Social Network Analysis and Mining (2022) 12:20
https://doi.org/10.1007/s13278-021-00846-9

ORIGINAL ARTICLE

Learning network embeddings using small graphlets

Luce le Gorrec[1] · **Philip A. Knight**[1] · **Auguste Caen**[2]

Received: 14 February 2021 / Revised: 13 September 2021 / Accepted: 14 September 2021

Abstract
Techniques for learning vectorial representations of graphs (*graph embeddings*) have recently emerged as an effective approach to facilitate machine learning on graphs. Some of the most popular methods involve sophisticated features such as graph kernels or convolutional networks. In this work, we introduce two straightforward supervised learning algorithms based on small-size graphlet counts, combined with a dimension reduction step. The first relies on a classic feature extraction method powered by principal component analysis (PCA). The second is a feature selection procedure also based on PCA. Despite their conceptual simplicity, these embeddings are arguably more meaningful than some popular alternatives and at the same time are competitive with state-of-the-art methods. We illustrate this second point on a downstream classification task. We then use our algorithms in a novel setting, namely to conduct an analysis of author relationships in Wikipedia articles, for which we present an original dataset. Finally, we provide empirical evidence suggesting that our methods could also be adapted to unsupervised learning algorithms.

Keywords Complex networks · Graph embeddings · Graphlets · Graph classification

1 Introduction

Complex networks are an established tool for analysis in several scientific fields (Estrada and Knight 2015). They can represent a complex system by modelling local interactions between elements within this system—for example, social relations among individuals (Sapiezynski et al. 2019), regulatory relations between genes (He and Tan 2016), etc. The analysis of such networks built on these local interactions allows us to observe and understand phenomena at a larger scale (Viswanath et al. 2009; Cancho et al. 2001; Gargiulo et al. 2016).

Representing networks as low-dimensional vectors—that is, *network embeddings*—enables the use of machine learning approaches to analyse them (Hamilton et al. 2017b). While most research has focused on finding fixed-length vectors to represent nodes of a network (Perozzi et al. 2014; Grover and Leskovec 2016; Hamilton et al. 2017a), techniques to embed the whole network as a unique prescribed-length vector have also appeared which allow comparison between sets of networks, often as a step towards classification (Vishwanathan et al. 2010; Annamalai et al. 2017; Tu et al. 2019; Xu et al. 2019). Reducing networks to these vector forms often involves non-trivial tools and may require a large amount of training data.

Network embedding techniques share the desirable property that networks that share many substructures are close in the embedding space (Gärtner et al. 2003; Shervashidze et al. 2011; Defferrard et al. 2016; Annamalai et al. 2017). This desirable property can be related to motifs in complex networks (Milo et al. 2002, 2004; Stouffer et al. 2007; Mesgar and Strube 2015; Tran et al. 2015; Felmlee et al. 2018). Indeed, it is well-known that induced subgraphs can have a functional meaning and can explain phenomena within the network. For instance, certain stimuli in transcription regulation are related to a certain type of feed-forward loop in pathways (Alon 2007), and in neuronal networks, bifan motifs are known to cooperatively propagate information within synapses (Benson et al. 2016).

In this context, motifs are small subgraphs (typically of three to seven nodes) that appear significantly more often

✉ Luce le Gorrec
luce.le-gorrec@strath.ac.uk

Philip A. Knight
p.a.knight@strath.ac.uk

[1] Department of Mathematics and Statistics, University of Strathclyde, Glasgow G1 1XH, UK

[2] School of GeoSciences and NCEO, University of Edinburgh, Edinburgh EH9 3FF, UK

Published online: 15 December 2021

in a network of interest than in random networks sharing common properties. In particular, networks from a number of common fields (e.g. transcription pathways, neuronal networks, food webs, social networks, etc.) exhibit similarities in three- and four-node motif distributions [also called the significance profile or subgraph ratio profile (SRP)] (Milo et al. 2004). The classical way to build an SRP is to compute the number of occurrences of subgraphs in a sequence of explicitly generated random graphs and to compare these numbers with the corresponding numbers in the real-world network (Milo et al. 2002, 2004; Felmlee et al. 2018).

Based on the observation that motifs can be used to identify network families, we propose two supervised techniques for learning embeddings of directed networks that rely on the distribution of k-node induced subgraphs. Unlike SRP, our algorithms do not compare these distributions with those of random graphs. The comparison with a random model is often a computational bottleneck for large networks, and there is no consensus as to the type of model that should be used to generate random graphs (Milo et al. 2003; Artzy-Randrup et al. 2004; Tu et al. 2019). Thus, not only are our methods effective for graph classification, but they are also robust.

This article is the extended version of the conference paper (le Gorrec and Knight 2020). In this extended version, we conduct in-depth comparisons with existing methods, including graph convolutional networks. We also use our methods to detect and analyse the differences between authors' relations within stable and contested Wikipedia articles, which constitute a new application on an original dataset.

We summarise our contributions as follows.

- *New algorithms* We propose two new supervised methods for embedding networks. The first is based on graphlet counts and a feature extraction procedure built on PCA. Despite its simplicity, it is competitive with or outperforms significantly more complex state-of-the-art methods on a downstream classification task. The second is derived by adapting PCA to a feature selection procedure, thus providing a new measure of feature significance. This feature selection procedure appears to be more meaningful than a popular alternative. Our second algorithm is also competitive and require less supervision than the first.
- *A new application* We use our algorithms to conduct an analysis of author relationships in the French version of Wikipedia. This analysis highlights a correlation between these relationships and the level of controversy in Wikipedia articles.
- *New test data* We introduce two brand new datasets which we believe will be of value to researchers. The first one is of networks we use to benchmark embedding techniques. It is the result of a careful curation of graph structured datasets from a range of repositories. The second provides graphs of author relationships in Wikipedia. It was built using the Wikipedia API to extract article history, along with a home-brewed process for generating relationships graphs.

Supporting our paper, we provide a MATLAB implementation of our algorithms, as well as the two datasets, to be found at le Gorrec et al. (2021).

The paper is organised as follows: Sect. 1.1 presents related work and Sect. 1.2 provides the definitions used throughout the paper. Section 2 describes our algorithms. Comprehensive numerical experiments are then conducted in Sect. 3. In Sect. 4, we use our algorithms to detect controversy within Wikipedia articles. We present our conclusions in Sect. 5.

1.1 Related work

The purpose of this study is to build network embeddings which provide a useful structure for a classification task downstream. These tasks include classifying protein functions, recommending friends or items of interest, and predicting the therapeutic effects of new drugs (Hamilton et al. 2017b). Broadly speaking, embedding techniques can be divided into three families, namely graph kernel methods, extensions of node embeddings and graph neural networks.

Graph kernel methods emerged at the beginning of the twenty-first century and offered a direct method for measuring network similarity that can be interpreted in functional analysis terms (Gärtner et al. 2003; Vishwanathan et al. 2010) and may only provide an embedding implicitly. Methods differ in the means of capturing similarity (Gärtner et al. 2003; Borgwardt and Kriegel 2005; Shervashidze et al. 2011) but the technique closest to ours is (Shervashidze et al. 2009), where the structure of networks is captured by computing the k-node graphlet occurrences (for $k \in \{3, 4, 5\}$), and where the embeddings are explicitly provided. But since the embeddings are not concatenated for the different values of k and no feature reduction procedure is applied, this technique still differs significantly from ours.

In the last decade, a number of methods have been proposed which aim to find embeddings for the nodes of one (or several) network(s) that are consistent with some similarity measure. For instance, one may want to consider nodes to be similar if they share an edge, neighbourhood or structural role and the chosen measure leads to a number of different embedding tools (Ahmed et al. 2013; Perozzi et al. 2014; Ou et al. 2016; Cao et al. 2016; Grover and Leskovec 2016; Hamilton et al. 2017a; Ribeiro et al. 2017). Node embeddings can be used to induce network embeddings, for example using a weighted sum (Hamilton et al. 2017b). However,

network representations based on postprocessing of such node embeddings are known to lead to suboptimal results (Annamalai et al. 2017).

More recently still, automated machine learning methods have been developed that can prove competitive with those based on handcrafted features. For instance, using spectral graph theory (Defferrard et al. 2016), grid-based node ordering (Niepert et al. 2016; Peng et al. 2018), or Weisfeiler–Lehman kernels (Kipf and Welling 2017; Hamilton et al. 2017a), a convolutional neural network can be adapted to a graph setting. Attention mechanisms have also been adapted to create graph attention layers (Veličković et al. 2018). These graph layers are naturally defined to provide one output per node, but can be used for graph classification by adding an aggregation or a pooling layer at the end of the neural network (Xu et al. 2019).

Network embeddings often emerge as a side effect when processing the output prior to its synthesis for a given learning task (e.g. the label of the network for a classification task). To circumvent dependence on the downstream learning task, the authors of `graph2vec` use task-agnostic embeddings (Annamalai et al. 2017), extending the idea behind the neural embedding `doc2vec` from natural language processing.

Finally, we mention `gl2vec` (Tu et al. 2019) which builds a network embedding by computing its three-node SRP. The authors propose a random model which preserves only the number of nodes and edges from the given network, thereby reducing much of the computational cost of enumerating graphlets.

While sharing a focus on graphlets, our work and `gl2vec` exhibit fundamental differences. First, our algorithms do not rely on a random model for comparison. This should make them more robust as there can be considerable variation between SRPs obtained using different random models (Artzy-Randrup et al. 2004). Furthermore, our methods use training samples to perform dimension reduction so strictly speaking our methods are supervised algorithms while `gl2vec` is unsupervised. But by performing the reduction in combination with an enumeration of graphlets with different number of nodes, our method outperforms `gl2vec` while providing embeddings of similar sizes.

1.2 Preliminaries

In this section, we provide some definitions to be used throughout the paper. Networks are assumed to be directed, unweighted without self-loop. The set of all such networks is denoted by $\mathcal{G}$. The network with nodes $V = \{v_1, \dots, v_n\}$ and edges $E = \{(u, v) \in V \times V, u \neq v\}$ is denoted $G = (V, E)$. Given an integer k, k-node graphlets are the set of non-isomorphic graphs in the set

$$\mathcal{G}_k = \{G = (V, E) \in \mathcal{G} : |V| = k \text{ and } G \text{ is connected}\}.$$

We denote by ϕ_k the function that counts the number of occurrences of induced k-node graphlets in a network, that is, $\phi_k : \mathcal{G} \to \mathbb{R}_+^{m_k}$ where m_k is the number of isomorphically distinct k-node graphlets ($m_3 = 13$, $m_4 = 199$ and $m_5 = 9364$) and the ith coordinate of $\phi_k(G)$ is the number of occurrences of the ith k-node graphlet in G, using a standard ordering of graphlets detailed in le Gorrec et al. (2021).

Given vectors $\mathbf{v}_1 \in \mathbb{R}^{n_1}, \dots, \mathbf{v}_p \in \mathbb{R}^{n_p}$, we use the term "concatenation" to mean $[\mathbf{v}_p^T, \dots, \mathbf{v}_p^T]^T \in \mathbb{R}^N$ with $N = \sum_{i=1}^p n_i$ and which we write $\boxplus_{i=1} \mathbf{v}_i$.

The aim of our study is to find embeddings of networks that fit with some downstream classification task. In this context, a network embedding is a function $f : \mathcal{G} \to \mathbb{R}^d$. For the classification task, we have a dataset $\mathcal{D} = \{(G_i \in \mathcal{G}, \ell_i \in \mathcal{L})\}_{i \in \mathcal{I}}$, where $\mathcal{L}$ is the set of the labels. The aim of this task is to find a function $g : \mathcal{G} \to \mathcal{L}$ called a classifier such that for (most) $i \in \mathcal{I}$, $g(G_i) = \ell_i$.

In the following, we will perform a supervised classification for which the dataset $\mathcal{D} = \{(G_i \in \mathcal{G}, \ell_i \in \mathcal{L})\}_{i \in \mathcal{I}}$ is partitioned into a training set and a test set. The indices of elements in the training and test sets will be denoted by $\mathcal{S} \subset \mathcal{I}$ and $\mathcal{T} \subset \mathcal{I}$, respectively.

Throughout the study, we will consider the classifier and the embedding separately, that is, we extend g so that $g : \mathbb{R}^d \to \mathcal{L}$ whereby the whole classification process is then given by $g(f(G))$.

Furthermore, as the focus of our work is on the quality of the embeddings rather than on the classifier, we will use the following straightforward classifier:

$$g(\mathbf{x}) = \arg\min_{\ell \in \mathcal{L}} \|\mathbf{x} - \overline{\mathbf{x}}_\ell\|_2 \tag{1.1}$$

$$\text{where } \overline{\mathbf{x}}_\ell = \frac{1}{|\{i \in \mathcal{S} \text{ s.t. } \ell_i = \ell\}|} \sum_{i \in \mathcal{S}: \ell_i = \ell} f(G_i).$$

2 Network embedding algorithms

In this section, we present our two supervised algorithms to learn network embeddings based on a graphlet count procedure, followed by a dimension reduction process.

