Principal Patterns on Graphs: Discovering Coherent Structures in Datasets

Kirell Benzi\(^1\)*, Benjamin Ricaud\(^1\), Pierre Vandergheynst\(^1\)

\(^1\) Laboratoire de Traitement des Signaux 2, École Polytechnique Fédérale de Lausanne, Lausanne, Vaud, Switzerland

* Corresponding author
E-mail: kirell.benzi@epfl.ch (KB)

Abstract

In this work, we propose a fast, robust and scalable method for retrieving and analyzing recurring patterns of activity induced by a causal process, typically modeled as time series on a graph. We introduce a particular type of multilayer graph as a model for the data. This graph is structured for emphasizing causal relations between connected nodes and their successive time series values. Within the data, the patterns of activity are assumed to be dynamic, sparse or small compared to the size of the network. For some applications they are also expected to appear in a repeated manner over time but are allowed to differ from an exact copy. The analysis of the activity within a social network and within a transportation network illustrates the power and efficiency of the method. Relevant information can be extracted, giving insights on the behavior of group of persons in social networks and on traffic congestion patterns. Moreover, in this era of big data, it is crucial to design tools able to handle large datasets. Our approach scales linearly with the dataset size and is implemented in a parallel manner. By leveraging a state-of-the-art data analytics framework, our implementation can be distributed on clusters of computers and easily handles millions of nodes on a single commodity server.

1 Introduction

The number of applications that can be modeled as graphs as well as the number of methods dedicated to their analysis are exploding. In the present work we focus on retrieving information from graph-structured data, i.e. data associated to the vertices of a given graph, as is done in the emerging field of graph signal processing \[31\]. Moreover, we take advantage of some property of the data set, namely causality, to model it as a multilayer graph \[21\]. Typically, such structure and data organization can be seen in applications involving time-dependent, or more generally, causal processes on networks.

Dynamical activity in networks is a subject of high interest, for example in neuroscience for the study of the brain \[33\] or in physics for the study of epidemics \[20\] or rumor spreading \[14\]. In such problems, there are evidence of cascades or avalanches of activity as it propagates and / or spreads over the network \[6\] \[14\]. One approach to quantify dynamics has been the study of self-organized criticality \[4\]. It consists in measuring the distribution of sizes of the activity components (avalanches). This distribution exhibits a power law in what is called the critical regime. It has given new insights on the brain mechanisms \[33\] \[6\] although the relation to \[4\] and to the term ‘criticality’ is still controversial \[7\]. In the present work we want to go beyond the distribution of sizes of dynamic components. We provide a general method able to extract dynamic patterns in causal processes happening in networks. Furthermore, we propose several ways to analyze the patterns.
Indeed, analyzing a few representative patterns may provide information on the dynamical activity complementary to or even more accurate than the study of global properties of the whole set of patterns. An example of pattern is given on Fig. 1. To illustrate our idea, let us take three examples (treated in the application part).

Let us assume the dataset is a record of the activity of persons connected through a social network. A person can become active spontaneously or as a consequence of the activity of its neighbors in the network. This activity has been recorded over time and is given as a time series associated to each user. For example, it would be of interest to detect communities of persons that frequently activate together or to characterize a group of initiators that often react early within activation components.

In the second example, we would like to study the traffic in a transportation network. The dataset is a graph (transportation network) where each vertex may be a road or a pathway to which is associated a time-series of congestion rate. The number of vehicles is recorded periodically during several days or weeks. To make the traffic more fluid, it would be of interest to spot the places of recurrent congestion and understand its spreading. A congestion pattern is a dynamic event made of several connected pathways that are congested during the same period of time. It may be the consequence of a traffic jam, a slow down, or an accident. From an original event in the network, the congestion spreads to its neighboring roads which in turn can cause congestion on other adjacent roads at later time.

In the third example, we show the versatility of our method by studying the creation of music playlists. Using our model, we propose to build a recommender system based on the causal relation between songs, that is, their specific arrangement in a playlist. From a large corpus of user-annotated playlists, we build a graph of songs and use the appearance of each track as a dynamic activity over it.

![Figure 1: Illustration of a causal process on a graph evolving from neighbor to neighbor.](image)

To each node is associated a value giving its state (congestion rate for a transportation network or activation rate for a brain region). The value of the signal on the graph is pictured by a color at each node (gray is zero, blue one). At each time-step, non-zero values shift or propagate to some adjacent nodes. The blue edges represent the causality between events. An activation pattern is the set of all the activated nodes spatially and temporally connected (blue nodes). A node at time $t$ is connected to its counterpart and its neighbors counterpart at time $t+1$.

These examples show that complex systems can have some regularity or peculiarity in their dynamics, which is an important information to be retrieved for their understanding. Based on this “activation” behavior, our approach aims at retrieving activity patterns representative of the dynamics.

