Leveraging semantic for community mining in multilayer networks

Mohamed Benabdelkrim, Céline Robardet, Jean Savinien

Leveraging semantic for community mining in multilayer networks

Mohamed Benabdelkrim[†,‡,*], Céline Robardet[†], Jean Savinien[‡]
[†] INSA Lyon
[‡] emlyon business school

Abstract

Community detection in multilayer networks aims to identify groups of well-connected nodes across multiple layers. While existing methods have been developed to deal with large graphs with few layers (typically less than 10), many real-world datasets are structured by transitive relationships that give rise to networks with thousands of extremely dense layers (e.g. co-citation networks, IMDb actor graphs, co-reference information network, social networks). In addition, in these datasets the layers are often associated with textual summaries which provide important and hitherto unexploited information on the nature of the relation encoded by the layer. In this paper, we propose a new method which exploits the text associated with the layers in order to identify communities grouping together nodes connected through several semantically close layers. The method consists in embedding layer textual information in an Euclidean space, and to use it to group together, in the same community, nodes belonging to semantically close layers. To that end, we develop a pattern mining approach that extracts communities from numerical data. This approach, which mixes both symbolic and numeric techniques, is particularly well suited to identify communities in multilayer graphs. Indeed, we show that it obtains more diverse and better quality communities than those obtained by state-of-the-art competitors on datasets where ground truth is known. We also show that taking into account the semantic information improves the quality of the communities.

Keywords: Multilayer networks, detection of semantic communities, pattern mining, Social networks analysis.

1. Introduction

The search for communities in graphs has given rise to numerous research works [1]. Indeed, communities, or groups of nodes similar by their strong connectivity, make possible to structure a graph into fairly independent components that facilitate its analysis. During the last decade, driven by the study of complex systems coming from various application domains (biology, computer science, engineering, economics, politics, sociology, etc.), graphs representing real-world systems have been enriched with additional information. First, attributes have been associated to nodes and community detection methods have been extended to the search of dense and homogeneous communities in terms of attribute values [2, 3]. Networks have also been enriched with additional information on the connections between nodes, as multiple bonds between any two nodes may exist, reflecting different kinds of relationships. Such multilayer networks are made of multiple interdependent graphs, where each graph stands for one aspect of the relationships. Several methods have been proposed to address community detection problem in multilayer networks [4, 5], as well as for multiplex networks [6], where all the layers share an identical set of nodes.

However, an aspect very widespread in real-world graphs has been little considered so far: The fact that many relationships between nodes are *transitive*. For example, if we consider family networks, the relationships are transitive that is to say if persons a and b are relatives and persons b and c are also parents, then a and c are from the same family as

[*]Mohamed.Benabdelkrim@insa-lyon.fr

well. It is the case for many other network types, e.,g. co-citation or co-purchase networks. If we consider that each layer of the graph corresponds to a group of nodes completely connected by the transitive relation, communities in such a multilayer network correspond to new groups of nodes which appear simultaneously in different layers.

While trying to analyze these real-world networks, we noticed that each layer is generally associated to textual unstructured data that characterize and contextualize the interaction. Several examples are given in [7] where the multilayer networks result from the combination of multiple information sources. For example, when a co-author and a co-citation networks are combined, each layer is also associated with additional information such as the conference venues where the paper was published, its keywords or abstract. Another example concerns social networks where layers represent different types of social relationships like "friends" or "colleagues". The additional information associated with the layers is then the messages exchanged between group members. Three real-world examples are considered in our experiments which illustrates the numerous real contexts where one has a multilayer graph with transitive relationships and where semantic information is associated to layers. This information is valuable and can be used when building communities to group nodes that are often related to other nodes in layers that have similar characteristics. Taking into account the semantic similarity between the layers makes it possible to create communities which group together nodes appearing in semantically close contexts, for example authors who have co-authored articles on analogous subjects.

In the following, we present the method MIMETIC (*coMmunities In Multilayer networks using sEmanTIC*) to detect communities of nodes across semantically similar layers. Each layer is supposed to be composed of a clique of nodes. As said before and illustrated in the experiments, this structure is natural in many real contexts. If however the analyzed data did not lend itself easily to this modeling, we can follow the way paved by [8], and compute semantic communities in each layer using for example [3], each obtained communities being a layer for our method. MIMETIC uses the semantic to guide the community construction and also to provide semantic information on the discovered communities. Taking semantics into account makes the detection of communities more robust. Indeed, building the communities on the sole topology of the network, information that can be noisy, makes the communities sensitive to epiphenomenon. Considering the semantic associated to layers can attenuate this phenomenon and increase the community quality, as we show in our experiments. MIMETIC consists in computing embedding in an Euclidean space of the textual information associated to each layer. We use a similarity measure between vectors as a proxy for the membership relation of nodes to layers. This results in a numerical relationship on which we compute communities using top-k diversified interval patterns. A community is then a set of nodes which have a high membership value for the same set of layers of the graph. As it is verified in the experiments, the obtained communities are the most important of the network and are obtained in a time of one or two orders of magnitude less than that of the competitors. We also show that the textual information associated with the layers can complement the network structure, leading to more accurate communities with respect to ground truth, less prone to noise effects.