The graphlet count procedure is built using a concatenation of the vectors containing the number of occurrences of k-node graphlets, for different values of k. Let $\mathcal{K} \subset \mathbb{N}$ denote the numbers of nodes the graphlets of interest must have (e.g. $\mathcal{K} = \{3, 4\}$ if we investigate three- and four-node graphlets). The network embedding is built using a function $f : \mathcal{G} \to \mathbb{R}^M$, with $M = \sum_{k \in \mathcal{K}} m_k$, such that $\forall G \in \mathcal{G}$:

$$f(G) = \frac{\boxplus_{k\in\mathcal{K}} \alpha_k(G)\phi_k(G)}{\sqrt{\sum_{k\in\mathcal{K}} \alpha_k(G)^2 \phi_k(G)^T \phi_k(G)}} \tag{2.1}$$

where $\alpha_k(G)$ are positive real numbers used to mitigate the order of magnitude difference between the $\phi_k(G)$. We tested several normalisations—see (le Gorrec et al. 2021)-Sect. IV-B—and found that the best results were obtained using $\alpha_k(G) = \frac{m_k}{k!}\binom{n}{k}^{-1}$.

The dimension of vectors returned by f is high (for example, $m_5 = 9364$); however, some graphlets appear extremely rarely, or even not at all. Furthermore, graphlet enumerations produce redundant information and so there is no need to keep each graphlet in the embedding. As shown in le Gorrec et al. (2021)-Sect. IV-D, we can use as few as 18 features to accurately classify networks. We thus combine the function f from (2.1) with dimension reduction in both of our schemes. We define the feature extraction procedure in Sect. 2.1 and the feature selection procedure in Sect. 2.2.

2.1 Feature extraction

We have based our extraction procedure on principal component analysis (PCA) with supervision. Given a dataset containing samples described by real-valued features, PCA can be viewed as a change of basis of the feature space. In a nutshell, it finds an orthogonal basis where the new axes (called principal axes) are sorted such that the first principal axis maximises the variance of the samples if they were projected onto a 1D space, the second principal axis also maximises the variance, but is orthogonal to the first one, and so on. Thus, the p first principal axes form the p-dimensional space which provides the best representation of the dataset, in terms of variance. We use PCA as a supervised feature extraction procedure for our downstream classification task by following the steps proposed in Turk and Pentland (1991). That is, we compute a matrix $\widetilde{\mathbf{U}}$ containing the principal axes of the training set, and we project the samples from both training and test sets onto the lower-dimension space whose directions are provided by the columns of $\widetilde{\mathbf{U}}$.

Once the dimension of the dataset has been reduced, we apply the classifier from (1.1) to these embeddings. The complete process is described in Algorithms 1 and 2, where $\mathbf{e} \in \mathbb{R}^p$ is a vector of ones.

Algorithm 1: Classification with Feature Extraction

Input: $G \in \mathcal{G}$, $\widetilde{\mathbf{U}} \in \mathbb{R}^{M\times k}$, $\overline{\mathbf{x}} \in \mathbb{R}^M$, $\mathcal{C} = \{(\overline{\mathbf{c}_i}, \ell_i) \in \mathbb{R}^k \times \mathcal{L}, i = 1, \ldots, |\mathcal{L}|\}$
Output: $\ell^* \in \mathcal{L}$ the predicted class of G

```
begin
    compute {φ_k(G)}_{k∈K}, {α_k(G)}_{k∈K}
    x ← f(G) according to (2.1)
    c ← (Ũ^k)^T × (x − x̄)
    (c*, ℓ*) = argmin_{(c̄_i, ℓ_i)∈C} ‖c − c̄_i‖_2
return ℓ*
```

Algorithm 2: Preprocessing for Algorithm 1

Input: A training set $\{(G_\sigma, \ell_\sigma) \in \mathcal{G} \times \mathcal{L}, \forall \sigma \in \mathcal{S}\}$, an integer $k \leq M$.
Output: $\overline{\mathbf{x}} \in \mathbb{R}^M$, $\widetilde{\mathbf{U}}^k \in \mathbb{R}^{M \times k}$, $\mathcal{C} = \{(\overline{\mathbf{c}_i}, \ell_i) \in \mathbb{R}^k \times \mathcal{L}, i = 1, \ldots, |\mathcal{L}|\}$

begin
  for $\sigma \in \mathcal{S}$ **do**
    compute $\{\phi_k(G_\sigma)\}_{k \in \mathcal{K}}, \{\alpha_k(G_\sigma)\}_{k \in \mathcal{K}}$
    $\mathbf{x}_\sigma \leftarrow f(G_\sigma)$ according to (2.1)
  $\overline{\mathbf{x}} \leftarrow \frac{1}{|\mathcal{S}|} \sum_{\sigma \in \mathcal{S}} \mathbf{x}_\sigma$
  apply PCA on $\mathbf{X} = \left[\mathbf{x}_{\sigma_1} \ \cdots \ \mathbf{x}_{\sigma_{|\mathcal{S}|}}\right]^T$ to obtain $\widetilde{\mathbf{U}} \in \mathbb{R}^{M \times k}$
  $\mathbf{C} \leftarrow (\mathbf{X} - \mathbf{e} \times \overline{\mathbf{x}}^T) \times \widetilde{\mathbf{U}}$
  $\mathcal{C} \leftarrow \emptyset$
  for $\ell \in \mathcal{L}$ **do**
    $\overline{\mathbf{c}_\ell} \leftarrow \frac{1}{|\{\sigma \in \mathcal{S} : \ell_\sigma = \ell\}|} \sum_{\sigma \in \mathcal{S} : \ell_\sigma = l} \mathbf{c}_\sigma$
    $\mathcal{C} \leftarrow \mathcal{C} \cup \{(\overline{\mathbf{c}_\ell}, \ell)\}$
return $\overline{\mathbf{x}}$, $\widetilde{\mathbf{U}}$, $\mathcal{C}$

A potential limitation is that PCA cannot exploit available knowledge about the training set labels. But extracting via Fisher's discriminant analysis (FDA) (Theodoridis 2015), which resolves this issue, provides no tangible benefit and so we stick with PCA. Comprehensive comparison between PCA and FDA can be found in le Gorrec et al. (2021)-Sect. IV-A.

2.2 Graphlet selection

Algorithm 1 starts by counting all k-node graphlets for $k \in \mathcal{K}$ for each network. This operation is expensive for large networks and can be avoided. Furthermore, the construction of the projection matrix $\widetilde{\mathbf{U}}$ in Algorithm 1 is strongly dependent on the training set. To avoid these drawbacks, we have designed a feature selection procedure to replace the feature extraction so that we can avoid both the full computation of graphlets and the projection using $\widetilde{\mathbf{U}}$.

We build our feature selection procedure by imitating PCA-based feature extraction methods, namely by expressing it as a problem for maximising the so-called inertia. First, we use this maximisation problem to showcase important advantages of PCA-based procedures. Then, we state the problem to solve for our feature selection procedure, and in Theorem 1, we show that our procedure benefits from similar advantages. Finally, we use the theorem to derive a measure, given in (2.6), to select the most significant features.

Maximising the inertia. Suppose $\mathbf{X} = \left[\mathbf{x}_1 \ \ldots \ \mathbf{x}_n\right]^T \in \mathbb{R}^{n \times M}$ is a dataset in which each row is a sample, and where $\overline{\mathbf{x}}$ is the centre of gravity (i.e. the average sample). The inertia of $\mathbf{X}$ is a scalar $\mathcal{I}(\mathbf{X})$ which measures the deviation of the samples around the centre, that is,

$$\mathcal{I}(\mathbf{X}) = \frac{1}{n} \sum_{i=1}^{n} \|\mathbf{x}_i - \overline{\mathbf{x}}\|_2^2. \tag{2.2}$$

To simplify our exposition, we will assume that $\overline{\mathbf{x}} = 0$, but the results hold without this assumption.

PCA-based extraction procedures can be expressed using inertia (Abdi and Williams 2010) since finding the q principal axes is equivalent to finding the q-dimensional vector space that maximises the inertia of the projections of the rows of $\mathbf{X}$. This can be written as follows:

$$\begin{cases} \underset{\mathbf{Y} \in \mathbb{R}^{n \times q}}{\arg\max} \ \mathcal{I}(\mathbf{Y}) \\ \text{with} \quad \mathbf{y}_i = \mathbf{U}^T \mathbf{x}_i, \\ \text{and} \quad \overline{\mathbf{y}} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{y}_i, \\ \text{subject to} \quad \mathbf{U}^T \mathbf{U} = \mathbf{I}_q. \end{cases} \iff \begin{cases} \underset{\mathbf{U} \in \mathbb{R}^{M \times q}}{\arg\max} \ \mathcal{I}_{\mathbf{X}}(\mathbf{U}) = \frac{1}{n} \sum_{i=1}^{n} \|\mathbf{U}^T \mathbf{x}_i\|_2^2 \\ \text{subject to} \quad \mathbf{U}^T \mathbf{U} = \mathbf{I}_q. \end{cases} \tag{2.3}$$

That is, the columns of the matrix $\widetilde{\mathbf{U}}^q$ that solves (2.3) are the q principal axes of dataset $\mathbf{X}$.

A convenient property follows: We can append a column to the q-dimensional solution to obtain the $q+1$-dimensional solution, that is,

$$\widetilde{\mathbf{U}}^{q+1} = \left[\widetilde{\mathbf{U}}^{q} \mid \mathbf{u}_{q+1} \right],$$

giving a consistent sorting of features by significance order (Abdi and Williams 2010).

We also remark that (2.3) is equivalent to finding the q-dimensional affine space that minimises the approximation error, that is, if $\widetilde{\mathbf{U}}^q$ solves (2.3), then it is also the solution of

$$\underset{\mathbf{U}\in\mathbb{R}^{M\times q}:\mathbf{U}^T\mathbf{U}=\mathbf{I}_q}{\arg\min} \mathcal{E}_{\mathbf{X}}(\mathbf{U}) = \frac{1}{n}\sum_{i=1}^{n} \|\mathbf{U}\mathbf{U}^T\mathbf{x}_i - \mathbf{x}_i\|_2^2.$$

Feature selection procedure Our aim is to build a feature selection procedure—that is to pick k vectors from the canonical basis—based on $\widetilde{\mathbf{U}}^q$. We design our feature selection procedure as a problem of inertia maximisation. Denoting by $\widehat{\mathbf{X}} = \mathbf{X}\widetilde{\mathbf{U}}^q\widetilde{\mathbf{U}}^{qT} = \left[\hat{\mathbf{x}}_1 \ \dots \ \hat{\mathbf{x}}_n\right]^T$ the reconstruction of $\mathbf{X}$ after its projection in the q-dimensional vector space, we aim to solve:

$$\begin{cases} \underset{\mathbf{Z}\in\mathbb{R}^{n\times k}}{\arg\max}\ \mathcal{I}(\mathbf{Z}) & \\ \text{with} & \mathbf{z}_i = \mathbf{E}^T\hat{\mathbf{x}}_i, \\ \text{and} & \overline{\mathbf{z}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{z}_i, \\ \text{subject to} & \mathbf{E} = \left[\mathbf{e}_{\sigma_1} \ \dots \ \mathbf{e}_{\sigma_k}\right] \\ \text{and} & \mathbf{E}^T\mathbf{E} = \mathbf{I}_k \end{cases} \iff \begin{cases} \underset{\mathbf{E}\in\mathbb{R}^{M\times k}}{\arg\max}\ \mathcal{I}_{\widehat{\mathbf{X}}}(\mathbf{E}) = \frac{1}{n}\sum_{i=1}^{n}\|\mathbf{E}^T\hat{\mathbf{x}}_i\|_2^2 \\ \text{subject to} \quad \mathbf{E} = \left[\mathbf{e}_{\sigma_1} \ \dots \ \mathbf{e}_{\sigma_k}\right] \\ \text{and} \quad \mathbf{E}^T\mathbf{E} = \mathbf{I}_k \end{cases} \tag{2.4}$$

where $\mathbf{e}_j$ is the jth column of the identity matrix.

Theorem 1 *$\widetilde{\mathbf{E}}^k$, the solution of (2.4), defines the basis of a vector space in which projections of rows of $\widehat{\mathbf{X}}$ also minimise the approximation error. Furthermore, the solution of the $k+1$-dimension problem can be written as $\widetilde{\mathbf{E}}^{k+1} = \left[\widetilde{\mathbf{E}}^k \mid \mathbf{e}_{\sigma_{k+1}}\right]$.*

These assertions establish the analogy with PCA.