2
The evolution or propagation of the network activity both in space and time calls for an equal treatment of the spatial and temporal part of the data. In this regard, we introduce a particular type of multilayer graph that we call a causal multilayer graph. This graph models the temporal correlation of the signal on the nodes of the spatial graph. A multilayer graph is a graph $H$ made of layers $\{L_1, L_2, L_3, \ldots, L_N\}$, each one being a distinct subgraph of $H$. Layers are connected together by inter-layer edges. These structures are also called multislice or multiplex networks see for instance [16, 29] and references therein. A graph where each vertex is associated to a time series can be cast into a multilayer graph. Each layer $L_t$ of a graph $H$ is a (possibly modified) copy of the spatial graph, associated to one time-step, and possesses the same number of vertices but may have different edge configurations or edge weights. Inter-layer connections can be made according to different rules and we propose here a rule adapted to our applications, accounting for the causality relations between neighbors: neighbors on the initial (spatial) graph are connected across two successive time-steps to account for their proximity in space-time. This is well suited for tracking activity from vertex to vertex and time-step to time-step. On the causal multilayer graph, each vertex is now associated to a datum (one value of the initial time series), see Fig. 1. This can be seen as a label describing the vertex “state”, the rate of congestion for the transportation network or the person state or activity in the social network.

There is a huge amount of literature concerning the analysis of multilayer graphs and the applications they model [21]. However, our approach differs from the existing ones in many ways. Firstly multilayer graph analysis is mainly focused on extracting information from the graph and not from data associated to it (to its vertices in our case). Secondly, many approaches give a limited importance to inter-layer connections. For example, temporal graphs [17] only assume connections between a vertex and its counterpart on the next layer. Whereas our causal multilayer graph allows a vertex to be connected to its counterpart and to the counterparts of its neighbors in the next layer (even more freedom is given in the generalized version of our graph). Thirdly, many methods (e.g. node centrality, clustering) are graph analysis tools adapted to the multilayer case but do not really leverage the multilayer model. The multilayer graph is seen as a standard graph, possibly with distinctive treatment of inter-layer edges. In our case, having a multilayer graph is a crucial point allowing to take advantage of the regularity brought by this structure. Fourthly, we provide a fast and scalable implementation of our method, which is rarely the case in the analysis of multilayer graphs.

The dataset can initially associate a real number (or even a feature vector) to each vertex of the causal multilayer graph, and a first pre-processing step must cast the vertices into two (or more) classes. In the case of the traffic or brain examples, the congestion rate and region activity respectively are cast into a binary value of congested / not congested and activated / not activated possibilities. After labeling, one of the classes of vertices is chosen and its elements are called activated vertices. Keeping only these vertices and their edges, the initial graph becomes a union of disconnected subgraphs as the rest of the vertices and the edges to them are ignored. The patterns of activity we are interested in are these subgraphs, named activated components. We must stress that we assume here some sparsity in the activations: the subgraphs must be small compared to the causal multilayer graph, in order to find repeated patterns.

With the exception of grid-like structures, networks do not generally have a regular topology. It is then difficult to detect special configurations of the vertices and edges which may appear in a repeated manner. One may rely on graph matching algorithms to compare different activated components but it is computationally intensive and does not scale for large graphs. In addition, if the patterns differ slightly from each other by a few vertices or connections, graph matching will fail to group them into the same class of patterns. Our causal multilayer graph structure brings a crucial help as it i) extracts subgraphs in linear time and ii) classifies them robustly in quasi-linear time.

Our aim is now to classify these subgraphs knowing that many of them are repeated inside $H$. Depending on the dataset nature, its level of noise or its error rate in the data, the labels may contain errors (misclassified vertices). In addition, activation patterns may slightly depart from their typical
shape in real world applications. As a consequence, the subgraphs representing the same activity may differ and have to be classified in a robust manner. The classification relies on a unsupervised clustering algorithm. Since there can be a large number of activated components, the $k$-means algorithm was chosen as it scales nicely with the load in quasi-linear time using Lloyd’s method [23]. However, the entries of the algorithm must be feature vectors and not subgraphs. A crucial step of our method resides in the conversion from subgraph to feature vector, with the help of the layered graph structure.

In the first part of this work we are interested in a particular type of multiplex networks, one where each layer contains the same number of vertices which implies that each vertex possesses a counterpart in all the layers. The motivation is given by our applications that are well modeled by this type of network. The time scale of our analysis is a few days for the transportation network and for the social network activity. We assume the networks to be static, with the same number of vertices and connections over the recording. In addition, the concepts and the different steps are made easier to present and understand for this particular subgroup of networks.

We describe this basic causal multilayer graph, explain how to construct it efficiently, how to extract dynamic activation components and how to cluster them. The next section is dedicated to the full generalization of our approach by allowing a different number of vertices from layer to layer, different intra-layer and inter-layer connections, hence generalizing the approach to a larger class of multilayer graphs.

