to infer the dorm attribute in general across all 40 networks. Each mean value in Table 1 is simply the average value of the histogram to the left of it. For details on the classifier used, see the appendix.

In addition to these histograms, we provide some additional statistics related to how each community detection method performed on the University of Chicago network (the right-most two columns, under the heading UChicago Stats). To calculate the *median smallest* column, for each node we first recorded the size of the smallest community it belonged to, and then we took the median of this distribution. The *# Comms* columns simply shows the number of communities found by each algorithm on the UChicago network. Our previous discussion of this network as well as the visualization of the adjacency matrix in Fig. 3 provided us with a basic understanding of the communities network, allowing us to treat it as a case study. While we cannot claim to know the true set of communities in that network, we can at least say that we would expect a community detection algorithm to find at least 100 communities, and that most nodes will belong to at least one community which has fewer than 100 members.

We now turn our attention to interpreting the results presented in Table 1. Let us begin by considering the top four rows, in which we ran each of the four community detection algorithms in the usual manner. We see that GCE has the best performance on inferring values for both the dorm and year attributes; in particular, it performs substantially better than the other methods on the dorm inference task. The UChicago stats indicate that the Louvain and InfoMap methods detected a smaller number of larger communities, whereas GCE tended to find more and smaller communities. In particular, the Louvain method detected only 27 communities, and placed most nodes only in communities with more than 1000 nodes. The LC method also behaved strangely on the UChicago network and several of the other networks: the vast majority of the communities found were singleton communities (containing one edge and two nodes—as we mention in the appendix, these are filtered out before classification occurs), and only 10 of the communities detected in the UChicago network contained four or more nodes. It appears

that the LC method, which is based on cutting a hierarchical clustering, made the cut too low (i.e. too close to the leaves of the dendrogram).

These results indicate the cause of the Louvain method's generally poor performance: in all but the smallest networks it missed the fine-grained community structure. To test this hypothesis out, we also tried two variations of the Louvain method which have been proposed as improving the method's ability to detect structure at all relevant scales. In the appendix, we show that one of these generalizations—based on the parametrized definition of modularity proposed in [42]—does indeed allow the Louvain method to detect smaller-scale structure and improve the classification accuracy.

This improved accuracy suggests that when a community detection method is run only once with the default settings, it may only detect the communities at one scale and that the method may thus fail to find relevant structure at other scales. In the appendix, we also outline a final experiment in which, for each of the three methods with a resolution parameter, we create a combined set of communities by merging communities found using various values of the resolution parameter. The results are presented in the final three rows of Table 1, and they indicate that this merging heuristic based on several runs of the algorithm significantly improves classification accuracy. This result leads us to suspect that the methods benchmarked here, when run with their default settings, fail to detect communities at all relevant scales. This finding is noteworthy because two of the methods, InfoMap (we used the hierarchical variant) and the Louvain Method, both claim to detect community structures at all relevant scales but in fact do not appear to be capable of doing so on this Facebook data.

6. Conclusion

The example evaluation presented in the previous section illustrates how the evaluation framework proposed here can reveal problems that community detection algorithms encounter when run on mined datasets that are much larger than Zachary's Karate Club. As we described in the introduction, modularity's problems have already been described elsewhere, as have some of InfoMap's [43]; so we do not wish to claim that we are discovering these problems for the first time here. However, these problems were only discovered years after modularity and InfoMap had been proposed. We believe that if better evaluation tools had been available, then even before the creators of these methods published them, it might have been clear that something about these methods was not functioning properly, at least on a particular class of empirical data such as Facebook networks. We hope that the evaluation scheme provided here, along with the open-source implementation based on the Facebook100 data, will provide the community detection community with such an evaluation tool.

We conclude by noting that while the evaluation framework proposed here was based on the task of inferring missing node attributes, we could construct conceptually similar benchmarks based on different tasks. One natural example would be to use network communities to perform supervised link prediction [44]; this is a natural fit because presumably the processes responsible for link formation are closely related to the processes which form network communities. Another possibility would be to use network communities to compress the network: because communities are believed to account for many of the edges in a network (and a lack of community structure should indicate a sparse region of the network), they should also be relevant for lossless compression of a graph [45].

There are likely many such tasks for which one could make a case that community structure should be highly relevant. By compiling a collection of these tasks, and measuring how well community detection algorithms perform on the various tasks, we might find that some community detection algorithms are generally more useful for these tasks than others.