Proof First, observe that $\overline{\hat{\mathbf{x}}} = \overline{\mathbf{x}} = 0$ and $\overline{\mathbf{z}} = 0$. Then, for any $\mathbf{F}$, a feasible matrix for (2.4), we have $\mathcal{I}_{\widehat{\mathbf{X}}}(\mathbf{F}) = \frac{1}{n}\sum_{i=1}^{n}\|\mathbf{F}^T\hat{\mathbf{x}}_i\|^2 = \frac{1}{n}\sum_{i=1}^{n}\hat{\mathbf{x}}_i^T\mathbf{F}\mathbf{F}^T\hat{\mathbf{x}}_i$.

Furthermore, the approximation error of $\widehat{\mathbf{X}}$ from $\mathbf{F}$ can also be rewritten as:

$$\begin{aligned}\mathcal{E}_{\widehat{\mathbf{X}}}(\mathbf{F}) &= \frac{1}{n}\sum_{i=1}^{n}\|\mathbf{F}\mathbf{F}^T\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_i\|_2^2 = \frac{1}{n} \\ &\sum_{i=1}^{n}\left(\hat{\mathbf{x}}_i^T\mathbf{F}\underbrace{\mathbf{F}^T\mathbf{F}}_{\mathbf{I}_k}\mathbf{F}^T\hat{\mathbf{x}}_i - 2\hat{\mathbf{x}}_i^T\mathbf{F}\mathbf{F}^T\hat{\mathbf{x}}_i + \hat{\mathbf{x}}_i^T\hat{\mathbf{x}}_i\right) \\ &= -\mathcal{I}_{\widehat{\mathbf{X}}}(\mathbf{F}) + \frac{1}{n}\sum_{i=1}^{n}\|\hat{\mathbf{x}}_i\|^2\end{aligned}$$

and since $\|\hat{\mathbf{x}}_i\|^2$ does not depend on $\mathbf{F}$, minimising $\mathcal{E}_{\widehat{\mathbf{X}}}(\cdot)$ is equivalent to maximising $\mathcal{I}_{\widehat{\mathbf{X}}}(\cdot)$.

To show that $\widetilde{\mathbf{E}}^{k+1} = \left[\widetilde{\mathbf{E}}^k \mid \mathbf{e}_{\sigma_{k+1}}\right]$, we remark that we can rewrite $\mathcal{I}_{\widehat{\mathbf{X}}}(\mathbf{F})$ as the sum of some diagonal elements of the covariance matrix of $\widehat{\mathbf{X}}$, that is,

$$\begin{aligned}\mathcal{I}_{\widehat{\mathbf{X}}}(\mathbf{F}) &= \sum_{t=1}^{k}\mathbf{S}_{\widehat{\mathbf{X}}}(\sigma_t,\sigma_t), \text{ with } \mathbf{S}_{\widehat{\mathbf{X}}} = \frac{1}{n}\widehat{\mathbf{X}}^T\widehat{\mathbf{X}} \\ &= \frac{1}{n}\widetilde{\mathbf{U}}^q\begin{bmatrix}\lambda_1 & & \\ & \ddots & \\ & & \lambda_q\end{bmatrix}\widetilde{\mathbf{U}}^{qT}\end{aligned} \tag{2.5}$$

with $(\lambda_t, \mathbf{u}_t = \widetilde{\mathbf{U}}^q\mathbf{e}_t)$ eigenpairs of the covariance matrix[1] of $\mathbf{X}$.

Now suppose that $\mathbf{E}^* = \left[\mathbf{e}_{\pi_1} \ \dots \ \mathbf{e}_{\pi_k} \ \mathbf{e}_{\pi_{k+1}}\right]$ and $\mathbf{F}^* = \left[\mathbf{e}_{\sigma_1} \ \dots \ \mathbf{e}_{\sigma_k}\right]$ are solutions of (2.4) for dimension $k+1$ and k, respectively. If it were possible to find $i \in \{1,\dots,k\}$ so that $\mathbf{e}_{\pi_i} \notin \mathbf{F}^*$, then $\mathcal{I}_{\widehat{\mathbf{X}}}(\left[\mathbf{F}^* \mid \mathbf{e}_{\pi_i}\right]) > \mathcal{I}_{\widehat{\mathbf{X}}}(\mathbf{E}^*)$, which contradicts the assumption that $\mathbf{E}^*$ is a solution of (2.4). □

From Theorem 1, we see that it is possible to write the global inertia $\mathcal{I}_{\widehat{\mathbf{X}}}(\widetilde{\mathbf{E}}^k)$ as the sum of the individual contributions from each canonical axis. This enables us to derive a score from the first q principal axes that assesses the significance of a canonical axis.

Definition 1 Following the notation of (2.5), the expression level of the ith canonical axis within the q first principal axes is defined to be

[1] Principal axes are the eigenvectors of the covariance matrix associated with the highest eigenvalues—see Abdi and Williams (2010).

$$\gamma(i) = \frac{\sum_{t=1}^{q} \lambda_t \mathbf{u}_t(i)^2}{\sum_{s=1}^{q} \lambda_s}. \tag{2.6}$$

The measure $\gamma(i)$ gives an indication of the significance of the ith canonical axis, and is normalised so that $\sum_{i=1}^{M} \gamma(i) = 1$. It allows one to derive a feature selection procedure for networks in the test set: knowing the q principal axes of the training set, one can choose the k canonical axes with the highest γ-score to embed the networks from the test set. We arrive at a set $\Gamma \subset \{1, \dots, M\}$, $|\Gamma| = k$, of indicators of the best features to select.

To derive our embedding function, we split the indicators of graphlets according to the number of nodes t. That is, we build

$$\Gamma_3 \subset \{1, \dots, m_3\}, \Gamma_4 \subset \{1, \dots, m_4\}, \dots, \Gamma_p \subset \{1, \dots, m_p\},$$

such that

$$\Gamma = \bigcup_{t=3}^{p} \left(\Gamma_t + \sum_{s=3}^{t-1} m_s \right), \quad \text{with}$$
$$\Gamma_t + \sum_{s=3}^{t-1} m_s = \left\{ \sigma_i + \sum_{s=3}^{t-1} m_s, \sigma_i \in \Gamma_t \right\}.$$

Our embedding function thus becomes

$$h(G) = \frac{\boxplus_{t=3}^{p} \alpha_t(G) \psi_t(G, \Gamma_t)}{\sqrt{\sum_{t=3}^{p} \alpha_t(G)^2 \psi_t(G, \Gamma_t)^T \psi_t(G, \Gamma_t)}}, \tag{2.7}$$

with $\alpha_t(G)$ as defined in (2.1), and

$$\psi_t(G, \Gamma_t) = \mathbf{P}_t \times \phi_t(G),$$

with

$$\mathbf{P}_t \in \mathbb{R}^{|\Gamma_t| \times m_t} \text{ s.t. } \mathbf{P}_t(i,j) = \begin{cases} 1 & \text{if } \Gamma_t(i) = j, \\ 0 & \text{otherwise.} \end{cases}$$

The γ-measure we defined above allows one to simplify the classification algorithm presented in Sect. 2.1. Algorithm 3 is an adaptation of Algorithm 1 based on this γ-measure, and Algorithm 4 contains preprocessing steps for Algorithm 3.

Algorithm 3: Classification with Feature Selection

Input: $G \in \mathcal{G}$,
$\Gamma_3 \subset \{1, \dots, m_3\}$, …, $\Gamma_p \subset \{1, \dots, m_p\}$,
$\mathcal{C} = \{(\overline{\mathbf{z}_i}, \ell_i) \in \mathbb{R}^k \times \mathcal{L}, i = 1, \dots, |\mathcal{L}|\}$
Output: $\ell^* \in \mathcal{L}$ the predicted class of G

```
begin
    for t = 3, ..., p do
        compute ψ_t(G, Γ_t), α_t(G_σ)
    z ← h(G) according to (2.7)
    (z*, ℓ*) = argmin_{(z̄_i, ℓ_i) ∈ C} ||z − z̄_i||_2
return ℓ*
```

Algorithm 4: Preprocessing for Algorithm 3

Input: A training set $\{(G_\sigma, \ell_\sigma) \in \mathcal{G} \times \mathcal{L}, \forall \sigma \in \mathcal{S}\}$; two integers $k, q < M$; p max. size of graphlets.
Output: $\mathcal{C} = \{(\overline{\mathbf{z}_i}, \ell_i) \in \mathbb{R}^k \times \mathcal{L}, i = 1, \ldots, |\mathcal{L}|\}$, $\Gamma_3 \subset \{1, \ldots, m_3\}$, ..., $\Gamma_p \subset \{1, \ldots, m_p\}$

begin
  for $\sigma \in \mathcal{S}$ **do**
    for $t = 3, \ldots, p$ **do**
      compute $\phi_t(G_\sigma), \alpha_t(G_\sigma)$
    $\mathbf{x}_\sigma \leftarrow f(G_\sigma)$ according to (2.1)
  $\overline{\mathbf{x}} \leftarrow \frac{1}{|\mathcal{S}|} \sum_{\sigma \in \mathcal{S}} \mathbf{x}_\sigma$
  apply PCA on $\mathbf{X} = \left[\mathbf{x}_{\sigma_1} \ \ldots \ \mathbf{x}_{\sigma_{|\mathcal{S}|}}\right]^T$ to obtain the q ppal. axes $\mathbf{U} \in \mathbb{R}^{M \times q}$
  $\Gamma \leftarrow \mathbf{0}_{k \times 1}$, $\Lambda \leftarrow \mathbf{0}_{k \times 1}$
  for $t = 1, \ldots, M$ **do**
    if $\gamma(t) > \min(\Lambda)$ **then**
      $i^* \leftarrow \arg\min_{i=1,\ldots,k}(\Lambda(i))$
      $\Lambda(i^*) \leftarrow \gamma(t)$
      $\Gamma(i^*) \leftarrow t$
  $m_{tmp} \leftarrow 0$
  for $t = 3, \ldots, p$ **do**
    $\Gamma_t \leftarrow \{t - m_{tmp}, \forall t \in \Gamma : t \leq m_t\}$
    $m_{tmp} \leftarrow m_{tmp} + m_t$
  for $\sigma \in \mathcal{S}$ **do**
    $\mathbf{z}_\sigma \leftarrow h(G_\sigma)$ according to (2.7)
  $\mathcal{C} \leftarrow \emptyset$
  for $\ell \in \mathcal{L}$ **do**
    $\overline{\mathbf{z}_\ell} \leftarrow \frac{1}{|\{\sigma \in \mathcal{S} : \ell_\sigma = \ell\}|} \sum_{\sigma \in \mathcal{S} : \ell_\sigma = l} \mathbf{z}_\sigma$
    $\mathcal{C} \leftarrow \mathcal{C} \cup \{(\overline{\mathbf{z}_\ell}, \ell)\}$
return $\Gamma_3, \ldots, \Gamma_p, \mathcal{C}$

2.3 Complexity

Recall that $\mathcal{S}$ and $\mathcal{T}$ correspond to the training set and the test set, respectively, and that $\mathcal{L}$ is the set of classes (or labels). Let us denote by p the number of axes extracted from the PCA. Our first algorithm, presented in Sect. 2.1, performs the following steps:

(1) Compute $f(G_\sigma)$ as stated in (2.1) for each network G_σ, with $\sigma \in \mathcal{S} \cup \mathcal{T}$.
(2) Use the training set $\mathcal{S}$ to:
  (a) Build a frame of dimension p from the PCA applied on $\mathcal{S}$.
  (b) Project the learning samples onto this new frame.
  (c) Compute a representative of each label in $\mathcal{L}$ within the new frame.
(3) Project the samples from the test set $\mathcal{T}$ within the new frame.
(4) Label each sample with the class of the closest representative.