Finally, we present three applications to illustrate our method. As a first application, we extract dynamic patterns of activity in a social network. The second application is devoted to the analysis of the flux of pedestrians in a train station. Finally, we show interesting patterns extracted from more than 100,000 collaborative music playlists. As part of the open science movement, the code related to this work is freely available under the GPL v2 license [9].

2 A simplified causal multilayer graph for the analysis of dynamic patterns in 3 particular examples

The causal multilayer graph in its basic version is a convenient data structure to represent causality between consecutive events on a graph. To construct one, several ingredients are required: 1) a spatial graph $G$ of $N$ vertices which is to be replicated at each time step, forming the layer structure. 2) A wiring rule to connect layers together. 3) A temporal signal $S$ on the graph made of $N$ time-series of length $T$, one per vertex of $G$. This signal allows for the selection of subgraphs of interest (through an activation mask). We review and define each of them in the following section.

2.1 Definition

We call spatial graph $G = (V_G, E_G, W_G)$, the graph which encodes the structural connectivity between the vertices onto which the time-series is defined. For the sake of simplicity we assume $G$ to be unweighted and undirected although it is possible to apply our method to weighted and directed graphs. Here $V_G$ is the set of nodes (vertices) with $|V_G| = N$, $E_G = \{(i,j,w_{ij}) | i,j \in V_G, w_{ij} \in W_G\}$ is the set of edges and $W_G$ is the weight matrix. The graph $G$ defines the layers of the causal multilayer graph. To each time step $t \in [0, \cdots, T-1]$ is associated a layer $L_t = G$. Note that while we use a static spatial graph in our applications, a dynamic one (a graph which evolves through time) is completely supported by our framework. In that case $L_t = G_t$ where the set of graphs $\{G_t\}_t$ is given instead of the single graph $G$. This is developed in the last section. We denote by $i_t$ the vertex of layer $L_t$ that corresponds to vertex $i$ on $G$. Since each layer is a copy of $G$, the set of intra-layer connections in the causal multilayer graph, connecting vertices within each layer, is the set $\Omega = \{(i_t, j_t, w_{ij}) | i, j \in V_G, w_{ij} \in W_G, t \in [0, \cdots, T-1]\}$.

The second requirement is the rule for connecting the layers $\{L_t\}_t$ together in order to form the causal multilayer graph $K$. In the applications presented here, the graph elements have an influence
over their neighbors during successive time steps (causality). Since we want to follow this influence, we construct the inter-layer connection accordingly. A vertex $i_t$ will be connected to its neighbors at time $t$ and time $t + 1$. The set of edges connecting vertices to their neighbors at the next time step is $\Omega_x = \{(i_t, j_{t+1}, 1) \mid i, j \in V_G, w_{ij} \in W_G > 0, t \in [0, \cdots, T - 2]\}$. The weight value for the inter-layer edges may be set to an arbitrary value, we set it to one here as we want to have an equal treatment of the spatial and temporal dimensions. In order to follow an activation of the same vertex through time, for all time steps $t \in \{0, 1, \cdots, T - 2\}$ each vertex $i_t$ of $L_t$ is also connected to "itself", $i_{t+1}$, in $L_{t+1}$. This set is denoted $\Omega_{xs} = \{(i_t, i_{t+1}, 1) \mid i \in V_G, t \in [0, \cdots, T - 2]\}$. Therefore the causal multilayer graph $K$ is made of $N \times T$ vertices and the set of edges:

$$E_K = \Omega \cup \Omega_x \cup \Omega_{xs}. \quad (1)$$

The Fig. 2 illustrates the causal multilayer graph $K$. Each edge color corresponds to a set, black for $\Omega$, dark blue for $\Omega_x$ and light blue for $\Omega_{xs}$. Notice that if the weights of all the edges of the causal multilayer graph $K$ are equal to one and the connections are undirected, $K$ is exactly the strong cartesian product of the graph $G$ with the path graph of $T$ vertices, as defined in [30].

Figure 2: The causal multilayer graph $K$ with its edges configuration for a given graph $G$ and 3 time steps. $G$ is a path graph with 3 nodes labeled $A$, $B$ and $C$. In black are the intra-layer connections corresponding to the ones of $G$. In dark blue are the inter-layer connections between a vertex and its neighbors (in $G$) at the next time step. In light blue we can see the connection of the vertices with themselves at the next time step.

In a variant of the causal multilayer graph, we may also drop out the spatial edges i.e. the intra-layer connections. This strengthens the dynamical aspect as the emphasis is put on the causal relations only. Reducing the number of edges also speeds up the computations and allows clearer representations on figures. This variant of the causal multilayer graph possesses the following set of edges: $\Omega_x \cup \Omega_{xs}$.