The paper presents four contribution: (1) the formalization of the problem of detecting communities in transitive multilayer networks associated with textual data, problems which are very widespread in practice; (2) the use of efficient NLP techniques to evaluate the similarity between two layers in a discriminating way; (3) the proposal of an original method which inherits from the mining of closed interval patterns to identify k diverse communities; (4) experiments which show the capacities of the proposed method to identify communities on three large very different datasets and with better results than the existing methods.

The rest of the paper is structured as follows: Section 2 discusses related work. Section 3 presents the method. Section 4 describes our experiments and explains the results. Section 5 concludes and gives directions for future work.

2. Related Work

Communities in graphs denote groups of nodes that are more strongly or frequently connected among themselves than with the remainders. There are many methods to decompose a single layer network into communities, and we refer to [1] as a review of this field. The identification of communities in a graph proceeds from a Cartesian approach consisting in breaking down a graph into a set of smaller-sized modules to facilitate analysis. However, many complex systems are composed of multiple layered networks, where each layer represents one of many possible types of interactions. In order to determine communities in this type of graph, we can distinguish between local methods, which build a single community around a few seed nodes (e.g., [9]), from global methods which build communities on the whole network. Among the latter, [6, 10] classify the methods into three main categories. First, the flattening methods collapse the layers' information into a simple weighted graph and process it with usual single network community detection methods. Such approaches have the major default to lose information on the layers from which the aggregated links originate [11]. Second, aggregation methods discover communities in each layer independently and then merge them with an aggregation procedure. EMCD [12], that relies on the clustering ensemble paradigm, and ABACUS [8], based on frequent itemset mining, are examples of this category. Third, direct methods compute communities at once on the multilayer network. Mucha et al. [13] extend the modularity function to multilayer graphs, by coupling communities in neighboring layers. Edler et al. [14] propose the information-theoretic and flow-based community detection method Infomap. They convert the community detection task into solving a coding problem following the minimum-description-length principle. Using this compression-based framework, they are able to compare how much different representations compress modular flows using random walks. Communities in multilayer networks can also be identified using graph embedding approaches. One of them, MNE (Multiplex Network Embedding) [15], generates a layer-wise embedding for every node and layer, which combines an embedding shared across all the layers and that describes the node globally, with a layer-wise embedding. The obtained embedding represents the information of the multilayer graph into an Euclidean space that preserves proximity information. Appropriate distance measures between embedded vectors make possible to identify communities.

A very important aspect of multilayer graphs has so far been little taken into account by community detection methods: the semantics associated with each layer. Indeed, each relation, which corresponds to a layer of the network, has particular semantics making certain layers close or on the contrary more distant from others. As explained by Bothorel et al. [11], the analysis of real-world multilayer graph is made difficult by the fact that a large number of semantically different layers are considered all-together. With their MiMag method, Boden et al. [7] propose a first contribution to take into account attributes associated with layers when building communities. Their communities satisfy both aspects of structural density and edge label similarity. To that end, communities are enforced to be γ-quasi-cliques whose edge labels vary at most by a certain threshold w. Redundancy is avoided by selecting only the most interesting, non-redundant subgraphs. However, as we show in the experiments, the communities extracted by MiMag are not of very good quality and do not allow the ground truth to be identified when it is known. Taking into account a simple numerical value associated with the layers is not sufficiently informative.

This is why we propose in this article a method based on the semantic analysis of texts describing the layers to group those that have similar purposes.

3. Detection of communities in attributed multilayer graph

Multilayer graphs encode several relationships between a set of nodes. Different layers may capture similar phenomenon and reinforce the relationships observed between nodes. It is known that the links observed in a graph are prone to noise and this problem can be mitigated by crossing several sources of information in order to have a set of concordant hypotheses on the membership of a subset of nodes to the same community. This is what we propose to do using the concept of *attributed multilayer graph*.

Definition 1 (Attributed Multilayer Graph)**.** *An attributed multilayer graph $\mathcal{G}$ is a set of layers $L = \{1, \ldots, d\}$, each associated (1) to a graph $G_i = (V_i, E_i)$ with a transitive relationship $E_i = V_i \times V_i$ between nodes, and (2) to a text, denoted* Text_i*, that describes the content of the information encoded by the layer.*

This graph model is widespread and can be used in various contexts. For example, let us consider a co-citation network, where nodes stand for articles. Two nodes are linked by an edge in the layer i if the corresponding articles are cited by a same article that coincides with layer i. The text describing layer i can be the title or the abstract of the paper. Another example concerns social networks such as Twitter where users can be grouped into lists. Each list can be modeled as a layer, and two nodes, that stand for two users, are linked in the graph i if they belong to the list i. The title and description of the list are used as attribute of i. Thus, each layer corresponds to a source of information on the relationship between nodes. To exploit this type of graph, we could derive a weighted graph in which the weight of the edges would correspond to the number of layers in which the adjacent nodes are connected. However, as discussed by [11], this approach can hide relevant information, such as the fact that certain layers of the graph are more important than others in defining a community, by scrambling the semantic information associated with each layer. This is why we propose an approach which extracts the communities directly on the multilayer network by exploiting the semantic similarity of the layers.

Definition 2 (Semantic similarity between layers)**.** *Let* Sim *be a similarity measure between two layer descriptions:* $\forall i, j \in L$, $\text{Sim}(\text{Text}(i), \text{Text}(j)) \in [0, 1]$ *and* $\text{Sim}(\text{Text}(i), \text{Text}(i)) = 1$.