Assume we make use of k-node graphlets, with $k \in \mathcal{K}$. Then, the output of $f(G)$ in Step 1 is of size $M = \sum_{k \in \mathcal{K}} m_k$, where m_k is the number of k-node graphlets. The complexity of Step 2(a), which is dominated by PCA, is $O\left(\min\left\{M^3, |\mathcal{S}|^3\right\}\right)$. The projections at Steps 2(b) and 3 have a complexity of $O(|\mathcal{S}|Mp)$ and $O(|\mathcal{T}|Mp)$. The computational cost of Step 2(c) is $O(|\mathcal{S}|p)$, and for Step 4 it is $O(p|\mathcal{T}||\mathcal{L}|)$. In general, the dominant cost for training and test sets is at Step 1, in counting the number of graphlets. In our implementation, we use the `acc-Motif` algorithm: given a graph G with n nodes and m edges, it counts all three-node graphlets in G in $O(nm)$, all four-node graphlets in $O\left(m^2\right)$, and all five-node graphlets in $O\left(m^2 n\right)$ (Meira et al. 2014). This has to be done for every graph in $\mathcal{S} \cup \mathcal{T}$. We note that these complexities may not be optimal. As far as we know, Lin et al. (2012) provides the best complexity algorithm for counting 4-node graphlets, namely $O\left(n + min\{m^{1.61}, \alpha^2 \times m\}\right)$, where α is the arboricity of the graph—bounded by $m^{1/2}$ for a connected graph.

```
l1:   # 4 inputs
l2:   # 1 outputs
l3:   # 3 D-type flip-flops
l4:   # 3 inverters
l5:   # 8 gates(1 ANDs +1 NANDs
                +2 ORs +4 NORs)
l6:   INPUT(G0)
l7:   INPUT(G1)
l8:   INPUT(G2)
l9:   INPUT(G3)
l10:  OUTPUT(G17)
l11:  G5 = DFF(G10)
l12:  G6 = DFF(G11)
l13:  G7 = DFF(G13)
l14:  G14 = NOT(G0)
l15:  G17 = NOT(G11)
l16:  G8 = AND(G14,G6)
l17:  G15 = OR(G12,G8)
l18:  G16 = OR(G3,G8)
l19:  G9 = NAND(G15,G16)
l20:  G10 = NOR(G14,G11)
l21:  G11 = NOR(G5,G9)
l22:  G12 = NOR(G1,G7)
l23:  G13 = NOR(G2,G12)
```

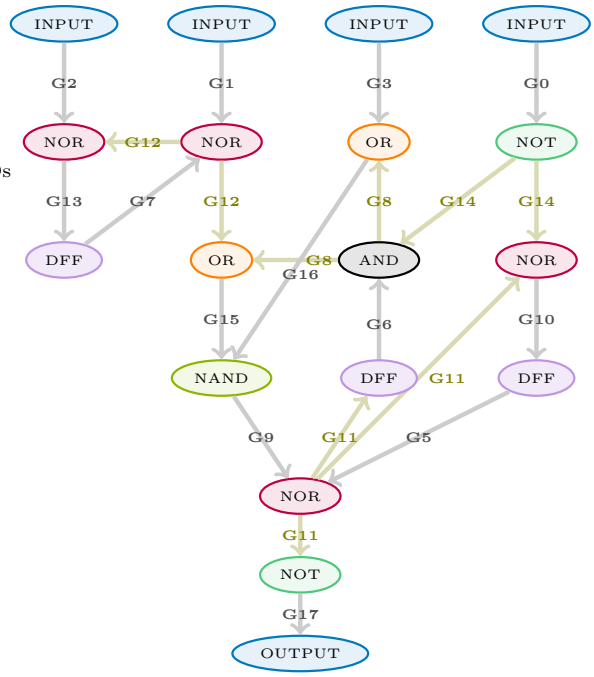


Fig. 1 Left: Netlist of the electronic circuit `s27` from Brglez et al. (1989). Right: The resulting network, with labels displayed. Each gate/input/output corresponds to one node. The edges are signal transfers

γ_1 : John had a lovely evening last night.
γ_2 : He had a great meal.
γ_3 : He ate salmon.
γ_4 : He devoured lots of cheese.
γ_5 : He won a dancing competition.
Γ_6 : $\gamma_2 \cup \gamma_5$
Γ_7 : $\gamma_3 \cup \gamma_4$

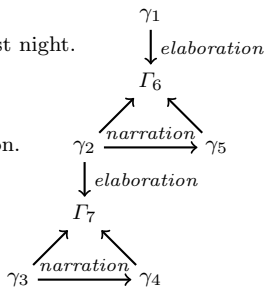


Fig. 2 Left: A short text divided into discourse units. The EDUs are identified by a γ, the CDUs by a Γ. Right: Discourse structure of the text with a network representation. The edge labels correspond to rhetorical relations. An edge with no label links an EDU to the CDU it is a member of

In practice, when counting four-node graphlets on a laptop, `acc-Motif` was capable of dealing with our dataset—whose largest graph has about 870K edges—in a reasonable length of time. Obviously, if one wants to deal with larger-scale networks, other methods should be employed, among which parallel or approximate methods should be considered. See Ribeiro et al. (2021) for a recent survey listing such methods.

The complexity of our second method is *a priori* similar to the first algorithm if we suppose that we extract the selected graphlets after having performed the count of all k-node graphlets for all networks. While we still have to perform this whole count for networks within the training set, in the test set, it is actually possible to compute the number of occurrences of the selected graphlets only. This trick can amortise the complexity. Indeed, enumeration algorithms such as `GTrie` (Ribeiro and Silva 2014) or `QuateXelero` (Khakabimamaghani et al. 2013) can benefit from this simplification. Furthermore, the best four-node graphlet complexity algorithm from Lin et al. (2012) computes the number of occurrences of each four-node graphlet in turn, and can thus be accelerated by performing this for only a subset of four-node graphlets. Finally, some

Table 1 Number of nodes (n) and edges (m) of the networks

	Food webs		Elec. circ.		Disc. struct.		Soc. net.	
	n	m	n	m	n	m	n	m
Min	51	113	54	71	50	53	51	114
Max	214	5643	24,097	52,344	237	285	82,168	870,161
Mean	104.4	1023	3358.9	6006.7	75.92	94.46	6430	48,371

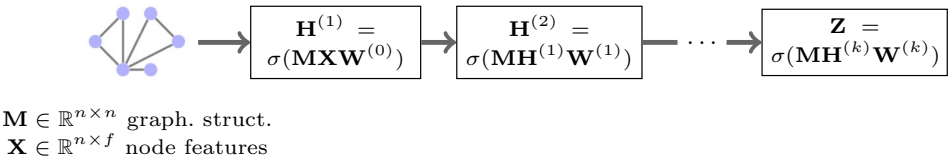


Fig. 3 Pipeline of a GCN

efficient algorithms exist for enumerating selected graphlets. For instance, in Chiba and Nishizeki (1985), an algorithm for counting quadrangles, and another for closed triads both run in $O(m^{1.5})$ time.

3 Numerical experiments

In this section, we compare the quality of the embeddings provided by our new algorithms on our new benchmark dataset against some existing alternatives (`gl2vec`, `graph2vec`, graph convolutional networks (`GCNs`), and relational graph convolutional networks (`RGCNs`)).

This section is organised as follows: In Sect. 3.1, we present our dataset; in Sect. 3.2, we outline the existing methods, and in Sect. 3.3, we discuss the results.

3.1 Testing benchmark

Existing benchmarks for testing network classification algorithms tend to be dominated by small, undirected networks (Annamalai et al. 2017; Kipf and Welling 2017) which limits their effectiveness in assessing algorithms designed for use on directed graphs.

To overcome this limitation, we have curated a new testing benchmark to use in our numerical experiments. It contains directed networks with between 50 and 80K nodes, from four different application fields, namely food webs, electronic circuits, discourse structures and social networks. Here, we give brief information about the networks, as well as any post-processing we have applied to them to obtain networks that are well-suited to our analysis—unweighted directed static networks without self-loops. The complete dataset can be found at le Gorrec et al. (2021)-Sect. II.

Food webs. Food webs represent the consumer–resource relations between species in an ecosystem. A relation between species a and b indicates that a consumes, preys, or acts parasitically on b. In these networks, the nodes are the species, and there is an edge from a to b if species a consumes species b. Edges can be weighted to represent either a level of confidence in the relation, or the quantity of species b that is consumed by species a. In such cases, we have kept all edges but removed the weight. And occasional self-loops, representing cannibalism, have been excised.

In total, we have collected 70 food webs, primarily from the (www.globalwebdb.com) (Batagelj and Mrvar 2006) websites and an article focused on the impact of parasitism (Dunne et al. 2013).

Electronic Circuits. Here, an electronic circuit is an object composed of one or more inputs that provide one or more outputs by passing through logic gates (*AND*, *NOR*, etc.). It can be represented by a network where a node is an input, an output or a logic gate, and an edge indicates that a signal enters or leaves a logic gate. For instance, the relation

$$\Sigma_1 = AND(\Sigma_2, \Sigma_3)$$

is represented by one node, with two edges entering the node (signals Σ_2 and Σ_3), and one edge leaving the node (signal Σ_1). An example for circuit `s27` from Brglez et al. (1989) is provided in Fig. 1.

For this field, 52 networks have been extracted from two classical benchmarks, namely ISCAS'89 (Brglez et al. 1989) and ITC'99 (Corno et al. 2000), found in the repositories (Jenihhin 2020; Clauset et al. 2016).

Discourse Structures Segmented Discourse Representation Theory (Asher and Lascarides 2003) can be used to represent the rhetorical relations between discourse units by means of networks. An example extracted from Lascarides and Asher (2007) is shown in Fig. 2 to explain how the network representation of a discourse is built. The discourse units can be of two kinds: elementary discourse units (EDUs) or complex discourse units (CDUs) that correspond to units composed of several EDUs.

We have extracted the networks from the STAC dataset (www.irit.fr/STAC/corpus.html) Asher et al. (2016), which is a corpus of chat conversations between players from an online version of the game board "The Settlers of Catan". Two corpora are available to download: a linguistic one, in which there are only human exchanges (players and observers), and a situational one, which is the linguistic corpus supplemented with information about the game given by the computer by means of canned messages—such as "It's ⟨player id⟩ to roll the dice", "⟨player id⟩ ends its turn", etc.. We have extracted 195 networks—22 from the linguistic corpus, 173 from the situational corpus. As shown in Fig. 2, the edges are labelled. We do not consider these labels in our study.

Social relations The term "social relations" covers a wide range of relation types, from friendship relations on online

social networks to physical contacts; and from commercial exchanges to genealogical relations, or even domination among a group of animals. In our study, we have limited ourselves to human relations that are directed and can be symmetrical—this excludes, for instance, genealogical or student–teacher relationships. The resulting dataset consisting of 81 networks can be roughly divided into two kinds of relations, namely feelings and exchanges.

Networks representing feelings are mainly based on surveys of groups of individuals, such as pupils, students or co-workers, who were asked to name their friends, or who they turn to to ask for advice or support, etc. In such networks, the edges may be weighted according to the strength of the feeling, both positive and negative. We discard the weights and draw an edge whenever individual a expresses a feeling for individual b. The edges may also be labelled according to the nature of the relation if several kinds of relations have been tested. In this case, we have created one network per relation type, and one network merging all the relation types.

Networks representing exchanges (emails, phone calls, messages on online social websites) may be dynamic, may have multi-edges, and self-loops are sometimes allowed since, for example, one can send an email to oneself. After removing loops, we have transformed these networks into static unweighted networks by adding an edge from a to b so long as there is at least one exchange at some time from a to b.

Finally, 72 of our networks represent feelings: 41 between school students, 16 between co-workers (both primarily taken from surveys detailed in Freeman 2020), ten from website users and five miscellaneous. The remainder represents exchanges between individuals. These exchanges are mainly emails or phone calls. As for the food webs, the listing provided by Clauset et al. (2016) was very helpful for sourcing the data.

In summary, we have extracted networks from four different fields, 70 representing food webs, 52 networks corresponding to electronic circuits, 195 discourse structure networks and 81 social networks. Some general statistics are provided in Table 1.

3.2 Alternative approaches

Here, we present existing methods against which we compare our algorithms. These are two network embedding techniques: `gl2vec` and `graph2vec`, and two algorithms for classifying networks: graph convolutional networks (`GCNs`) and relational graph convolutional networks (`RGCNs`).

`gl2vec` This method, proposed in Tu et al. (2019), is based on the notion of motifs. To build network embeddings, `gl2vec` counts the number of occurrences of three-node graphlets and compares them against the expected number of instances of such three-node graphlets to occur in random networks. Given a network G, it computes its embedding $\psi(G) \in \mathbb{R}^{16}$ via the formula

$$\psi(G)_i = \frac{N_b(G)_i - \overline{N_r(G)_i}}{N_o(G)_i + \overline{N_r(G)_i} + \varepsilon}, \quad \forall i \in \{1, \dots, 16\},$$

where $N_o(G)_i$ is the number of instances of graphlet of type i that occurs in G and $\overline{N_r(G)_i}$ is the corresponding expected number that occurs in Erdős–Rényi random networks having the same number of nodes and edges as G. The scalar ε is chosen to avoid numerical problems linked to small denominators. The embeddings produced by `gl2vec` have been computed using the implementation from Tu (2018).