The third ingredient is a dataset made of $N$ time-series of the same length $T$, one for each vertex of $G$. The dataset is a matrix $S$ of size $N \times T$. One value of $S$ is associated to each vertex of $K$. This value describes the vertex state. In the applications, we want to extract patterns where vertices have the same state, for example activated regions of the brain or congested locations in the transportation network. From the signal giving this state, we compute what we call the activation mask $M$. It is a binary mask, the entries are either 0 for inactive or 1 for the “activated” vertices. Let us denote by $S(i, t)$ the value of $S$ at vertex $i_t$ of $L_t$. In our applications we use a simple threshold on the set of time-series values to create the activation mask:

$$M(i, t) = \begin{cases} 1 & S(i, t) > \mu, \mu \in \mathbb{R}, \\ 0 & \text{otherwise}. \end{cases}$$
2.2 Construction and extraction of subgraphs

At this point the reader should have noticed that the multilayer graph $K$ is of size $N \times T$ and can be extremely large. Since we are interested only in subgraphs of the global causal multilayer graph (the ones whose nodes have a non-zero mask entry) it is possible to avoid its full construction and build directly the subgraphs in a parallel manner.

The reduced causal multilayer graph $H = (V_H, E_H, W_H)$ is a directed graph constructed by only taking into account the set of activated vertices of $K$. We say that $i_t \in V_K$ is activated if $M(i, t) = 1$. The edges of $H$ are the ones of $K$ connecting activated vertices. In the applications we are only interested in $H$. Instead of constructing $K$ and then dropping inactive vertices and their connections, which can take a prohibitive computational time, we will create directly $H$ from $G$ and the activation mask $M$. The Fig. 3 illustrates a possible outcome of the creation of the reduced causal multilayer graph: given $G$ with $N = 3$ and the mask $M$ with $T = 3$, $H$ is created as a set of subgraphs (here there is only one subgraph). On the figure, only the inter-layer connections called causal edges, are drawn.

Figure 3: **Creation of the reduced causal multilayer graph** $H$. It uses the spatial graph $G$ (top right) and the activation mask $M$ (on the top left). $A, B, C, D$ are the vertices of $G$, $t_0, t_1, t_2, t_3$ are the time-steps. The activation mask has been constructed by applying a user-defined function to the signal $S$ to select important, "activated" entries of the time series (set to 1). The activated nodes on layer $t$ are connected to activated nodes on layer $t + 1$ if they are neighbors in the graph $G$ or if they represent the same node in $G$. The arrows are the inter-layer connections. They may be directed or not, this has no consequence in our method. For clarity, the intra-layer edges are not represented.
2.3 An efficient construction

Before starting the construction, we get rid of the vertices which are never activated over time, i.e. if for some $i$, $M(i, t) = 0$ for all $t$, the vertex $i$ of $G$ is dropped. This step can greatly reduce the memory requirements and improves the construction speed if the mask is sparse.

Each binary activation vectors $M(i, \cdot)$ is stored as a property (label) on each node $i$ of $G$, creating a property graph which is still denoted as $G$.

The causal multilayer graph is created in a series of steps.

1. Iterating over each edge $e \in E_G$ of $G$, linking a source node $i$ and a destination node $j$, the algorithm first reads the vectors $M(i, \cdot)$ and $M(j, \cdot)$. An inter-layer connection is made between layer $t$ and $t + 1$ whenever $i_t$ and $j_{t+1}$ are activated (since we already know that $i$ and $j$ are spatial neighbors) i.e. $M(i, t) = M(j, t + 1) = 1$. That is to say, an edge exists in $\Omega_x$ whenever $M(i, t) \& M(j, t + 1) = 1$ where $\&$ is the logical And. We introduce a new vector $u_e$ of size $T - 1$ associated to the edge $e$:

$$u_e(t) = M(i, t) \& M(j, t + 1), \quad \text{for all } t \in [0, \cdots, T - 2].$$

The value $u_e(t)$ encodes the existence (1) or absence (0) of an inter-layer edge between vertex $i_t$ and $j_{t+1}$. The vector $u_e$ is stored as a property of edge $e$.

2. Since we are also interested in self activation of vertices across time (the set $\Omega_{xs}$) we compute an additional vector $u_{\text{self}, i}$ for each vertex $i$ of $G$,

$$u_{\text{self}, i}(t) = M(i, t) \& M(i, t + 1), \quad \text{for all } t \in [0, \cdots, T - 2].$$

The vector $u_{\text{self}, i}$ is stored as a property of node $i$.

3. The construction of the graph $H$ is then done by reading the collection of edge vectors $\{u_e\}_e$, node vectors $\{u_{\text{self}, i}\}_i$ and adding edges between successive time layers when ones are encountered.