The way the similarity is computed between layers is detailed below. Now, to form communities in an attributed multilayer graph, we propose to group nodes that are connected in the same layer or in semantically close layers. This process is based on a layer membership function defined as follows:

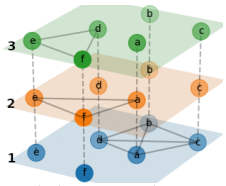
Definition 3 (Membership function)**.** *Given an attributed multilayer graph $\mathcal{G} = \{(G_i, \text{Text}_i) \mid i \in L\}$, the membership function* $\text{M}(v, i)$ *of a node v ($v \in \bigcup_{k=1}^{d} V_k$) to a layer i is defined by:*

$$\begin{aligned} \text{M}(v, i) &= 1 \text{ if } v \in V_i, \\ \text{M}(v, i) &= \max_{k \mid v \in V_k} \text{Sim}(\text{Text}_i, \text{Text}_k), \text{ otherwise.} \end{aligned}$$

Example 1. *Let us consider the attributed 3-layers graph in Figure 1:* $\mathcal{G} = \{(G_1, \text{Text}_1), (G_2, \text{Text}_2), (G_3, \text{Text}_3)\}$ *such that* $\text{Sim}(\text{Text}_1, \text{Text}_2) = 0.2$, $\text{Sim}(\text{Text}_1, \text{Text}_3) = 0.9$ *and* $\text{Sim}(\text{Text}_2, \text{Text}_3) = 0.7$. *Considering node a such that $a \in G_1$, $a \in G_2$, and $a \notin G_3$, we have*

$$M(a, 3) = max(\text{Sim}(\text{Text}_1, \text{Text}_3), \text{Sim}(\text{Text}_2, \text{Text}_3)) = 0.9$$

Figure 1 provides the derived numerical dataset and two examples of communities.



Multilayer graph

	Layer 1	Layer 2	Layer 3
a	1.0	1.0	0.9
b	1.0	0.0	0.9
c	1.0	0.0	0.9
d	1.0	0.7	1.0
e	0.9	1.0	1.0
f	0.9	1.0	1.0

Numerical dataset

Nodes	Layers and interval values
a b c d	(Layer 1)[1.0,1.0] (Layer 3)[0.9,1.0]
d e f	(Layer 1)[0.9,1.0] (Layer 2)[0.7,1.0] (Layer 3)[1.0,1.0]

Communities

Figure 1. Example of an attributed 3-layer graph, its corresponding numerical dataset, and two examples of communities ($\text{Sim}(\text{Text}_1, \text{Text}_2) = 0.2$, $\text{Sim}(\text{Text}_1, \text{Text}_3) = 0.9$ and $\text{Sim}(\text{Text}_2, \text{Text}_3) = 0.7$).

3.1. **Computing similarities between textual attributes**

To take advantage of the textual information associated with each layer, we propose to use recent machine learning techniques. The goal is to create a text embedding that synthesizes the semantics of the text into a single vector. Unsupervised techniques have been proposed to perform this type of task, such as Word2Vec [16]. But the limits of these approaches are that they do not take into account the context of the words when embedding. More recent methods consider the context of the word so that the vector of a polysemous word is closer to words with a similar meaning depending on the context. Learning such models is very resource-intensive, and pre-trained models are now available, offering significant improvements over vectors learned from scratch. Indeed, these models allow even low resource tasks to benefit from deep learning architectures allowing the same pre-trained model to successfully tackle a wide range of NLP tasks. The advantage of these approaches is that the parameters are not learned from random initialization, but improved from the pre-trained models to be more suited to the considered problem.

We use the deep learning framework for natural language processing BERT[1] (Bidirectional Encoder Representations from Transformers) [17], already pre-trained on BooksCorpus (800M words) and English Wikipedia. We fine-tune this model on the texts describing the layers. This makes the model more specific and increases the dispersion of the generated embedding in the associated vector space. The similarities between the BERT embeddings of layer's attributes are then obtained by the cosine distance and are used as Sim measure in the membership function.

3.2. **Computing communities in numerical dataset with interval patterns**

We propose to identify communities in multilayer graphs based on the membership function. Our objective is to identify groups of nodes that share common high membership values to one same subset of layers. To this end, we build a numerical dataset that contains values of membership $M(v,i)$ of nodes $v \in V$ to layers $i \in L$. These membership values are based on similarity measures between BERT embeddings of the layers' attributes. Finding communities in the multilayer network is then equivalent to extracting closed interval patterns from the numerical dataset [18]. This approach defines communities as closed patterns that have both an intention specifying intervals of membership values for the layers; and an extension which is the set of nodes that have membership values inside the intervals for all the layers. Thereby, the communities are well characterized by the layer intervals and can be interpreted as we will see in the experiments.

Definition 4 (Interval patterns)**.** *Let us consider the numerical dataset crossing the nodes of* $V = \bigcup_{i=1}^{d} V_i$ *with the layers of* L *and whose entries are the values* $\text{M}(v,i)$ *with* $v \in V$ *and* $i \in L$

[1]bert-base-uncased: https://huggingface.co/transformers/pretrained_models.html

(sse Definition 3). An interval pattern I is a vector of d intervals $\langle [a_i, b_i] \rangle_{i \in \{1,\dots,d\}}$. A node $v \in V$ belongs to the support of I iff $\forall i \in \{1,\dots,d\}, \mathrm{M}(v,i) \in [a_i, b_i]$. The support of I is denoted $\phi(I)$.