`graph2vec` This method, proposed in Annamalai et al. (2017), is a neural embedding framework that aims to learn network embeddings in an unsupervised fashion. It is based on two fundamental tools: the Weiseiler–Lehman (WL) test, used to define the context of a node, and an unsupervised learning process inspired by `doc2vec`, to derive graph embeddings. The WL test is a relabelling process that tests whether two graphs are isomorphic. Each node starts with its own label (the standard choice for a directed graph is a pair containing the in-and out-degrees), and then receives the list of its neighbours' labels. A new label is created for this node, two nodes having the same labels if and only if their previous labels and the list they received are equal. These steps are repeated d times, so that at the end of the process, two nodes have the same labels if and only if their d-hop neighbourhoods are indistinguishable.

Document embeddings produced by `doc2vec` are built by learning word embeddings from the word context (i.e. words coming before and/or after the word of interest), and using these embeddings to derive a vector representation of a document that maximises the log likelihood of words in this document.

By considering a network as a document with nodes playing the role of words, and their d-hop neighbourhood playing the role of context, the authors of Annamalai et al. (2017) produce task-agnostic embeddings for networks within a corpus.

To produce embeddings using `graph2vec`, we have used the implementation from Rozemberczki et al. (2020). Some parameters need to be set up in `graph2vec`, the most important being the depth of context to take into account (the d in the WL part), and the size of the embeddings produced by `doc2vec`. We tested with different sizes for both. For other parameters, we have used the default values from Rozemberczki et al. (2020).

In our experiments the best results are provided by a depth of 1 for the WL test (see le Gorrec et al. 2021 Sect. V-A). Our interpretation of this is that deep WL tests are not able to capture similarity in our dataset because of the wide range in the number of nodes and edges of networks

Table 2 Quality measurement of the embeddings classified by (1.1)

		Precision		Recall		F1-score	
		Mean	Std	Mean	Std	Mean	Std
Our method	Fw	9.50	2.7	9.78	2.6	9.64	1.9
	Elec	9.44	6.3	8.86	8.4	9.11	5.8
	Disc	9.86	0.8	9.95	0.3	9.90	0.4
	Soc	10	0	9.61	2.5	9.80	1.3
RF feat. select	Fw	9.17	4.0	8.74	6.7	8.93	4.0
	Elec	4.30	12.3	7.77	18.2	5.46	12.8
	Disc	9.70	2.2	9.77	3.7	9.73	2.4
	Soc	8.16	9.5	6.08	13.0	6.91	11.0

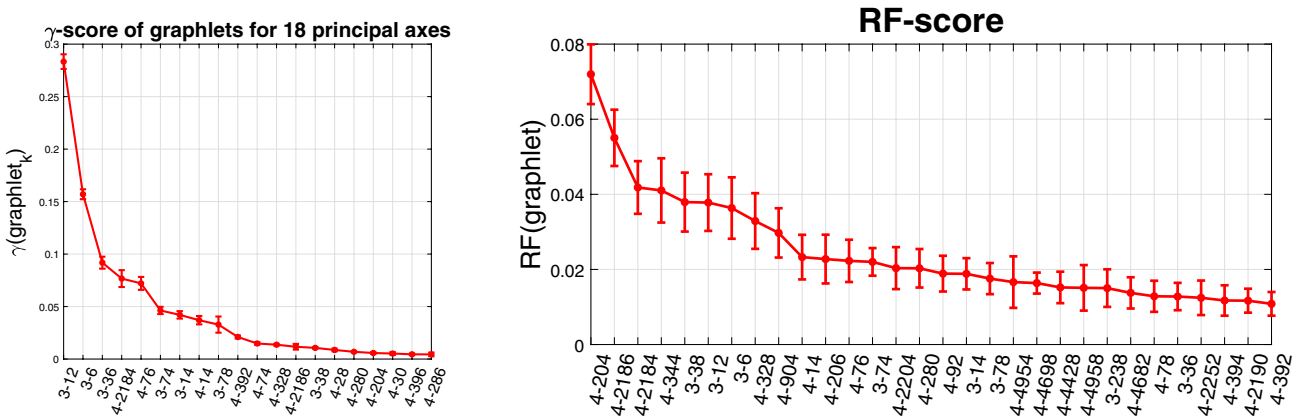


Fig. 4 Left: The 20 graphlets with the highest average γ-score. Right: The 30 graphlets with the highest score returned by Random Forest

in a class—see Table 1. Indeed, the WL test is explicitly designed so that networks that are far from isomorphic have very different labellings (Xu et al. 2019). In particular, two networks of different size will always be regarded as dissimilar by the WL test.

(R) GCN Graph convolutional networks (GCNs), proposed in Kipf and Welling (2017), are an instance of graph neuronal networks. They aim to produce node embeddings for some downstream classification task, such as clustering or community detection. A GCN is composed of k graph convolutional layers, as shown in Fig. 3. Each layer i can be written as:

$$\mathbf{H}^{(i+1)} = \sigma\left(\mathbf{M}\mathbf{H}^{(i)}\mathbf{W}^{(i)}\right) \tag{3.1}$$

with σ some nonlinear function (typically reLU), $\mathbf{M} \in \mathbb{R}^{n\times n}$ represents the graph structure, the rows of $\mathbf{H}^{(i)} \in \mathbb{R}^{n\times d_i}$ contain the ith hidden node embeddings, and $\mathbf{W}^{(i)} \in \mathbb{R}^{d_i\times d_{i+1}}$ are the weight matrices to be learnt.

Generally, $\mathbf{M}$ is some transformation of the adjacency matrix of the graph $\mathbf{A} \in \mathbb{R}^{n\times n}$. Classical choices are $\mathbf{M} = \mathbf{D}_{in}^{-1/2}\mathbf{A}\mathbf{D}_{out}^{-1/2}$ or $\mathbf{M} = \mathbf{A}\mathbf{D}_{in}^{-1}$, where $\mathbf{D}_{in}$ and $\mathbf{D}_{out}$ are diagonal matrices of in- and out-degrees. $\mathbf{H}^{(0)}$ contains the initial node features. If no feature is known, $\mathbf{H}^{(0)}$ can be a vector containing node degrees, an $n \times 2$ matrix containing node in/out-degrees, or the identity matrix.

Relational GCNs Schlichtkrull et al. (2018) have been adapted from GCNs to deal with graphs with different kinds of relations—and thus represented by a set of adjacency matrices $\{\mathbf{M}_r \in \mathbb{R}^{n\times n}\}_{r\in Relations}$—by allowing some weights to be different for each relation. Here, we use RGCNs as a tool to take into account edge directions in our networks. That is, we consider that a directed edge from node u to node v results in two different kinds of relations: u "points to" v; and v "is pointed at by" u. Thus, the adjacency matrices of our relations are actually the adjacency matrix of the directed graph and its transpose.

To classify graph instances instead of nodes from a (R)GCN, one has to add a layer to merge node embeddings, such as a pooling layer, and a dense layer that returns a vector whose dimension is equal to the number of classes (Xu

Table 3 Quality measurement of embeddings.

		Precision		Recall		F1-score		Run time	#1 %
		Mean	Std	Mean	Std	Mean	Std		
	fw	9.52	2.3	**10**	0	**9.75**	1.2	15min	**73**
Algo 1	Elec	**10**	0	8.93	9.7	9.41	5.7		39.6
	Disc	9.88	0.9	9.95	0.3	9.92	0.5		12.4
	Soc	**10**	0	9.67	2.2	**9.83**	1.1		**81.4**
	fw	9.50	2.7	9.78	2.6	9.64	1.9	15min	40
Algo 3	Elec	9.44	6.3	8.86	8.4	9.11	5.8		14
	Disc	9.86	0.8	9.95	0.3	9.90	0.4		7.6
	Soc	10	0	9.61	2.5	9.80	1.3		76.2
	fw	**9.78**	2.3	8.87	4.8	9.30	3.0	3min	1.6
gl2vec	Elec	9.67	6.1	6.82	11.0	7.94	7.8		0
	Disc	9.87	0.6	**10**	0	9.94	0.3		**58.4**
	Soc	9.07	3.3	**10**	0	9.51	1.8		6.4
graph2vec	fw	9.59	3.6	9.69	3.6	9.63	2.5	2min	33.8
Depth1	Elec	9.98	1.1	**9.75**	3.9	**9.86**	2.0		**99**
Emb64	Disc	**9.90**	0.6	**10**	0	**9.95**	0.3		58.2
	Soc	9.78	2.5	9.34	4.0	9.55	2.5		17.4
GCN	fw	8.10	4.2	9.09	7.7	8.55	4.4	1h30	
Mean	Elec	6.24	16.8	5.49	16.0	5.69	14.1		
2 layers	Disc	9.72	13.7	10	14	9.88	13.8		
Emb12	Soc	9.80	3.3	8.08	5.7	8.84	3.1		
RGCN	fw	9.29	4.6	9.55	4.3	9.40	2.7	2h45	
Mean	Elec	6.95	23.6	7.78	20.9	7.12	19.7		
2 layers	Disc	9.80	23.3	9.85	23.7	9.82	11.8		
Emb8	Soc	9.67	15.13	8.54	7.2	9.05	23.4		

For each class of networks and each metric, bold values indicate the best score

et al. 2019). Using a pooling layer has the benefit that we can avoid learning new variables.

In testing (see le Gorrec et al. 2021 Sect. V-B), we found that sum pooling shows very poor results compared to maximum and mean pooling layers. This is related to our observation about deep WL tests in `graph2vec`. It was proved in Xu et al. (2019) that a sum pooling at the end of a `GCN` tends to separate non-isomorphic networks from each other. Our last remark is that `RGCN`s outperform `GCN`s, which was expected since `RGCN`s can take advantage of the direction of edges in the networks from our dataset, while `GCN`s cannot.

3.3 Comparison of methods

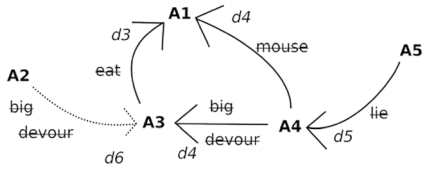
To assess the capacity of our methods to recover the labels of the networks in the benchmark described in Sect. 3.1, we divide our database into a training set and a testing set. The training set $\mathcal{S}$ is built by randomly selecting $N_B = 40$ networks from each class/field. Thus, $|\mathcal{S}| = |\mathcal{L}| \times N_B$. The remaining networks form the test set $\mathcal{T}$. We repeated this process on 50 pairs of sets $(\mathcal{S}, \mathcal{T})$, and we present the mean and standard deviation of the results. To improve presentation, the means and standard deviations have been multiplied by 10 and 100, respectively. In the tables, the labels *fw*, *elec*, *disc* and *soc* relate to the food web, electronic circuit, discourse structure and social relationship benchmarks, respectively. All the experiments were conducted on a laptop with 32GB RAM and a `Core i5 vPro 8th Generation` processor. To compute the ϕ_k in (2.1), we have used Version 2.2 of `acc-Motif` Meira et al. (2014).

We compare our methods against those described in Sect. 3.2. First we measure the consistency of the γ-scores of graphlets, described in Sect. 2.2, and then we discuss the capacity of classification.

Set up Detailed discussion about our set up and the optimisation of parameters for all methods can be found in le Gorrec et al. (2021). In our PCA-based approaches, we use $\mathcal{K} = \{3, 4\}$. We observed in le Gorrec et al. (2021) Sect. IV-C that when combining these graphlet sizes, the results are very similar whether we use $\mathcal{K} = \{3, 4\}$ or

Date	Author	Version of the article	Lemmatised article
d_1	A_1	The cat eat the mice	cat eat mouse
d_2	A_2	The cat eats the mice.	cat eat mouse
d_3	A_3	The big cat devours the mice.	big cat devour mouse
d_4	A_4	The cat is a lie!!	cat lie
d_5	A_5	The big cat devours the mice	big cat devour mouse
d_6	A_2	The cat eats the mice.	cat eat mouse

Fig. 5 An example of how we build an authors network from a Wikipedia article



$\mathcal{K} = \{3, 4, 5\}$. Since the computational cost of counting 5-node graphlets is enormous (and proved impossible in our set up on networks with more than 10K nodes), we chose to keep the combination of three- and four-node graphlets only. For the number of axes in the feature extraction procedure, we kept 18 principal axes, as justified in le Gorrec et al. (2021) Sect. IV-D. Optimisation of existing alternatives is described in le Gorrec et al. (2021) Sect. V.

Feature Selection To measure the consistency of the γ -scores of graphlets, we assessed them over the 50 training sets. We plot the error bar of the 20 graphlets with highest average γ-score in the left plot of Fig. 4. The vertical bars indicate a range of plus or minus one standard deviation around the average of each γ-score of a graphlet. Identifiers of graphlets are provided on the x-axis. More detail about the identifiers can be found in le Gorrec et al. (2021)-Sect. III.