Acknowledgements

We thank Sune Lehmann for bringing Conrad Lee out to Copenhagen to discuss this project. We acknowledge Mason Porter's helpful suggestions and corrections. For providing the excellent Facebook100 dataset, we thank Amanda L. Traud, Peter J. Mucha, Mason A. Porter, Eric D. Kelsic and Adam D'Angelo. Thanks are also due to Colm Ryan for his advice on attribute imputation tasks, and to Aaron McDaid for helpful discussions and references.

Funding

This work was supported by Science Foundation Ireland under grant no. 08/SRC/I1407, Clique: Graph and Network Analysis Cluster.

References

1. Fortunato, S. (2010) Community detection in graphs. *Phys. Rep.*, **486**, 75–174.
2. Good, B. H., de Montjoye, Y.-A. & Clauset, A. (2010) Performance of modularity maximization in practical contexts. *Phys. Rev. E*, **81**, 046106.
3. Lancichinetti, A., Kivelä, M., Saramäki, J. & Fortunato, S. (2010) Characterizing the community structure of complex networks. *PLoS One*, **5**, e11976.
4. Yang, J. & Leskovec, J. (2012a) Defining and evaluating network communities based on ground-truth. *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics*, p. 3. ACM.
5. Yang, J. & Leskovec, J. (2012b) Structure and overlaps of communities in networks. *Proceedings of the 6th SNA-KDD Workshop*.
6. Lancichinetti, A. & Fortunato, S. (2011) Limits of modularity maximization in community detection. *Phys. Rev. E*, **84**, 066122.
7. Newman, M. E. & Girvan, M. (2004) Finding and evaluating community structure in networks. *Phys. Rev. E*, **69**, 026113.
8. Berry, J. W., Hendrickson, B., LaViolette, R. A. & Phillips, C. A. (2011) Tolerating the community detection resolution limit with edge weighting. *Phys. Rev. E*, **83**, 056119.
9. Fortunato, S. & Barthelemy, M. (2007) Resolution limit in community detection. *Proc. Natl Acad. Sci.*, **104**, 36–41.
10. Traud, A., Kelsic, E., Mucha, P. & Porter, M. (2011) Comparing community structure to characteristics in online collegiate social networks. *SIAM Rev.*, **53**, 526–543.
11. Traud, A. L., Mucha, P. J. & Porter, M. A. (2012) Social structure of Facebook networks. *Phys. A*, **391**, 4165–4180.
12. Forsyth, E. & Katz, L. (1946) A matrix approach to the analysis of sociometric data: preliminary report. *Sociometry*, **9**, 340–347.
13. Moreno, J. (1934) *Who Shall Survive? : A New Approach to the Problem of Human Interrelations*. Washington, D.C: Nervous and Mental Disease Publishing Co.
14. Davis, A., Gardner, B. & Gardner, M. (1941) *Deep South*. Chicago: University of Chicago Press.
15. Sampson, S. (1968) A novitiate in a period of change: an experimental and case study of social relationships. *Ph.D. Thesis*, Cornell University.
16. Zachary, W. W. (1977) An information flow model for conflict and fission in small groups. *J. Anthropol. Res.*, **33**, 452–473.
17. Freeman, L. C. (2003) Finding social groups: a meta-analysis of the southern women data. *Dynamic Social Network Modeling and Analysis: Workshop Summary and Papers*, pp. 39–97. Washington, D.C: The National Academies Press.
18. Leskovec, J., Lang, K. J., Dasgupta, A. & Mahoney, M. W. (2009) Community structure in large networks: natural cluster sizes and the absence of large well-defined clusters. *Internet Math.*, **6**, 29–123.

19. Ahn, Y., Bagrow, J. & Lehmann, S. (2010) Link communities reveal multiscale complexity in networks. *Nature*, **466**, 761–764.

20. Gregory, S. (2011) Fuzzy overlapping communities in networks. *J. Stat. Mech. Theory Exp.*, **2011**, P02017.

21. Lee, C., Reid, F., McDaid, A. & Hurley, N. (2010) Detecting highly overlapping community structure by greedy clique expansion. *Proceedings of the 2010 SIGKDD Workshop on Social Network Mining and Analysis*. New York: ACM, p. 11.

22. McDaid, A. & Hurley, N. (2010) Detecting highly overlapping communities with model-based overlapping seed expansion. *International Conference on Advances in Social Networks Analysis and Mining (ASONAM), 2010*, pp. 112–119. IEEE.