As the value domain of the membership function is $[0,1]$, the support of the interval pattern $I = \bigtimes_{i=1}^{d} [0,1]$ is the whole set of nodes V. There may exist several interval patterns with the same support. With the following Galois connection, we restrict interval patterns to closed ones without loss of information.

Definition 5 (Closed interval patterns)**.** *There is a Galois connection on the interval pattern language, defined by $\phi(I) = \{v \in V \mid \mathrm{M}(v,i) \in [a_i, b_i], \forall [a_i, b_i] \in I\}$ and, for $N \subseteq V$, $\psi(N) = \langle [a_i, b_i] \rangle$ with $a_i = \min_{v \in N} \mathrm{M}(v,i)$ and $b_i = \max_{v \in N} \mathrm{M}(v,i)$. A closed interval pattern satisfies $\psi(\phi(I)) = I$.*

The number of potential closed interval patterns in a numerical dataset is huge. It is necessary to define a quality measure to only keep patterns that correspond to interesting communities in the multilayer network. Recalling that the patterns we are looking for are groups of nodes with high membership values on a subset of layers, we only consider half interval patterns with $b_i = 1$ for all intervals. Among the intervals, the ones that bring some information are those with $a_i \neq 0$. Indeed, such intervals makes it possible to identify a community. So, we propose to evaluate the quality of a closed interval pattern as the sum of the membership values of the nodes in its support on the intervals with $a_i \neq 0$.

Definition 6 (Top-k communities with respect to measure Q)**.** *A community is made of nodes having a high membership value on the intervals that are restricted on the left. We can measure this thanks to the function Q defined as:*

$$Q(I) = \sum_{i=1,\, a_i > 0}^{d} \sum_{v \in \phi(I)} \mathrm{M}(v,i).$$

The top-k communities are the k closed interval patterns with maximal values on Q.

In our view, communities have also to be different from one another. We can measure the similarity between two communities by the proportion of their common nodes:

$$Jaccard(I,J) = \frac{\phi(I) \cap \phi(J)}{\phi(I) \cup \phi(J)}.$$

Two communities are diversified if they have a proportion of common nodes that is lower than a threshold δ. Thus, we compute a set of k communities, each containing at least α nodes. The communities have also to be diversified two by two, and if a pattern is excluded from the set because it is similar to another one, then we preserve the one of greatest value on Q. This is formalized in the following definition.

Definition 7 (Top-k diversified communities)**.** *Let $\mathcal{S}$ be the set of closed interval patterns. Given three parameters, σ, δ and k, we compute the set $\mathcal{K}$ of top-k diversified communities such that (1) $\mathcal{K}$ is of size k with $\mathcal{K} \subseteq \mathcal{S}$; (2) $\forall I \in \mathcal{K}$, $|\phi(I)| \geq \sigma$, (3) $\forall I, J \in \mathcal{K}$, $Jaccard(I,J) < \delta$ and (4) $\forall I \in \mathcal{K}$, if $\exists J \in \mathcal{S}$ such that $Jaccard(I,J) \geq \delta$, then $Q(I) \geq Q(J)$.*

An efficient algorithm to compute closed interval patterns has been proposed in [18]. It avoids to generate several times the same closed interval pattern by using a canonicity test according to lectic order $<$ on the set L of layers: if a pattern I has been generated by a change on attribute i, we consider that it is canonically generated iff I and $\psi(\phi(I))$ do not differ on any attribute h such that $h < i$. The fact that $\psi(\phi(I))$ differ from I on at least one attribute $h \leq index$ is denoted $Closed \prec_{index} I$ in Algorithm 1 and leads to stop the recursion (see line 2). The recursion is also stopped when the support of the pattern is below σ, as none of the patterns generated by the following recursive calls can have a support higher than the threshold.

Algorithm 1: MIMETIC(I, σ, k, δ, index, $\min_Q$, $\mathcal{K}$)

Input: I is an interval pattern generated at the previous step with a minimal change on layer numbered $index$. σ, k and δ are fixed thresholds, $\min_Q$ is a dynamic threshold on Q, and $\mathcal{K}$ is the current set of top-k diversified communities.

Output: $\mathcal{K}$ the set of top-k diversified communities.

```
1  Closed ← ψ(φ(I))
2  if (|φ(I)| < σ) or (Closed ≺_index I) or (UBQ(I) < min_Q) then
3  |  return
4  TopKDiv(K, Closed, k, δ, min_Q)
5  for i = index to d do
6  |  [a, b] ← Closed[i]
7  |  if a = b then
8  |  |  Continue
9  |  c ← nextValue(i,a)
10 |  PatL ← [1...i−1][c, b][i+1...d]
11 |  MIMETIC(PatL, σ, k, δ, i, min_Q, K)
```

We extend this algorithm by adding another condition that may stop the recursion: the fact that an upper bound of Q, noted UBQ, is less than $\min_Q$, an internal threshold that is dynamically updated by the algorithm when computing the top-k communities. Let us define this upper bound UBQ. During the enumeration process of Algorithm 1, the interval pattern I with $[a_i, b_i]$ as i^{th} interval is specialized in an interval pattern J with i^{th} interval $[c_i, d_i]$ such that $a_i < c_i$ and $b_i > d_i$. For all such J, we have:

$$\begin{aligned} Q(J) &\leq \sum_{i=1,\, a_i>0}^{index} \sum_{v\in\phi(J)} \mathrm{M}(v,i) + \sum_{i=index+1}^{d} \sum_{v\in\phi(J)} \mathrm{M}(v,i) \\ &\leq \sum_{i=1,\, a_i>0}^{index} \sum_{v\in\phi(I)} \mathrm{M}(v,i) + \sum_{i=index+1}^{d} \sum_{v\in\phi(I)} \mathrm{M}(v,i) = UBQ(I). \end{aligned} \tag{3.1}$$

Indeed, for the intervals that have not been enumerated so far, we can sum all memberships values. Furthermore, as $\phi(J) \subseteq \phi(I)$, we can also upper bound $Q(J)$ by taking the sum over the vertices in the support of pattern I.