The scalability of our method comes from the properties of $H$. Layers are time-ordered, allowing connections between layers to be encoded as binary vectors. To construct $H$ a few operations on vectors are needed, for each edge and node of $G$. Hence the complexity of this process is linear in the number of edges and vertices of $G$. It is also linear in the number of time-steps. In addition, the creation of edges only depends on the state of pairs of nodes. Therefore, it relies on local values which allows an efficient parallel implementation. The main task consists in handling the properties (vectors) of triplets $\{i, e, j\}$, where $e \in E_G$ is the spatial edge connecting source node $i$ and destination node $j$. This is sufficient to create all causal edges between $i$ and $j$ ($i \neq j$). This interesting property is particularly suited to large scale graph analytics frameworks such as GraphLab Create\footnote{https://dato.com/products/create/index.html} or Apache Spark, which possess dedicated processes applying functions to all triplets in parallel. Thus, our implementation gracefully scales with the number of cores and is much faster than the sequential naive implementation.

2.4 Implementation details

This section describes additional implementation tricks for a better computer efficiency of the method. The reader interested in the general method more than its implementation may skip it.

The first step of the algorithm reads both activation mask vectors $M(i, t)$, $M(j, t + 1)$ from $i$ and $j$ respectively as arbitrary precision integers \cite{Liu2016}. An arbitrary precision integer can store any integer number (limited by available memory), and can be seen as a list of “standard” 32 bits integers with an common interface. GraphLab Create only allows to store basic datatypes as properties of nodes
and edges such as integer, double, bool, and string (at the time of the writing). We had to transform
the rows of our activation mask $M$ to bitstring to be able to store it on vertices and edges. Any bit
compression algorithm can be used to reduce storage space. The arbitrary precision integer stored as
a string offers a compression ratio of more than 3 over the raw bitstring.

In the second step of the algorithm, the vector $u_e(t)$ is created by performing a logical And ($\&$)
between $M(i, t)$ and $M(j, t+1)$. It amounts to performing a logical And between two vectors: $M(i, \cdot)$
and the left-shift version of $M(j, \cdot)$ (hence involving a logical And and a bit shift). The last (least
significant) bit in this operation is dropped ($u_e$ is of size $T-1$) as it would correspond to a link between
layer $T-1$ and $T$ and this latter does not exist. Once the vector $u_e(t)$ is created, all the ones have to
been found to create the causal edges. For each 1, its position in the vector gives the time $t$ and allows
the creation of two pairs $(i, t)$, $(j, t + 1)$, a causal edge, which is then added in the causal multilayer
graph $H$. Instead of looping through all the bits of $u_e(t)$ and checking for a 1 at each position, we have
implemented another strategy which jumps from 1 to 1 in $u_e(t)$ and gives the position of the layer
number $t$. This optimization is better than the classical For loop when $u_e(t)$ is sparse. The details
are given in the algorithm provided in Algorithm 1. We invite the interested reader to refer to [3] for
more information on this low level bit manipulation trick. The Fig. 4 illustrates the algorithm which
is formally defined in Algorithm 1.

![Diagram](image)

Figure 4: Creation of all the edges of the causal multilayer graph. It is going from $A$ on one
layer to $B$ on the next layer. It requires the properties associated to $A, B \in V_G$ and $e \in E_G$ of the
triplet $(A,e,B)$, top left picture. The binary mask vector of the destination node $B$ is shifted (in
this illustration left-shifted) to align source layer $t$ and destination layer $t + 1$. Then, a logical And
is performed between the two vectors, top right picture. On the bottom figure, the result $u_e$ stored in $e$,
is read to create the edges (and nodes) of $H$.

### 2.5 Dynamic activated components

We define dynamic activation components to be the weakly connected components of the causal
multilayer graph $H$. Each component represents the activated values in the original timeseries and
how those values are causally linked together. For each component, we define its width to be the
number of layers over which the component spans and its spatial spread to be the number of distinct
nodes of $G$ over which the component spans.
**Input**: Graph $G$ having the binary activation vectors stored on its nodes

**Output**: Graph $H$

$H \leftarrow$ create empty directed graph;

**Parallel foreach** triplet $(src, e, dst) \in G$ do
  $ms \leftarrow$ read src binary vector $m$ as arbitrary-sized integer;
  $md \leftarrow$ read dst binary vector $m$ as arbitrary-sized integer;
  $u \leftarrow ms \& (md \gg 1)$;

  // Find all activated causal edges
  while $u$ is not 0 do
    // extract least significant bit on a 2s complement machine
    index $\leftarrow u \& \neg u$;
    $u \leftarrow u \oplus$ index // toggle the bit off;

    // Get activated layer number
    layer $\leftarrow -1$;
    while index is not 0 do
      index $\leftarrow index \gg 1$;
      layer $\leftarrow layer + 1$;
    end

    AddEdge($H$, $(src, layer)$, $(dst, layer + 1)$);
  end
end

**Algorithm 1**: Algorithm to create the causal multilayer graph $H$ on little-endian systems. The least significant bit is on the right.

The extraction of the weakly connected components (WCC) is done in parallel using the built-in function from GraphLab Create. Under-the-hood, it is implemented using the HCC algorithm [18]. The algorithm can be summarized as follows. Until convergence, set each vertex as a component, the component ID is the vertex ID. The IDs are increasing with $t$. Propagate the component ID to neighbors, if a vertex receives a smaller component ID, update its current value by the one received. Repeat till all vertices are read. The computational complexity is linear in the number of vertices of $H$.