23. Reid, F., McDaid, A. & Hurley, N. (2011) Partitioning breaks communities. *International Conference on Advances in Social Networks Analysis and Mining (ASONAM), 2011*, pp. 102–109. IEEE.

24. Porter, M. A., Onnela, J.-P. & Mucha, P. J. (2009) Communities in networks. *Notices Amer. Math. Soc.*, **56**, 1082–1097.

25. Watts, D. J. (2004) The "new" science of networks. *Annu. Rev. Sociol.*, **30**, 243–270.

26. Clauset, A., Moore, C. & Newman, M. (2007) Structural inference of hierarchies in networks. *Statistical Network Analysis: Models, Issues, and New Directions*, Springer Berlin Heidelberg, pp. 1–13.

27. Duch, J. & Arenas, A. (2005) Community detection in complex networks using extremal optimization. *Phys. Rev. E*, **72**, 027104.

28. Girvan, M. & Newman, M. (2002) Community structure in social and biological networks. *Proc. Natl Acad. Sci.*, **99**, 7821–7826.

29. Newman, M. (2006) Modularity and community structure in networks. *Proc. Natl Acad. Sci.*, **103**, 8577–8582.

30. Marras, E., Travaglione, A., Chaurasia, G., Futschik, M. & Capobianco, E. (2010) Inferring modules from human protein interactome classes. *BMC Syst. Biol.*, **4**, 102.

31. Condon, A. & Karp, R. M. (2001) Algorithms for graph partitioning on the planted partition model. *Random Struct. Algor.*, **18**, 116–140.

32. Lancichinetti, A. & Fortunato, S. (2009a) Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Phys. Rev. E*, **80**, 016118.

33. Lancichinetti, A., Fortunato, S. & Radicchi, F. (2008) Benchmark graphs for testing community detection algorithms. *Phys. Rev. E*, **78**, 046110.

34. Lancichinetti, A. & Fortunato, S. (2009b) Community detection algorithms: a comparative analysis. *Phys. Rev. E*, **80**, 056117.

35. Lee, C., Reid, F., McDaid, A. & Hurley, N. (2011) Seeding for pervasively overlapping communities. *Phys. Rev. E*, **83**, 066107.

36. Xie, J., Kelley, S. & Szymanski, B. K. (2011) Overlapping community detection in networks: the state of the art and comparative study. Preprint arXiv:1110.5813.

37. Färber, I., Günnemann, S., Kriegel, H.-P., Kröger, P., Müller, E., Schubert, E., Seidl, T. & Zimek, A. (2010) On using class-labels in evaluation of clusterings. *MultiClust: 1st International Workshop on Discovering, Summarizing and Using Multiple Clusterings Held in Conjunction with KDD*.

38. Blondel, V., Guillaume, J., Lambiotte, R. & Lefebvre, E. (2008) Fast unfolding of communities in large networks. *J. Stat. Mech. Theory Exp.*, **2008**, P10008.

39. Bishop, C. M. *et al.* (2006) *Pattern Recognition and Machine Learning*, vol. 1. New York: Springer.

40. Laumann, E. O., Marsden, P. V. & Prensky, D. (1989) The boundary specification problem in network analysis. *Res. Methods Soc. Netw. Anal.*, **61**, 87.

41. Rosvall, M. & Bergstrom, C. (2011) Multilevel compression of random walks on networks reveals hierarchical organization in large integrated systems. *PloS One*, **6**, e18209.

42. Delvenne, J., Yaliraki, S. & Barahona, M. (2010) Stability of graph communities across time scales. *Proc. Natl Acad. Sci.*, **107**, 12755–12760.

43. Schaub, M. T., Lambiotte, R. & Barahona, M. (2012b) Encoding dynamics for multiscale community detection: Markov time sweeping for the map equation. *Phys. Rev. E*, **86**, 026112.
44. Lichtenwalter, R., Lussier, J. & Chawla, N. (2010) New perspectives and methods in link prediction. *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 243–252. ACM.
45. Chierichetti, F., Kumar, R., Lattanzi, S., Mitzenmacher, M., Panconesi, A. & Raghavan, P. (2009) On compressing social networks. *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 219–228. ACM.
46. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. & Duchesnay, E. (2011) Scikit-learn: machine learning in Python. *J. Mach. Learn. Res.*, **12**, 2825–2830.
47. Schaub, M., Delvenne, J., Yaliraki, S. & Barahona, M. (2012a) Markov dynamics as a zooming lens for multiscale community detection: non clique-like communities and the field-of-view limit. *PloS One*, **7**, e32210.