Thus, in line 4 of Algorithm 1, *Closed* is a candidate community and the function TopKDiv ensures that the set of returned communities contains the top k diversified communities (see Algorithm 2). In lines 5 to 11, it generates recursively the next candidate patterns by considering the next interval (line 6). This new interval is a specialization of the previous one by increasing the value of a (lines 9 and 10), the left hand-size interval bound. MIMETIC is recursively called in line 11.

The first call is MIMETIC(I, σ, k, δ, 1, 0, $\emptyset$) with $I = \bigtimes_i [0,1]$.

The function TopKDiv (see Algorithm 2) computes the top-k diversified communities. It loops over the ordered list of k already found diversified communities (lines 4 to 10) and stores the ones that are similar to pattern *Closed* in the variable *similarPatterns*. If a similar pattern has a higher Q value than *Closed*, the loop stops as pattern *Closed* will not be inserted to the list $\mathcal{K}$ (lines 8 and 9). If there is no similar pattern in $\mathcal{K}$, either (1) *Closed* will replace the worst pattern and $\min_Q$ value is updated, if the size of $\mathcal{K}$ is k (lines 13 to 16), or (2) it will simply be added to $\mathcal{K}$ (line 18). If similar patterns exist and if *Closed* has a higher Q value than all of them, then all of the similar patterns are removed and *Closed* is inserted to $\mathcal{K}$. Notice that $\min_Q$ is only updated if the list $\mathcal{K}$ contains k patterns.

Algorithm 2: TopKDiv($\mathcal{K}$, $Closed$, k, δ, $\min_Q$)

Input: $\mathcal{K}$ is a set ordered with respect to Q of up to k diversified communities.
Output: $\mathcal{K}$, the updated sorted list, as well as the current value of $\min_Q$.

```
similarPatterns ← ∅
i ← 0
isBetter ← true
while i < |K| and isBetter do
    J ← IP[i]
    if Jaccard(J, Closed) ≥ δ then
        insert(J, similarPatterns)
        if Q(J) > Q(Closed) then
            isBetter ← false
    i ← i + 1
if (|similarPatterns| = 0) then
    if (Q(Closed) ≥ δ) then
        if (|K| = k) then
            remove(K[0]), K)
            insert(Closed, K)
            min_Q ← Q(K[0])
        else
            insert(Closed, K)
else
    if isBetter then
        forall J ∈ similarPatterns do
            remove(J, K)
        insert(Closed, K)
        if (|K| = k) then
            min_Q ← Q(K[0])
```

4. Experimentation

We evaluate the performance of MIMETIC[2] on the following aspects: (1) Scalability: How does MIMETIC performs on real-world networks of large size? (2) Effect of parameters: What are the impact of the parameters on the communities quality? (3) Description of the obtained communities: What are the characteristics of the communities? (4) Comparison with competitors: How does MIMETIC perform in comparison to state-of-the-art methods on a dataset having ground-truth communities? (5) Impact of semantic on community quality: Does the use of semantic improve upon community detection? (6) Does MIMETIC provide human-understandable network communities?

4.1. Description of the datasets and baselines for comparison

For our evaluation, we consider three real-world multilayer networks for which we have a text that describes each layer. For one of them, we also have access to explicit ground-truth community labels. **Cit-HepTh**[3] is a citation network based on high-energy physics theory papers where each paper corresponds to a layer and the referenced papers are the nodes of the graph that are linked by a co-citation relation. Paper's abstracts are used as layer semantic description. **IMDB**[4] is a movie database from which we derived a multilayer network of actors with movies as layers. Actors who play in the movie are part of the layer. The layer's descriptions are the movie's synopsis. **Wiki-Topcats**[3] is an information network derived from Wikipedia articles. Layers represent Wikipedia articles and are attributed with the page's title. Nodes of the layers are linked by a co-reference relation. Each page is also associated to one or several categories that we use as ground-truth communities. The availability of such ground-truth allows us to evaluate community detection methods by quantifying the degree of agreement between the obtained and the ground-truth communities. The table below lists the networks and their properties:

[2]MIMETIC is implemented in Python and experiments are performed on 8 Intel(R) Xeon(R) W-2125 CPU @ 4.00GHz cores 126GB main memory.
[3]https://snap.stanford.edu/data/
[4]https://www.kaggle.com/stefanoleone992/imdb-extensive-dataset

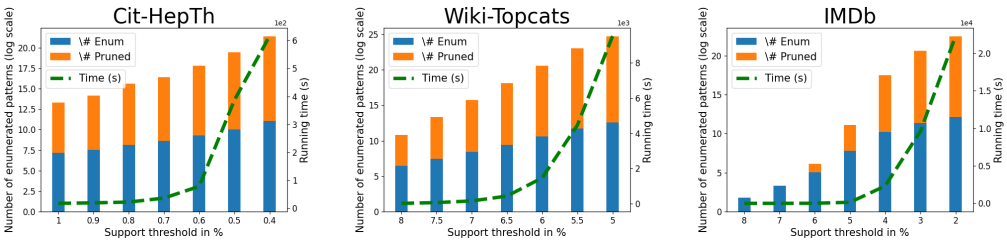


Figure 2. Number of enumerated and pruned patterns, and running time ($\delta = 1$, $k = 15$).