At the end of this process, each subgraph of $H$ has been extracted and is now a graph of its own. We call this set of graphs the activated components. The graph IDs are the IDs in $H$ of their first activated vertex (if two or more vertices are activated at the same time, the ID is the one of the smallest index $i$). In order to detect repeated patterns in applications, we compare the activated components in the following section.

### 3 Extracting global patterns from the components

We assume that the set of activated components (subgraphs of $H$) is made of several patterns that repeat themselves (same vertices activated and same dynamical behavior). However, when looking at one particular pattern, each repetition may not be a perfect copy of itself. Some variations are allowed and account for the fact that activation patterns follow from real-world applications with possibly noisy measurements. In this section we show how to cluster all these imperfect copies together without relying on complex analysis of subgraphs.
3.1 Feature vectors for the description of activated components

Dynamic activated components in their raw form cannot be efficiently used as an input of a machine learning algorithm. It would require the use of graph matching algorithms in order to compare graphs and they are computationally intensive. Instead we propose a faster approach that emphasizes on the important information involved in the graph comparisons. We introduce two types of feature vectors. The first one, the dynamic-feature vector, encodes the time evolution of the activations and give information on the dynamics. It contains the full information concerning the dynamic arrangement of the component while being memory friendly and computationally efficient. The second one is the static-feature vector, a time-averaged version of the dynamic-feature vector. It is used as a more robust input to cluster the thousands of components before looking at the dynamics inside them.

Let $P_\ell$ be an activated component (a weakly connected subgraph of $H$), where $\ell \in [1, N_c]$ is the component label and $N_c$ is the number of components. The graph $P_\ell = (V_\ell, E_\ell)$ has a set of vertices $V_\ell$ and a set of edges $E_\ell$.

**Dynamic features.** The dynamic features are a straight-forward binary vectorization of the component. Let us assume $P_\ell$ is activated during $k_\ell$ time steps from $t_1$ to $t_{k_\ell}$. Let $K = \max_\ell k_\ell$ be the maximal width of a component (in number of layers). The dynamic-feature vector $d_\ell$ describing $P_\ell$ is of length $N \times K$, with binary entries: 1 if a node belongs to the graph $P_\ell$ and 0 otherwise, arranged such that:

$$d_\ell(j) = \begin{cases} 1 & \text{if } j = i \times k \text{ and } i_k \in V_\ell, \\ 0 & \text{otherwise.} \end{cases}$$ (2)

Notice that most of the vectors are very sparse and memory-friendly.

**Static features.** A layer-invariant feature vector is constructed by compressing the activated component’s dynamic into one normalized histogram of spatial node occurrence. The static-feature vector $s_\ell \in \mathbb{N}^N$ has the following entries:

$$s_\ell(j) = \sum_{k=1}^{K} d_\ell(j \times k).$$ (3)

If necessary, the static-feature vector may be normalized using the $\ell^2$-norm to help cluster together components with similar activated nodes but of different temporal width. The idea is similar to the bag-of-word feature vector in text mining. While losing the dynamical aspect, an activation signature still remains in the static-feature vector. This is enough to perform a meaningful clustering in our 3 examples of applications.

3.2 Clustering

Depending on the application, several clustering algorithms can be used. In our applications, we have chosen the $k$-means++ algorithm for its simplicity and scalability. This algorithm partitions the dataset into $k$ sets ($k \leq N_c$), $S = \{S_1, S_2, \ldots, S_k\}$ so as to minimize the within-cluster sum of squares:

$$\argmin_S \sum_{\ell=1}^{k} \sum_{s_i \in S_\ell} \|s_i - \mu_\ell\|_2^2,$$ (4)

where $\mu_\ell$ is the centroid of $S_\ell$. We expect here $k$ distinct components that are repeating themselves, with copies possibly differing by a few nodes (due to noise or other activation behaviors). The value of $k$ is set manually and depends on the application. Using the Lloyd’s method, the complexity is proportional to $N_c N k i$ where $i$ is the number of iterations before convergence. In practice this number is small, so that $k$-means can be considered to scale linearly.
3.3 Analysis of the cluster properties and average activated component

While a richer dynamic behavior can be found by looking at raw dynamic activated components, averaging many components emphasizes common behaviors. Since we use static-feature vectors to create the clusters, the dynamic is lost in the “compression” and cannot be retrieved using the k-means centroids. However since each component has an Id we may go back their dynamic-feature vector. From them, we can construct average activated components by creating an histogram of the dynamic features for each cluster. The average activated component associated to each cluster is a causal multilayer graph. Each node $i_t$ has an associated value corresponding to the number of times the node $i$ is present in the layer $t$ of a component of the cluster. This average component, which can be computed from any clustering method, is representative of the average activity of the underlying patterns.