One can see that graphlet 3–12 ( ) has by far the highest score, accounting for almost a third of the total inertia in the space of the q principal axes. There is a noticeable drop in the curve between the ninth and tenth scores, and so we have chosen to keep only the nine graphlets with the highest average γ-score.

As a comparison, we have built input subsets Γ_3 and Γ_4 using a feature selection based on random forests (RF) (Breiman 2001) as detailed in le Gorrec et al. (2021)-Sect. VI. This implementation allows one to get a score of importance for each feature after having trained the RF model, which is the average Gini importance over all trees. For a fair comparison, we have kept the nine graphlets with the highest Gini importance to build Γ_3 and Γ_4, and then applied Algorithm 3.

Results are presented in Table 2. It is clear that selecting features based on γ-score provides far better results than using RF based feature selection. For every label, our new method is significantly better both in terms of precision and recall. Moreover, we can see in the right panel of Fig. 4, that

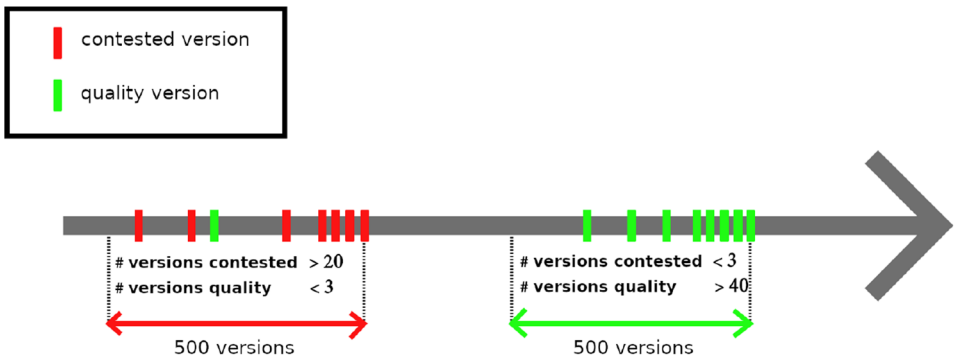


Fig. 6 Extracting time slots containing versions with tags of interest

Table 4 Scores of Algorithm 1 for discriminating among quality and contested articles

	Precision		Recall		$F1$-score	
	Mean	Std	Mean	Std	Mean	Std
Quality	4.78	3.9	7.69	6.3	5.87	2.3
Contested	8.90	2.0	6.83	6.7	7.71	3.9

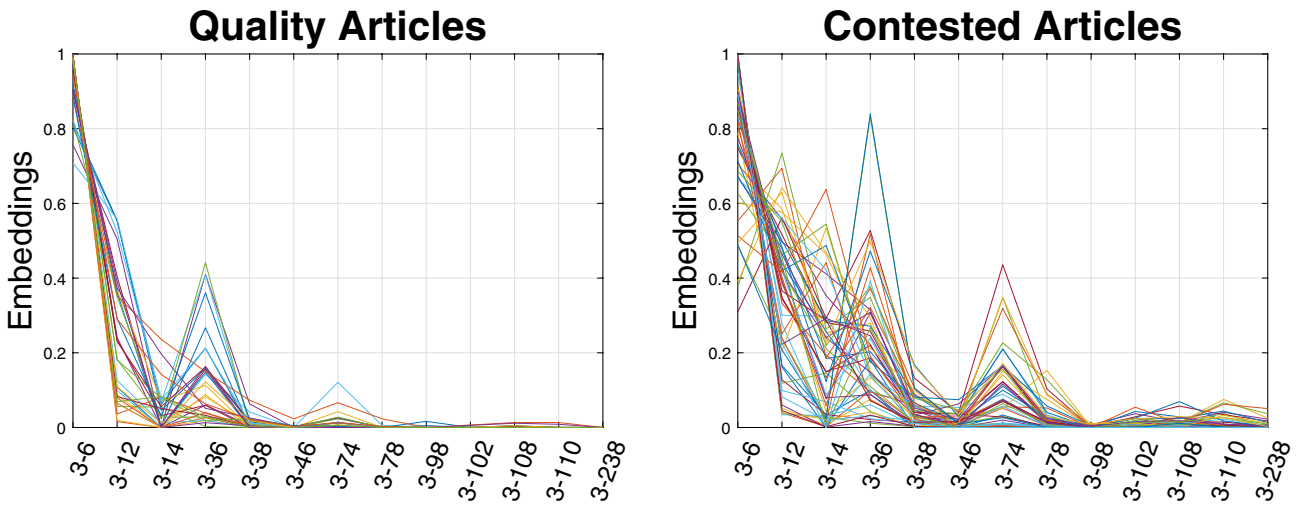


Fig. 7 Embeddings of quality and contested articles, for three-node graphlets

the shape of the curve of Gini importance over graphlets is much less sharp at the cut-off point than the γ-score curve.

Classification task We now compare the result of Algorithms 1 and 3 with the methods described in Sect. 3.2. The classifier from (1.1) is applied to embeddings provided by `graph2vec` and `gl2vec`, while for the `(R)GCNs`, we have trained a $d \times 4$ dense layer after the pooling, and associated with a network the class with the highest coordinate in the output of this layer. Results are given in Table 3.

While in our tests our method is slower than `gl2vec` and `graph2vec`, this can be improved greatly in a more practical implementation. As stated in Sect. 2.3, the most time-consuming part is the calls to `acc-Motif` (the time spent for performing the other steps is negligible). In our tests, these calls were done sequentially, but they are easily parallelisable. The maximum time for one call, namely 116s, reflects an achievable target for such a step. Other methods for improving the run time, especially for Algorithm 3, are discussed in Sect. 2.3. Note that `gl2vec` proposes three random models for computing the SRPs, and we have chosen the only one for which the run times were not prohibitive. Finally, for `graph2vec`, we note that, for deeper WL-kernels (1 being the smallest possible value), the run times naturally increase.

The results show that `(R)GCNs` cannot compete with the other simpler embedding methods. We believe this is because the training sets we fed these methods were too small to enable them to discriminate among classes of networks. With the architecture, we use, both neural network models need to learn around 200 weights for graph convolutional layers and the dense layer, and we provide only 140 networks for training and 20 networks for validation. This emphasises that for small datasets simpler techniques may be preferable.

We observe that all embedding techniques return consistent clusters for each class. With the exception of electronic circuit networks embedded via `gl2vec`, all $F1$-scores are above 0.9. We remark that `graph2vec` outperforms `gl2vec` as well as our method on the discourse structure networks (for which the gain is slight as all methods achieve very good $F1$-score), and on electronic circuits (where the gain is quite substantial). On the other hand, our method significantly outperforms `gl2vec` and `graph2vec` on food webs and social networks. We note that these last two classes are less well-defined in the sense that they are built from observations.

By comparing our two algorithms, we remark that Algorithm 3 produces similar results to Algorithm 1 at lower cost. The previous conclusions remain true: our second method is more accurate than `gl2vec` and `graph2vec` in uncovering food webs, and social networks. The accuracy for the rhetorical discourse benchmark is still high. With feature selection, there is a noticeable decrease in accuracy when identifying electronic circuits. Nevertheless, even with this decrease, it remains significantly more accurate than `gl2vec`.

Finally, in the last column of Table 3, we check whether the inferences we have made above hold in general. For each class, we display the percentage of times each embedding technique returns the best $F1$-score. This process has been repeated over 500 pairs of training/testing sets. With only one exception, we observe that the method with the highest average $F1$-score is also the one which provides the best score the most often. Furthermore, the difference between the best method and the runner-up is very large in almost all cases. The only exception is in the comparison between `graph2vec` and `gl2vec` on discourse structure. However, we remark that on this benchmark, both methods achieve essentially the same results.

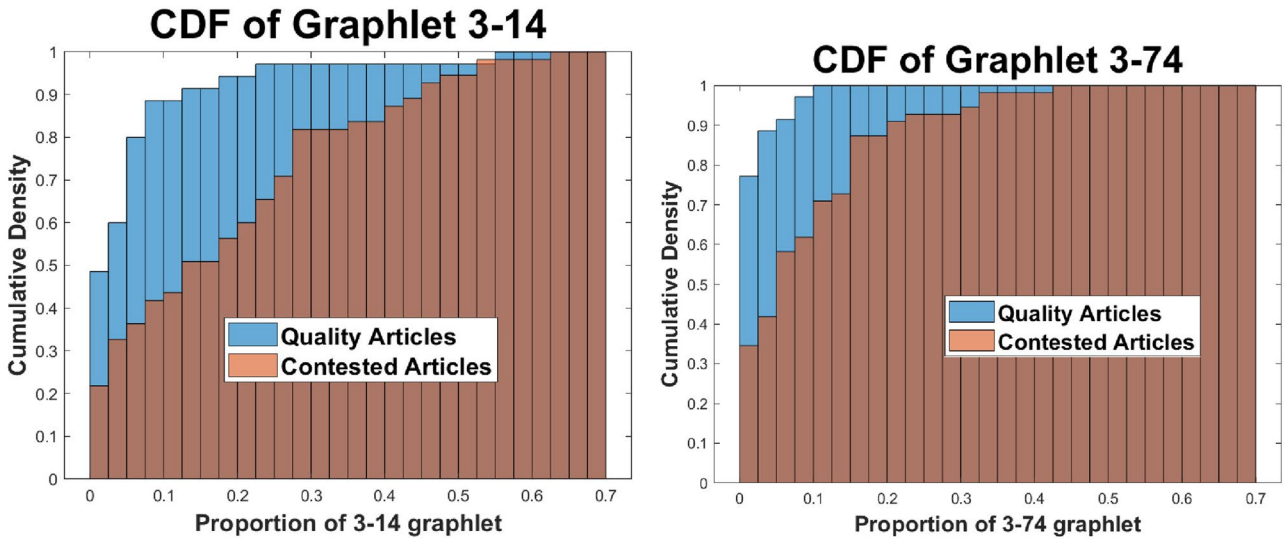


Fig. 8 Cumulative distributions of graphlets 3–14 and 3–74 within quality and contested articles

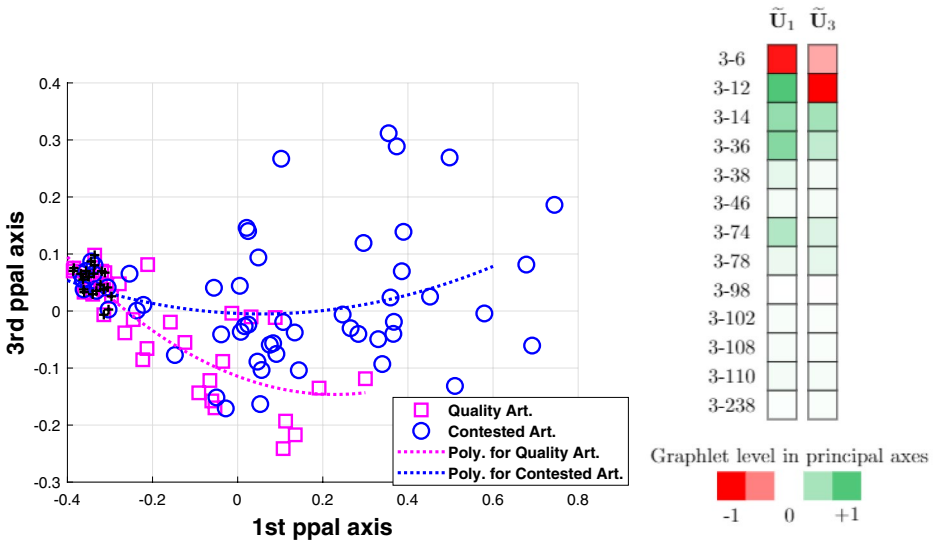


Fig. 9 Left: Projections of article graphs on the first ($\widetilde{\mathbf{U}}_1$) and third ($\widetilde{\mathbf{U}}_3$) principal axes. Right: Coordinates of principal axes within the graphlet basis

To conclude, despite their conceptual simplicity, our methods achieve better or comparable results than existing alternatives. We also note that while `graph2vec` is competitive with our methods, it required a careful setting of its hyperparameters and its performance strongly depends on the WL-test depth (see le Gorrec et al. 2021-Sect. V-A).

4 Detecting controversy and quality in wikipedia articles

Wikipedia's great success is due in part to the scope and structure of the information it provides, and to its reputation for seriousness and objectivity (Head and Eisenberg 2010). One consequence is that activists and lobbyists try to influence Wikipedia articles (Oboler et al. 2010), leading to arguments about article contents (Yasseri et al. 2012; Borra et al. 2014). In this section, we use the methods designed in Sect. 2 to detect and analyse the level of controversy and quality potential of Wikipedia articles. To do this, we use an original dataset extracted from French Wikipedia, with networks representing relationships between authors of articles. We describe our dataset in Sect. 4.1, and provide our analysis in Sect. 4.2.