Appendix

The purpose of this appendix is to provide additional details on the empirical evaluation carried out in Section 5 of the main text. For those interested in further details and replicating our experiments, we have put the source code necessary to run these experiments into a public repository so that the research community at large can benchmark their own algorithms in a replicable manner.

Implementations and parameter values used. We used the author's implementation of the Louvain method,[7] which allows for both flat and hierarchical partitions, both of which will be considered below. We also used the author's implementation of the InfoMap algorithm[8] presented in [41], which is designed to detect hierarchical community structure. Likewise, we used the author's C++ implementation of LC, which can detect either a flat or hierarchical clustering, both of which will be tested below.[9] Because LC often found vast numbers of extremely small communities, we removed all communities containing fewer than four nodes or three edges. For GCE, we used the author's implementation,[10] and set the value of the resolution parameter α to 1.5, as this value was recommended for the Facebook data in previous work. We should note that, because GCE has a resolution that has been tuned to this type of data, it has an unfair advantage over the other algorithms; however, in the latter part of this section we also try to optimally tune the resolution parameters of the other methods.

Classifier. Some of the community detection methods benchmarked here detect thousands of communities on these networks; so it is essential that the chosen classifier is good at selecting relevant features in situations with thousands of features; otherwise our benchmark may be biased against methods which detect many communities. After experimenting with several classifiers and feature selection schemes, we found that an ensemble method called stochastic gradient boosting both performed best and was least sensitive to large numbers of communities. In particular, we use the implementation provided in the Python package scikit-learn [46], with the learning rate set to 0.005 and the number of trees set to 1000.[11]

[7] https://sites.google.com/site/findcommunities/.
[8] http://www.tp.umu.se/rosvall/code.html.
[9] https://github.com/bagrow/linkcomm.
[10] https://sites.google.com/site/greedycliqueexpansion/.
[11] There were two more parameter values to set: we required at least five examples for a split in a decision tree (i.e. the max_samples_subsplit parameter= 5) and set the subsampling rate to 0.4. We arrived at these values through experimentation, choosing those values which maximized the performance of the classifier.

TABLE A1 *Performance of variations of the Louvain method of modularity maximization—the left-most column indicates the value of 'Markov time' used. Markov time is a resolution parameter; when set to* 1.0, *the original definition of modularity is recovered. The second column indicates whether a flat cut was used on the dendrogram, or all inner nodes of the dendrogram were used as communities*

Louvain parameters		Dorm accuracies		Year accuracies		UChicago stats	
Markov time	Multi-level	Histogram	Mean	Histogram	Mean	Median smallest	# Comms
1.0			25.4		60.0	1016.0	27
1.0	✓		27.2		61.1	923.0	97
0.5	✓		33.5		62.2	231.0	151
0.2	✓		42.7		63.5	93.0	254

The Louvain method and multi-scale community detection. To further investigate whether the Louvain method's poor performance is due to missing communities at the smallest scale, we perform additional experiments. The scores for the Louvain method presented in Table 1 are based on the optimal flat partitioning. However, as the Louvain method is based on an agglomerative, hierarchical clustering, one can also include communities from all levels of the dendrogram, not just the flat cut which optimizes modularity. Blondel *et al.* [38], the authors of the Louvain method, claim that the method 'unfolds a complete hierarchical community structure for the network', which suggests that the algorithm should detect community structures on all scales. In the second row of Table A1, we present the results when communities detected at all levels are used. We note that the accuracy increases slightly, and that the number of communities found increases—for example, on the University of Chicago network, the number of communities increased from 27 to 97.

The findings of [42] indicate that, to find community structure at all resolutions, the very definition of modularity should be parametrized with a parameter called 'Markov time'. We test this claim by checking whether such a parametrized version of modularity can yield fine-grained communities that improve accuracy. We use the implementation by Renault Lambiotte, which is fortunately based on the very same implementation of the Louvain method and so allows for direct comparison.[12] We set the resolution parameter to 0.5 and 0.2, which are values that should detect community structure on a smaller scale than the unparametrized version of modularity used above, which implicitly sets this value to 1.0. For each of these values, we extract all communities from the dendrogram, as described in the last paragraph. We observe that when the Markov time is decreased, the number of communities detected increases and the accuracy increases significantly for the dorm attribute. As the dorm attribute is more closely associated with finer-scale community structure, this indicates that the resolution parameter does indeed help to find community structure on a smaller scale.