Dataset	# nodes	# edges	# layers	average layers' density	average density with $M(v,i) > 0$
Cit-HepTh	13921	2336992	14932	1.61×10^{-6}	2.95×10^{-6}
Wiki-Topcats	10000	1561063	9891	3.16×10^{-6}	6.48×10^{-5}
IMDB	18125	65033	9975	3.96×10^{-8}	2.76×10^{-3}

We consider five competitors: MiMag [7], Mucha [13], EMCD[12], MNE[15] and Infomap[14], as discussed in section 2 following the classification in [6].

4.2. **Algorithm evaluation**

First, we consider the behavior of the algorithm MIMETIC on the datasets. Figure 2 presents the number of enumerated and pruned patterns (using the upper-bound UBQ defined in equation 3.1) as well as the running time depending on σ. We can observe that MIMETIC succeeds to extract communities for very low thresholds (between 0.5% and 5% depending on the dataset) on the number of nodes per community. We see that the upper bound on the Q measure makes it possible to prune a large number of patterns, especially when σ is low. Finally, we observe that computation time increases as the threshold becomes very low, an expected behavior since, in this case, very small communities are numerous.

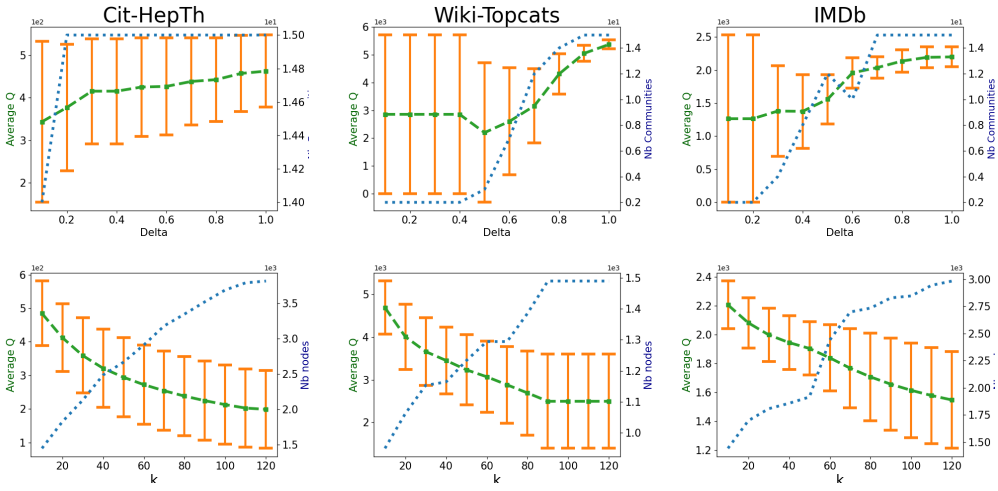


Figure 3. Mean and standard deviation of Q and the number of communities w.r.t. δ (Top), the number of nodes in the communities w.r.t. k (Bottom) for Cit-HepTh ($\sigma = 100$), Wiki-Topcats ($\sigma = 600$) and IMDB ($\sigma = 600$) (default $\delta = 0.8$, $k = 15$).

4.3. **Effect of parameters**

Figure 3 shows the behavior of MIMETIC as a function of the parameters δ and k. Recall that the diversity constraint requires that the proportion of nodes in common between two communities is less than δ. Thus, when δ is small, the number of communities may be less than k. Moreover, this constraint influences the quality of the communities, and, here again, low values of δ induce lower values of Q. From these experiments, it appears that $\delta = 0.8$ is a good compromise between quality and fairly diverse communities. Let us now consider the evolution of the quality of communities according to k. We observe that we can extract a large number of communities with a regular decrease in their average qualities. There is also a steady increase in the number of nodes present in at least one community.

4.4. **Describing the communities**

Figure 4 shows the communities obtained on each dataset for different values of k. The average number of nodes per community is shown on the first row. We can see that it remains well above σ, even when k increases. The average number of layers per community is shown on the second row. This number slowly decreases with k. It is quite small while being greater than 1, indicating that a few layers are involved in each community. This is because the generalized membership function of a node to a layer to which it does not originally belong to is significant only if the two layers are semantically very similar. Otherwise, we would have to bring together layers that are too different and do not correspond to the same reality. The third row displays the mean and standard deviation of the Jaccard index between community pairs. The communities in Cit-HepTh are very dissimilar, with an average value around 0.10. For the other two datasets, the communities are a little more similar, but the similarity decreases readily with k.