4.1 Dataset

Wikipedia is an online collaborative encyclopedia, containing articles that *a priori* everyone with an internet connection can amend. A Wikipedia article can be seen as a stack of

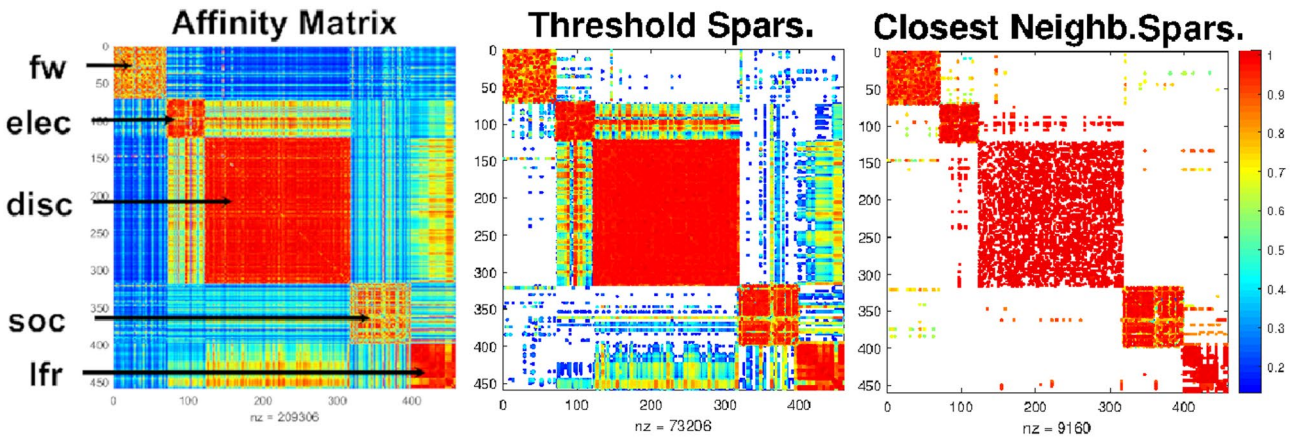


Fig. 10 Affinity matrix of the dataset, and its two sparsifications

Table 5 Confusion matrices of the resulting clustering (columns are clusters, rows are classes) and $F1$-scores.

	1	2	3	4	5		1	2	3	4	5	6	7	8	9	10
fw	*70*					fw										*70*
Elec			6		*46*	Elec			4	2					*46*	
Disc	1		*192*	2		Disc			*52*	*26*	*56*	*59*		1		1
Soc	1	*75*		5		Soc	*40*	*38*					1	1		1
lfr				*60*		lfr							*10*	*50*		
$F1$	9.86	9.62	9.77	9.45	9.39	$F1$	9.81		9.80				9.76		9.39	9.86

Left: Threshold sparsification. Right: knn sparsification, Clusters can be associated with classes, as highlighted by the italic values

versions, where a new version is created each time an author amends the article. An author amending an article can add new material, or modify what is written in the current version. As the text thus modified has been written by another author, it is possible to draw relations between authors from an article history.

Some technical points need to be carefully managed to extract graphs from these relations, and we summarise them in Fig. 5. Here, we process a toy article containing a unique sentence, which evolves through time. First, we extract text from article versions, using the method of Attardi et al. (2013). We process the text to keep only word roots, and we remove non-informative words such as "a", "the", etc. At d_1, A_1 creates the article. At d_2, author A_2 modifies it by correcting A_1's version. However, no edge is drawn because A_2 does not change the lemmatised text. Author A_3 then changes the sentence, and an edge is drawn from A_3 to A_1 because A_3 removes the lemma "eat" (she replaces "eat" by "devour"). We consider that A_3 impacts A_1 instead of A_2 because A_1 is responsible for the term "eat", since she was the first to write it in the sentence. Then, A_4, a "troll", radically modifies the sentence, which creates edges from A_4 to A_1 ("mouse" is removed) and A_3 (removal of "big" and "devour"). A_5 then restores the article by returning it to A_3's version. Finally, A_2 modifies A_5 version. However, if the last modification from A_2 comes more than one week after the restoration done by A_5 ($d_6 - d_5 > 7$ days), we do not consider an edge at d_6, to limit the number of relations with idle authors.

When amending a new version, an author can add a tag to the article to inform readers about its current state. For instance, in French Wikipedia, if the version of an article is well-written and well-sourced, it may be assigned the tag "Article de Qualité" or "Bon Article"—the corresponding English tags being "Featured Article" and "Good article". Henceforth, these articles will be denoted "Quality articles". An article which is not sufficiently sourced, or for which no consensus between authors can be achieved is labelled as "Article Non Neutre", "Désaccord de Neutralité", or "Article en Guerre d'Édition"—i.e. "Biased", "Disputed neutrality" or "Article in edit-war". We will refer to these as "Contested articles". The purpose of the following analysis is to investigate what discriminates a quality article from a contested one. To do this, we extract from a whole article the subgraphs that correspond to subsequent versions which contain a significant proportion of the corresponding tag. This is shown in Fig. 6, where we also provide parameters

(number of versions and proportions of tags). As shown in the figure, we favour time slots which contain tags in the latest versions, as the process that leads to tagging an article as quality or contested also derives from its previous untagged states. Once these time slots have been identified, we extract subgraphs by considering only the edges from versions that fall between time slot intervals.[2] Finally, for our analysis, we filter the networks and only keep those with at least 40 authors. This results in 33 quality networks, and 55 contested networks.

4.2 Analysis

We first verify that our method presented in Algorithm 1 is able to discriminate between quality and contested versions. As parameters, we have used four principal axes in PCA, and three-node graphlets only. Results were less accurate with three- and four-node graphlets as the networks are relatively small. We have kept 20 networks in our training set, and generate ten training/test sets. Results are summarised in Table 4.

We observe that while scores are not as high as those obtained on the simpler problem of detecting fields from networks in Sect. 3.3, we are still able to achieve consistent and accurate results.

Figure 7 shows the embeddings [i.e. outputs of function f in (2.1)] of quality and contested articles. Details about graphlets identifiers on the x-axis are given in le Gorrec et al. (2021)-Sect. III. We see that quality articles always have fewer instances of graphlet 3–12  than graphlet 3–6  , which is not the case for contested articles. We also remark that quality articles are characterised by fewer instances of graphlets 3–14  and 3–74  . That is, quality articles exhibit fewer reciprocal edges than contested articles. This absence of reciprocal edges is confirmed in Fig. 8, where we show that more than 90% of quality articles have a coordinate corresponding to graphlet 3–14 below 0.2 (below 0.1 for graphlet 3–74). On the other hand, the cumulative density of the proportion of graphlet 3–14 increases linearly for contested articles.

Using PCA, interesting patterns appear when projecting the networks onto the first and third principal axes, as shown in Fig. 9. The first principal axis is dominated by a negative coordinate for graphlet 3–6, and four positive coordinates for graphlets 3–12, 3–14, 3–36, and 3–74. Most of the quality articles have a negative coordinate on the first principal axis, which confirms our previous observations that these articles have only a few reciprocal edges. This also confirms that they show a lower proportion of graphlet 3–12 than graphlet 3–6. On the other hand, several contested articles have a positive coordinate for the first principal axis, which confirms the stronger presence of reciprocal edges in these articles. On the third principal axis, most of the quality articles have a negative coordinate. Once again, this means that quality articles show more 3–12 and 3–6 graphlets than graphlets with reciprocal edges. For contested articles, the distribution along this axis indicates that graphlets 3–12 coexist with reciprocal edges in a more homogeneous fashion. We also observe that around the coordinate $(-0.4, 0.1)$, quality and contested articles are indistinguishable. To highlight this phenomenon, we have approximated quality and contested articles with least square best-fit quadratic polynomials. These two polynomials show very different behaviours, except towards the left of the plot around the said coordinate. This region corresponds to networks having a coordinate higher than 0.985 for graphlet 3–6. Such networks have been marked with a black cross in the figure. Since network embeddings have been normalised, such networks are dominated by this graphlet, and our method cannot separate them.

In summary, we have highlighted that we can globally distinguish quality articles from contested ones via our embeddings built on three-node graphlets. A more in-depth analysis identifies the main features that permit discrimination among these articles: the absence of reciprocal edges within quality articles, and a tendency to contain fewer instances of graphlet 3–12 than 3–6. These results can be explained by the presence of authors having an authority status. Such authors frequently correct other authors' contributions (as highlighted by the high presence of graphlet 3–6), and their contributions/corrections are rarely discussed, as shown by the rarity of graphlets with reciprocal edges. Furthermore, the fact that these articles are also characterised by a higher proportion of graphlet 3–6 than graphlet 3–12 highlights the fact that authors with authority are also key contributors in terms of material removal. On the other hand, several contested articles are well separated from quality articles because of a high presence of graphlets with reciprocal edges (graphlets 3–14/3–74), and/or a lower proportion of graphlet 3–6 than graphlet 3–12. The former highlights the lack of consensus within these articles. The latter expresses the absence of authors with unchallenged authority.

Finally, the fact that some contested articles show a dominant proportion of graphlets 3–6 may come from the arrival of "troll" authors, whose contributions consist in the removal of a great part of the article. In such cases, the troll points to all the authors who have contributed to the article.

[2] For instance, extracting the subgraph corresponding to time slot (d_4, d_6) in the graph Fig. 5 means that we remove the edge from A_3 to A_1, since this edge appears at date $d_3 < d_4$.

5 Conclusion

5.1 Towards unsupervised clustering

In this paper, we have presented two supervised methods for network embeddings. However, we can also adapt Algorithm 3 to a non-supervised algorithm, and we predict that the nine graphlets with the highest average γ-score (see Fig. 4) should also be able to discriminate other classes of networks.

To support this claim, we have applied the embedding given in (2.7), with $\Gamma_3 = \{3\text{–}12,3\text{–}6,3\text{–}74,3\text{–}14,3\text{–}36,3\text{–}78\}$ and $\Gamma_4 = \{4\text{–}2184,4\text{–}76,4\text{–}14\}$ to the whole dataset from Sect. 3.1, supplemented with 60 directed networks built using the LFR graph generator (Lancichinetti and Fortunato 2009). This generator builds random modular networks with features such as heterogeneous distributions of node degrees and community sizes described by power laws. A comprehensive description of our use of the LFR generator is provided in le Gorrec et al. (2021)-Sect. VII-A.

To uncover the network clusters, we chose the unsupervised clustering algorithm from le Gorrec et al. (2020). This algorithm needs the dataset to be represented by an affinity matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$. We detail the construction of $\mathbf{A}$ in le Gorrec et al. (2021)-Sect. VII-B. We have applied this algorithm on two sparsifications of the affinity matrix. The first is derived from a thresholding procedure (all entries below 0.5 have been removed). The second is derived from a k-nearest-neighbour (knn) procedure. (Only the 20 highest values have been kept in each row.) The affinity matrix and both sparsifications are shown in Fig. 10.

The unsupervised clustering method applied to both sparsified matrices gives consistent results. For the first, it finds the five targeted clusters with only a handful of errors of assignation. For the second, it finds ten clusters, but these clusters can be merged to recover the actual classes. The confusion matrices are shown in Table 5, as well as the $F1$-score.

These preliminary results strongly suggest that the nine graphlets that discriminate between the four classes of networks in our numerical experiments from Sect. 3.3 are also able to discriminate other classes of networks. However, this hypothesis needs to be tested on other classes of real-world networks to get a better feeling for the extent to which these graphlets are sufficient to discriminate between classes.

5.2 Concluding remarks

We have proposed two supervised procedures for network embedding based on k-node graphlet decomposition. These procedures have been shown to provide accurate results in a downstream classification task. The whole process, despite its conceptual simplicity, is competitive with state-of-the-art methods. In contrast to most existing methods for classifying or comparing networks based on graphlet decomposition (Tu et al. 2019; Milo et al. 2004), we avoid an evaluation of the graphlet distribution on random models. We have used the fully supervised version of our method to discriminate between networks of authors' relationships within quality and contested Wikipedia articles, on an original database. Our embeddings and an in-depth analysis of the principal axes have highlighted typical users behaviours in these articles. Of particular note is that our feature selection method selects only a few graphlets (nine out of a possible 212), which means we can avoid computing all graphlets for the networks in the test set. Our bespoke feature selection has properties similar to those of PCA, and we have shown that this procedure outperforms a well-established selection feature method.