From the nodes and edges histograms, we may compute the node and edge likelihoods and store them as node and edge weights respectively. For a node, it represents the likelihood of activation at the particular layer. For an edge, it represents the chance of a node on layer $t+1$ to get activated from a node on layer $t$. Low weights are usually discarded to filter unlikely transitions and clean the component for further analysis or visualization.

4 The generalized causal multilayer graph model

In the previous section, we introduced a version of the causal multilayer graph model which will be used to extract patterns of interest in 3 different datasets in the application section. In this section, we generalize this model to account for dynamic spatial graphs, where edges or nodes are allowed to appear or disappear across layers. The location of nodes and edges through layers can be encoded as additional vectors associated to the nodes and edges of a base graph. Manipulating vectors as we do in the first part, we show that activated subgraphs within this generalized model can be constructed very efficiently using the technique introduced in 2.3.

Let us assume we have a set $\{G_t\}_t$ of $T$ graphs, one for each time-step $t \in [0, \cdots, T-1]$. The number of vertices and edges is allowed to change from graph to graph. However, at least some vertices must be matched through layers, i.e. can be associated to the same entity or identity. Otherwise there is no point treating the problem because repeated patterns in the sense presented in the first section do not exist. In other words, let $i_t$ denotes vertex $i$ of graph $G_t$, there exists vertices $i_{t_2}, \cdots, i_{t_n}$ in $G_{t_2}, \cdots, G_{t_n}$ which can be associated together by the id $i$.

A mask $M$ is given that associate a value ($0$ or $1$) to each vertex of the set $\{G_t\}_t$. In this case, it may not be a matrix. The mask can also be computed from a signal $S$ on the vertices of the collection of graphs $\{G_t\}_t$.

We first create a graph $G$, the base graph, which contains the vertices and edges between them that appear at least once in $\{G_t\}_t$. The base graph $G$ is defined as follows. A vertex $i$ belongs to $V_G$ if there exist some $t$ such that $i_t \in V_{G_t}$. An edge $e \in E_G$ connecting vertex $i$ and $j$ of $G$ exists if for some $t$ there is an edge $e_t \in G_t$ connecting $i_t$ and $j_t$.

The second step associates vectors to each vertex and edge of $G$. As in the first section, a vector $M(i, \cdot)$ of length $T$ is associated to each vertex $i$. The entries of the vector are the values of the mask. In the case where vertex $i$ is not present in some graph $G_t$ the entry $M(i, t)$ exists and is set to zero. It is equivalent to adding an extraneous vertex $i$, inactive and unconnected, in the layer $t$. We now introduce an additional mask: the edge mask $M_E$. It associate a binary vector of length $T$ to each edge of $G$. For each edge $e$ of $G$ between node $i$ and $j$, $M_E(e, t) = 1$ if there is an edge between $i_t$ and $j_t$, zero otherwise. Notice that the weights of the edges can be stored in an additional vector as we could do with the signal $S$ on vertices. For the sake of simplicity, we assume graphs to be unweighted here.

Eventually, we introduce a second set of vertex vectors $\{P_i\}_i$ and edge vectors $\{P_e\}_e$ called passage
vectors of size $T - 1$. They allow to configure the inter-layer connections, giving a complete freedom for the connection rules between successive layers $G_t$ and $G_{t+1}$. The only restriction is to have connections solely between successive time layers. For edge $e$ connecting $i$ and $j$ of $G$, the entry $P_e(t) = P(e, t)$ is one if a connection between $i_t$ and $j_{t+1}$ is allowed, zero otherwise. The model of the first section can be obtained as a particular case where all the entries are set to one if $i$ and $j$ are connected or $i = j$ and zero otherwise. Similarly for vertex $i$, the entry $P_i(t) = P(i, t)$ is set to one if a connection between $i_t$ and $i_{t+1}$ is allowed, zero otherwise.

Fig. 5 gives an example of the vectors values for a set of 3 graphs $G_0, G_1, G_2$ and some mask. For each vertex and edge, 2 vectors are given, one associated to the masks $M$ and $M_E$ respectively of length 3 and the other is the passage vector of length 2 $P_t$ and $P_e$. The resulting causal multilayer graph $H$ is given at the bottom.

Figure 5: **Generalization of the causal multilayer graph.** The spatial graph is directed and evolves across layers. The intra-layer connections are given by $G_0, G_1, G_2$ on the top right. On the middle right, the edges configuration is given as mask ($m$) and passage ($p$) vectors. The same is done for the nodes on the left. In the case of self-edges, some values of this mask can be irrelevant, marked with an $X$, if the node is not activated in the previous or next layer. On the bottom, the resulting causal multilayer graph $H$ is made of 2 activation components.