4.5. **Comparing the communities of MIMETIC with those of competitors**

To assess MIMETIC's ability to identify relevant communities, we consider the Wiki-Topcats dataset for which we have "ground-truth" communities. To compare the obtained communities M with the ground-truth ones $M^\star$, we adopt the measure in [2]:

$$\frac{1}{2|M^\star|}\sum_{m_i\in M^\star}\max_{m_j\in M}\Delta(m_i,m_j)+\frac{1}{2|M^\star|}\sum_{m_j\in M}\max_{m_i\in M^\star}\Delta(m_i,m_j).$$

Each community of a set is matched with its most similar community of the other set, with Δ a similarity measure between two communities. We consider two standard metrics: the F1 score and the Jaccard index. Hence, for each method, we obtain a score between 0 and 1, where 1 indicates the perfect recovery of ground-truth communities. As MiMag algorithm throws an out-of-memory error when the number of layers is greater than 2500, we consider a reduced Wiki-Topcats dataset made of the 50 categories with greatest number of pages. To run MIMETIC, we fix $k = 50$ and set σ to 10 so as to obtain 50 communities for different δ values from 0.2 to 0.8. Figure 5 (left) presents the results. We can observe that MIMETIC performs very well on both metrics and succeeds to identify the 50 communities. MiMag method provides 4 communities that only cover 2% of the nodes. Mucha's approach produces 371 communities: One of 1192 nodes, 7 with around 40 nodes, and the others with few nodes. The communities are not homogeneous in terms of page categories. Infomap gives also results that are below for both measures. MNE, that computes embedding vectors of the multilayer graph nodes while preserving their proximity, requires a post-treatment to obtain communities. We use k-means with $k = 50$ to get communities. The obtained results are also very low. Finally, we do not report the results obtained

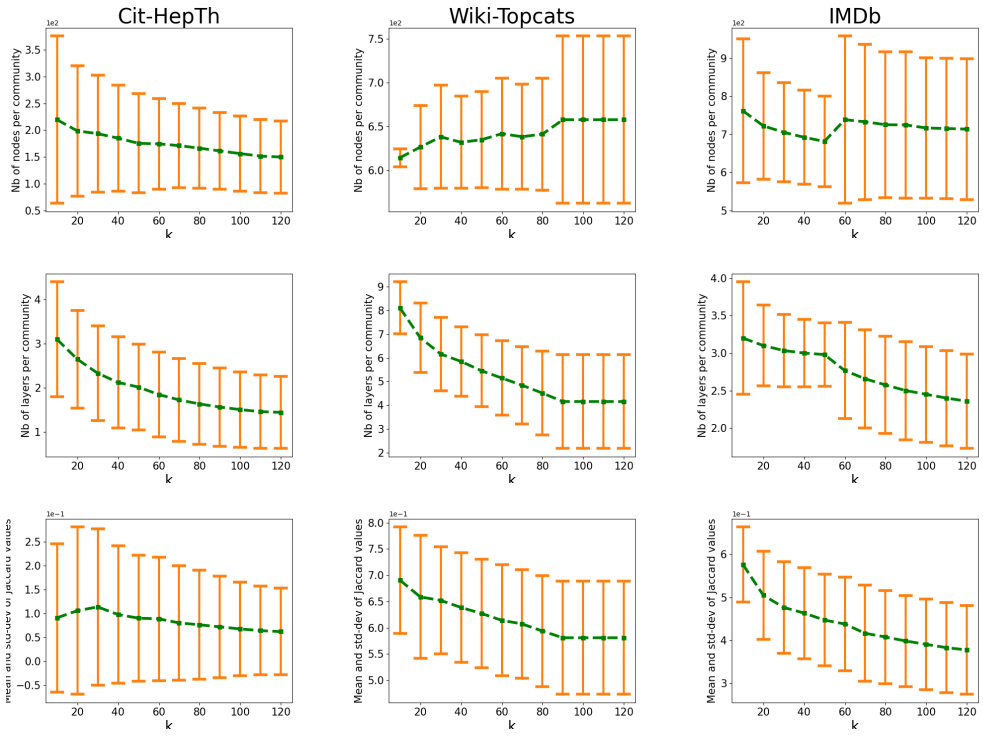
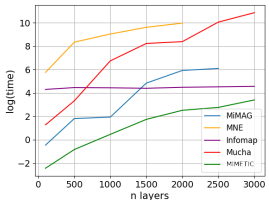


Figure 4. Number of nodes (Top) and layers (Middle) per community. Mean and standard-deviation of the Jaccard values between communities (Bottom) w.r.t. k for Cit-HepTh ($\sigma = 100$), Wiki-Topcats and IMDB ($\sigma = 600$) ($\delta = 0.8$).

by EMCD as this method is not working properly when the number of layers exceeds 30 and returns one node per community. Regarding the execution time (Figure 5 Middle), all methods are slower than MIMETIC.

Method		Jaccard	F1 score
MIMETIC	$\delta = 0.8$	0.283	0.381
	$\delta = 0.6$	0.307	0.422
	$\delta = 0.4$	0.311	0.431
	$\delta = 0.2$	0.313	0.426
Mucha		0.137	0.187
MiMag		0.071	0.124
Infomap		0.197	0.285
MNE		0.036	0.070



	MIMETIC-Bool		MIMETIC/MIMETIC-Bool	
δ	Jaccard	F1 score	Jaccard	F1 score
0.8	0.240	0.332	+17.9%	+14.7%
0.6	0.224	0.315	+37.0%	+33.9%
0.4	0.261	0.364	+19.1%	+18.4%
0.2	0.265	0.374	+18.1%	+13.9%

Figure 5. Wiki-Topcats: Adequacy to ground truth (Left); Running times in $\log(s)$ w.r.t. # layers ($\sigma = 10$, $k = 50$ and $\delta = 0.4$) (Middle). Upturn of MIMETIC over MIMETIC-Bool (Right).