This finding suggests that when modularity maximization techniques are used, then in order to find community structure at smaller scales, it is not enough simply to use a hierarchical clustering technique and make cuts at all levels in the resulting dendrogram. Rather, the very definition of modularity itself must be parametrized with a resolution parameter. While there is much theoretical literature on 'resolution limit' inherent in modularity, here we find strong empirical evidence of this limit.

Our findings here also place the results of Traud *et al.* [11] into doubt. They analysed the community structure in the Facebook100 network using a modularity maximization technique, but paid no

[12] Available at http://www.lambiotte.be/codes.html.

consideration to the resolution limit. Traud *et al.* found that, in larger universities, year of graduation was more relevant for community structure than dormitory assignment. Our results indicate that this finding is likely not inherent in the data, but rather due to a limitation of modularity maxmimization techniques: in larger networks, a naïve application of these techniques does not detect finer-grained communities.

The importance of a resolution parameter for modularity raises the question of whether InfoMap could also perform better if its objective function, the Map Equation, were parametrized with a resolution parameter. As mentioned above, while the implementation of InfoMap that we used is designed to detect community structure at all relevant resolutions, it tended to detect only larger communities. This problem was anticipated by recent work in [43], which also suggests a solution called the 'Markov Sweeping Map Equation', which is parametrized with the Markov time t. This solution embeds the Map Equation in the context of a Markov process that takes place on the graph. A linearization of such a diffusion process provides a natural dynamical parametrization of the original Map Equation, in which the original formulation of Rosvall *et al*. can been seen as the special case where the Markov time $t = 1$. This parametrization is analogous to the linearized diffusion process proposed for modularity, the so-called stability framework [42,47]. However, since a public implementation of the lineraized Markov Sweeping Map equation framework is pending at the time of writing, a comparison with this Markov Sweeping Map equation will be postponed to future work.

Combining multiple runs to find structure at all scales. Leaving InfoMap aside, each of the three other algorithms has a resolution parameter: for the Louvain method, we have the Markov time; for the LC method we have the threshold at which to cut the hierarchical clustering of edges, and GCE has a parameter α which is built into its local objective function. To detect communities at all scales, one could run the algorithm multiple times using different values for the resolution parameter, and then combine the results. In our final experiment, we check whether such a procedure increases the performance on the benchmark. We combine runs of the Louvain method where the Markov time is set to $t = 0.1, t = 0.2, \ldots, t = 1.0$; we combine runs of LC where the threshold for the cut is set to each integer-valued percentage point between 1 and 100 and we combine runs of GCE where α is set to 0.8, 1.0, 1.3, 1.5, 1.7 and 2.2.

When combining several runs of a community detection algorithm, the resulting set of communities can contain a very large set of near-duplicate communities. For example, in a single run, the LC method finds over 100000 communities on some of the Facebook100 networks, and so when several runs are combined, this number can reach into the millions. The vast majority of these millions of communities are near-duplicates of other communities, an undesirable property in most settings, and one which in the current context makes the training of the classifier computationally expensive. When we combine several runs of an algorithm, we therefore remove the near-duplicates by following the procedure outlined in Section 2 of [21], setting ϵ to 0.5; this technique basically removes communities that have a Jaccard similarity of >0.5 with any communities of equal or lesser size.

The results of this final experiment are displayed in the last three rows of Table 1. We see that each method has benefitted by combining the results of multiple runs at different settings of the resolution parameter.

We now wrap up this demonstration benchmark with a summary of our findings. Many community detection techniques strive to be parameter-free so that they can automatically detect communities without requiring a user to experiment with different parameter values [1]. While this is a worthy goal, the results of this section indicate that to achieve good performance, one must tweak the resolution parameter of every method tested here. If we compare the results in the first four rows of Table 1 with those in

last three rows, we see that if one simply trusts the algorithm to automatically set the resolution parameter, then the method may in practice struggle to find structure at all relevant levels. For example, the naive application of the Louvain method produces a set of communities which allow a classifier to infer the dorm attribute with an accuracy of only 25.6%, whereas by combining multiple runs with different values of the Markov time parameter, one can obtain an accuracy of 50.4%. This dramatic increase in performance (as well as our analysis above) indicates that the naive application of the algorithm failed to detect much of the finer-grained community structure.