We have also provided some early-stage empirical experiments that suggest that the graphlets selected by our feature selection procedure in the numerical experiments are actually able to discriminate other kinds of networks apart from those classes on which these graphlets have been learnt. We infer that these graphlets are sufficient to discriminate among wider classes of networks, and should enable one to derive unsupervised methods for network embedding based on just these few graphlets. We remark that the graphlets we used to highlight users' behaviour within the Wikipedia database all appear within the nine graphlets selected by our technique.

Our future work will be to extend these experiments to understand the full potential (as well as the limitations) of these graphlets to characterise fields of networks, in order to design unsupervised techniques for learning network embeddings.

Finally, our method can be applied to other applications of network classification, and to undirected networks as well. For instance, in le Gorrec and Knight (2020), we showed it was able to accurately classify the *MUTAG* dataset (Debnath et al. 1991).

Acknowledgements We would like to thank Kun Tu from the University of Massachusetts Amherst, for providing us with the code of `gl2vec` and answering our questions. We also thank the reviewers from SNAM, ASONAM and FAB for their constructive feedback which has resulted in a much improved manuscript.

Author Contributions LlG conducted the data collection and analysis of the benchmark from Sect. 3.1, and the designing and implementation of all the presented methods, under the supervision of PK; AC conducted the data collection and analysis of the Sect. 4; LlG and PK wrote the article.

Funding This project was supported by the Royal Academy of Engineering and the Office of the Chief Science Advisor for National Security under the UK Intelligence Community Postdoctoral Fellowship Programme.

Availability of code and data The materials used in this study are available at https://github.com/luleg/DiscriminantMotifs. The network repository from (Freeman 2020) is no longer accessible.

Declaration

Conflict of interest The authors declare that they have no conflict of interest.

References

Abdi H, Williams LJ (2010) Principal component analysis. Wiley Interdiscip Rev Comput Stat 2(4):433–459

Ahmed A, Shervashidze N, Narayanamurthy S, Josifovski V, Smola AJ (2013) Distributed large-scale natural graph factorization. In: International conference on world wide web, pp 37–48

Alon U (2007) Network motifs: theory and experimental approaches. Nat Rev Genet 8(6):450–461

Annamalai N, Mahinthan C, Rajasekar V, Lihui C, Yang L, Shantanu J (2017) graph2vec: learning distributed representations of graphs. In: International workshop on mining and learning with graphs

Artzy-Randrup Y, Fleishman S, Ben-Tal N, Stone L (2004) Comment on network motifs: simple building blocks of complex networks and superfamilies of evolved and designed networks. Science 305(5687):1107–1107

Asher N, Lascarides A (2003) Logics of Conversation. Cambridge University Press, Cambridge

Asher N, Hunter J, Morey M, Benamara F, Afantenos S (2016) Discourse structure and dialogue acts in multiparty dialogue: the STAC corpus. In: LREC 2721–2727

Attardi G, Fuschetto A, Souza L, Caicedo JM, Pereira H, Gevatter S-A (2013) https://gist.github.com/baojie/5294784

Batagelj V, Mrvar A (2006) Pajek datasets. http://vlado.fmf.uni-lj.si/pub/networks/data/. Accessed Mar 2020

Benson A, Gleich D, Leskovec J (2016) Higher-order organization of complex networks. Science 353(6295):163–166

Borgwardt KM, Kriegel H-P (2005) Shortest-path kernels on graphs. In: IEEE international conference on data mining, p 8

Borra E, Weltevrede E, Ciuccarelli P, Kaltenbrunner A, Laniado D, Magni G, Mauri M, Rogers R, Venturini T et al (2014) Contropedia-the analysis and visualization of controversies in wikipedia articles. In: OpenSym 34-1

Breiman L (2001) Random forests. Mach Learn 45(1):5–32

Brglez F, Bryan D, Kozminski K (1989) Combinational profiles of sequential benchmark circuits. In: IEEE international symposium on circuits and systems, pp 1929–1934

Cancho IRF, Janssen C, Solé RV (2001) Topology of technology graphs: small world patterns in electronic circuits. Phys Rev E 64(4):046119

Cao S, Lu W, Xu Q (2016) Deep neural networks for learning graph representations. In: AAAI conference on artificial intelligence, pp 1145–1152

Chiba N, Nishizeki T (1985) Arboricity and subgraph listing algorithms. SIAM J Comput 14:210–223

Clauset A, Tucker E, Sainz M (2016) The Colorado index of complex networks. https://icon.colorado.edu/. Accessed Feb 2020

Corno F, Reorda MS, Squillero G (2000) RT-level ITC 99 benchmarks and first ATPG results. IEEE Des Test comput 17(3):44–53

Debnath AK, Lopez de Compadre RL, Debnath G, Shusterman AJ, Hansch C (1991) Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. J Med Chem 34(2):786–797

Defferrard M, Bresson X, Vandergheynst P (2016) Convolutional neural networks on graphs with fast localized spectral filtering. In: Advances in neural information processing systems, pp 3844–3852

Dunne J, Lafferty K, Dobson A, Hechinger R, Kuris A, Martinez N, McLaughlin EA (2013) Parasites affect food web structure primarily through increased diversity and complexity. PLoS Biol 11(6):e1001579

Estrada E, Knight P (2015) A first course in network theory. Oxford University Press, Oxford

Felmlee D, McMillan C, Towsley D, Whitaker R (2018) Social network motifs: a comparison of building blocks across multiple social networks. In: Annual meetings of the ASA

Freeman LC (2020) http://moreno.ss.uci.edu/data.html. Accessed Feb 2020. Out-of-date

Gargiulo F, Caen A, Lambiotte R, Carletti T (2016) The classical origin of modern mathematics. EPJ Data Sci 5:26

Gärtner T, Flach P, Wrobel S (2003) On graph kernels: hardness results and efficient alternatives. In: Learning theory and kernel machines. Springer, pp 129–143

Grover A, Leskovec J (2016) Node2vec: scalable feature learning for networks. In: ACM SIGKDD international conference on knowledge discovery and data mining, pp 855–864

Hamilton WL, Ying R, Leskovec J (2017a) Inductive representation learning on large graphs. In: Advances in neural information processing systems

Hamilton WL, Ying R, Leskovec J (2017b) Representation learning on graphs: methods and applications. In: IEEE data engineering bulletin

He B, Tan K (2016) Understanding transcriptional regulatory networks using computational models. Curr Opin Genet Dev 37:101–108

Head A, Eisenberg M (2010) How todays college students use wikipedia for course-related research. First Monday 15(3)

Jenihhin M (2020) http://pld.ttu.ee/~maksim/benchmarks/iscas89/bench/. Accessed Feb 2020

Khakabimamaghani S, Sharafuddin I, Dichter N, Koch I, Masoudi-Nejad A (2013) Quatexelero: an accelerated exact network motif detection algorithm. PLoS ONE 8(7):1–15

Kipf TN, Welling M (2017) Semi-supervised classification with graph convolutional networks. In: International conference on learning (**Representations**)

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

Lascarides A, Asher N (2007) Segmented discourse representation theory: dynamic semantics with discourse structure. Springer, pp 87–124

le Gorrec L, Knight PA (2020) A simple embedding for classifying networks with a few graphlets. In: 2020 IEEE/ACM international conference on advances in social networks analysis and mining (ASONAM). IEEE, pp 635–642

le Gorrec L, Mouysset S, Duff IS, Knight PA, Ruiz D (2020) Uncovering hidden block structure for clustering. In: Machine learning and knowledge discovery in databases, pp 140–155
le Gorrec L, Knight PA, Caen A (2021) Supplementary material for embeddings of networks using small-size graphlets: comparisons and analysis. http://github.com/luleg/DiscriminantMotifs
Lin MC, Soulignac FJ, Szwarcfiter JL (2012) Arboricity, h-index, and dynamic algorithms. Theoret Comput Sci 426–427:75–90
Meira LA, Máximo VR, Fazenda ÁL, Da Conceição AF (2014) Acc-motif: accelerated network motif detection. IEEE/ACM Trans Comput Biol Bioinf 11(5):853–862
Mesgar M, Strube M (2015) Graph-based coherence modeling for assessing readability. In: Joint conference on lexical and computational semantics, pp 309–318
Milo R, Shen-Orr S, Itzkovitz S, Kashtan N, Chklovskii D, Alon U (2002) Network motifs: simple building blocks of complex networks. Science 298(5594):824–827
Milo R, Kashtan N, Itzkovitz S, Newman ME, Alon U (2003) On the uniform generation of random graphs with prescribed degree sequences
Milo R, Itzkovitz S, Kashtan N, Levitt R, Shen-Orr S, Ayzenshtat I, Sheffer M, Alon U (2004) Superfamilies of evolved and designed networks. Science 303(5663):1538–1542
Niepert M, Ahmed M, Kutzkov K (2016) Learning convolutional neural networks for graphs. In: International conference on machine learning, pp 2014–2023
Oboler A, Steinberg G, Stern R (2010) The framing of political NGOs in wikipedia through criticism elimination. J Inf Technol Polit 7(4):284–299
Ou M, Cui P, Pei J, Zhang Z, Zhu W (2016) Asymmetric transitivity preserving graph embedding. In: ACM SIGKDD international conference on knowledge discovery and data mining, pp 1105–1114
Peng H, Li J, Gong Q, Wang S, Ning Y, Yu PS (2018) Graph convolutional neural networks via motif-based attention. arXiv:1811.08270
Perozzi B, Al-Rfou R, Skiena S (2014) Deepwalk: Online learning of social representations. In: ACM SIGKDD international conference on knowledge discovery and data mining, pp 701–710
Ribeiro P, Silva F (2014) G-tries: a data structure for storing and finding subgraphs. Data Min Knowl Discov 28:337–377
Ribeiro LF, Saverese PH, Figueiredo DR (2017) struc2vec: learning node representations from structural identity. In: ACM SIGKDD international conference on knowledge discovery and data mining, pp 385–394
Ribeiro P, Paredes P, Silva ME, Aparicio D, Silva F (2021) A survey on subgraph counting: concepts, algorithms, and applications to network motifs and graphlets. ACM Comput Surv (CSUR) 54(2):1–36
Rozemberczki B, Kiss O, Sarkar R (2020) Karate club: an API oriented open-source python framework for unsupervised learning on graphs. In: ACM international on conference on information and knowledge management
Sapiezynski P, Stopczynski A, David Dreyer L, Lehmann S (2019) Interaction data from the Copenhagen networks study. Sci Data 6(1):1–10
Schlichtkrull M, Kipf TN, Bloem P, Van Den Berg R, Titov I, Welling M (2018) Modeling relational data with graph convolutional networks. In: European semantic web conference. Springer, pp 593–607
Shervashidze N, Vishwanathan S, Petri T, Mehlhorn K, Borgwardt K (2009) Efficient graphlet kernels for large graph comparison. In: International conference on artificial intelligence and statistics, pp 488–495
Shervashidze N, Schweitzer P, Van Leeuwen EJ, Mehlhorn K, Borgwardt KM (2011) Weisfeiler–lehman graph kernels. J Mach Learn Res 12(9):2539–2561
Stouffer DB, Camacho J, Jiang W, Nunes Amaral LA (2007) Evidence for the existence of a robust pattern of prey selection in food webs. Proc R Soc B Biol Sci 274(1621):1931–1940
Theodoridis s (2015) Machine learning: a Bayesian and optimization perspective. Academic Press, London
Tran NTL, Mohan S, Xu Z, Huang C-H (2015) Current innovations and future challenges of network motif detection. Brief Bioinform 16(3):497–525
Tu K (2018) https://github.com/kuntu/JGraphlet-JMotif. Accessed June 2020
Tu K, Li J, Towsley D, Braines D, Turner LD (2019) Gl2vec: learning feature representation using graphlets for directed networks. In: IEEE/ACM international conference on advances in social networks analysis and mining, pp 216–221
Turk M, Pentland A (1991) Eigenfaces for recognition. J Cogn Neurosci 3(1):71–86
Veličković P, Cucurull G, Casanova A, Romero A, Liò P, Bengio Y (2018) Graph attention networks. In: International conference on learning (**Representations**)
Vishwanathan SVN, Schraudolph NN, Kondor R, Borgwardt KM (2010) Graph kernels. J Mach Learn Res 11:1201–1242
Viswanath B, Mislove A, Cha M, Gummadi KP (2009) On the evolution of user interaction in facebook. In: ACM workshop on online social networks, pp 37–42
Xu K, Hu W, Leskovec J, Jegelka S (2019) How powerful are graph neural networks? In: International conference on learning (**Representations**)
Yasseri T, Sumi R, Rung A, Kornai A, Kertész J (2012) Dynamics of conflicts in wikipedia. PLoS ONE 7(6):1–12