4.6. **Does the semantic improve the quality of the communities?**

To assess if the semantics has an influence on the quality of obtained communities, we changed the membership function of Definition 3 so as $M(v,i) = 0$ if $v \notin V_i$. We call the obtained method MIMETIC-Bool. Note that this method is equivalent to ABACUS[8]. We set $\sigma = 5$ so as to obtain 50 communities for δ values between 0.2 and 0.8. Figure 5 (right) presents the results and shows that using the semantic improves the quality of the returned communities from 14% up to 37%.

4.7. **Examples on IMDB**

We computed communities on a subset of 2000 movies involving 6847 actors of the IMDB dataset. While MiMag, and Mucha return small communities that cover at most 2 layers (movies), MIMETIC ($\sigma = 5$, $\delta = 0.15$) finds larger communities of 3 to up to 5 movies. The movies in a same community not only have common actors, but also share the same category and have close synopsis. For example, we find a community of 12 actors and 3 movies that all belong to the category “science-fiction" and have common words in their descriptions such as “alien", “spaceship", “astronauts" and “planet". Another one corresponds to "drama" with keywords “husband", “affair" and “love".

5. **Conclusion**

We introduced *attributed multilayer graphs* to endow multilayer networks with layer's semantic attributes. We defined a membership function to assess how semantically similar nodes are across layers, and computed these similarities in practice with sentence embedding. We proposed MIMETIC, that is build on closed interval patterns to fine top-k diversified communities. We compared it to state-of-the-art methods on a dataset with ground-truth communities and obtained more diversified and higher quality communities. We also demonstrated that considering the similarity between layers improves the quality of the communities, and all of that in less running time.

References

[1] S. Fortunato. “Community detection in graphs”. In: *Physics Reports* 486.3-5 (2010).

[2] J. Yang, J. J. McAuley, and J. Leskovec. “Community Detection in Networks with Node Attributes”. In: *ICDM*. 2013, pp. 1151–1156.

[3] X. Wang, D. Jin, X. Cao, L. Yang, and W. Zhang. “Semantic Community Identification in Large Attribute Networks”. In: *AAAI*. 2016, pp. 265–271.

[4] J. Kim and J.-G. Lee. “Community Detection in Multi-Layer Graphs: A Survey”. In: *SIGMOD Rec.* 44.3 (2015), pp. 37–48.

[5] S. Pramanik, R. Tackx, A. Navelkar, J.-L. Guillaume, and B. Mitra. “Discovering Community Structure in Multilayer Networks”. In: *DSAA*. 2017, pp. 611–620.

[6] O. Hanteer, R. Interdonato, M. Magnani, A. Tagarelli, and L. Rossi. “Community Detection in Multiplex Networks”. In: *arXiv* (2019).

[7] B. Boden, S. Günnemann, H. Hoffmann, and T. Seidl. “MiMAG: mining coherent subgraphs in multi-layer graphs with edge labels”. In: *KAIS* 50.2 (2017), pp. 417–446.

[8] M. Berlingerio, F. Pinelli, and F. Calabrese. “ABACUS”. In: *DataMin. Knowl. Discov.* 27.3 (2013), pp. 294–320.

[9] R. Interdonato, A. Tagarelli, D. Ienco, A. Sallaberry, and P. Poncelet. “Local community detection in multilayer networks”. In: *Data Min. Knowl. Discov.* 31.5 (2017), pp. 1444–1479.

[10] X. Huang, D. Chen, T. Ren, and D. Wang. “A survey of community detection methods in multilayer networks”. In: *Data Min. Knowl. Discov.* 35.1 (2021), pp. 1–45.

[11] C. Bothorel, J. D. Cruz, M. Magnani, and B. Micenkova. “Clustering attributed graphs: models, measures and methods”. In: *Network Science* 3.3 (2015), pp. 408–444.

[12] A. Tagarelli, A. Amelio, and F. Gullo. “Ensemble-based Community Detection in Multilayer Networks”. In: *Data Mining and Knowledge Discovery* (Sept. 2017).

[13] P. J. Mucha, T. Richardson, K. Macon, M. A. Porter, and J.-P. Onnela. “Community Structure in Time-Dependent, Multiscale, and Multiplex Networks”. In: *Science* 328 (2010).

[14] D. Edler, L. Bohlin, and M. Rosvall. “Mapping Higher-Order Network Flows in Memory and Multilayer Networks with Infomap”. In: *Algorithms* 10.4 (2017), p. 112.

[15] H. Zhang, L. Qiu, L. Yi, and Y. Song. “Scalable Multiplex Network Embedding”. In: *IJCAI-18*. 2018, pp. 3082–3088.

[16] T. Mikolov, K. Chen, G. Corrado, and J. Dean. “Efficient Estimation of Word Representations in Vector Space”. In: *ICLR 2013*. 2013.

[17] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *NAACL*. 2019, pp. 4171–4186.

[18] M. Kaytoue, S. O. Kuznetsov, and A. Napoli. “Revisiting Numerical Pattern Mining with Formal Concept Analysis”. In: *IJCAI*. 2011, pp. 1342–1347.