Fig. 6 explain the construction of the causal multilayer graph using the different vectors. Similarly to the first section model, for each edge $e$ the destination vertex mask is shifted to the left before performing a logical And with the source vector, giving $u_e$. In our generalized model we add a zero value at the end of the passage vector (no connection beyond layer $T - 1$) and perform a logical And with $u_e$. This sets to zero the unauthorized edges across layers. This is the same process at the vertex level, for self-edges (connections between $i_t$ and $i_{t+1}$).
The creation of causal edges in the generalized causal multilayer graph. On the top left are the data read for the creation of all edges between the source node $A$ and the destination node $B$ on the next layer. The passage vector $p$ stored on the edge $e$ allows or not the creation of a causal edge between successive layers. Bit wise operations are performed on the top right picture. The result is given on the bottom picture. In this example, the pair $(A_{t_0}, B_{t_1})$ is unconnected because the first entry of $p$ is 0. Only one edge between $A_2 = A_{t_2}$ and $B_3 = B_{t_3}$ is created.

The rest of the construction leading to $H$ follows the one of the first model. Once $H$ has been constructed, the creation of feature vectors and the clustering is identical.

5 Applications

We now apply the previously described method to various datasets and evaluate its efficiency at discovering dynamic patterns.

5.1 Scientific rumor via twitter activity

We study here a dataset of Twitter activity recorded over several days, available at SNAP [22]. This is a relatively large dataset with a graph of more than 0.4 million nodes and 14 million edges. From the time activity, we have created several set of time series with the longest ones being more than 14,000 samples. First, this dataset is a test for our parallelized algorithm and its ability to scale. More than showing the possibility of handling large datasets, it demonstrates its efficiency as the extraction of activated components takes only a few seconds on a community server with 24 cores. Second, these recordings give us a chance to compare our approach to state-of-the-art analyses of dynamic activity on networks such as the one provided in [14]. The code used for the analysis of this dataset is available online [8].

This dataset has been studied in [14] in order to understand rumor spreading in networks. The Twitter activity has been recorded before, during and after the announcement of the discovery of the Higgs boson by CERN on the 4th of July 2014 at 8:00 AM GMT. The recording starts from the 1st of July and last until the 7th. The authors have recorded the Retweet, Reply and Mention activity containing selected keywords related to the Higgs boson. The graph of twitter followers is provided together with the activity of the users during the event. The study has focused on dynamical aspects.
such as number of tweets per unit of time, interval of time between retweets and their evolution over
the days. The advantage of the twitter dataset over other datasets having a dynamic activity is that
the activity can be tracked over time using retweets. A retweet, in addition to giving a timestamp,
provides the causal link between two users. A user retweets an information as a consequence of a first
user having tweeted the information. The cascade of retweets can be followed without relying on a
causal multilayer approach. Note that the retweet information is asynchronous (not having regular
time steps) and can not be directly cast into a causal multilayer graph. It would involve connection
between layers not necessarily adjacent. The analysis of the retweet dynamics allowed the authors
of [14] to reveal the bursty behavior of retweet chains, in particular around the official announcement.

However, the retweet activity does not fully account for the spreading of rumors. For example a
user may see several tweets concerning the Higgs boson in his/her feed and decide to retweet only
one of them, or decide to come back to the initial source of information and retweet it instead of
the users he/she is following, or else tweet the information without mentioning his sources. In these
cases, the action of tweeting does not take into consideration the full user network even if it has a
clear influence on him/her. Actually, we have compared the graph of followers to the graph obtained
by connecting users according to the retweet data. Only 59% of the edges of the retweet graph are
matching the followers graph: a large portion of the retweets are not from direct neighbors. Our
multilayer graph approach can be used here to account for the neighbors influence on the users. This
brings complementary information on the dataset.

To track the activity over time and users, we cut the twitter recording into regularly spaced time
steps. Within a time step, a user is active if he/she has retweeted about the Higgs boson. Different
activity patterns may appear at different scale of time (from seconds to hours to days) and the time step
is chosen in order to select a particular scale. Remember that connections within the causal multilayer
graph are only allowed within layers and between two successive layers. The retweet action of a user
may occur after several time steps and this is not taken into account in our construction. However,
there is evidence of bursty behaviors in social activity [5, 19] and in particular in this dataset [14].
A retweet is likely to be done shortly after a tweet appears in the user feed and the probability of
retweeted decreases with time (following a power law). Hence the time series configuration capture
most of the dynamic activity. Moreover, it is also of high interest to focus on tracking the activity
solely due to the bursty behavior. Of course the time step length must be chosen in order to match
the time scale of the bursty processes: a length of 1 minute is a reasonable choice according to the
results of [14], see e.g. their Fig.5. The number of retweets drops exponentially after this time length.
We have also chosen a sampling length of 10 minutes to show the impact of such a choice.

As a proof of concept, the first goal in this example is to confirm the retweet dynamic made of
bursty events discovered in [14]. If this is so, our activated components should be large especially
around the announcement. The second task is to show that the activated components can bring new
insights on the dataset and on the rumor spreading mechanisms. Since the dataset is related to a
single extraordinary event we do not expect to find repeated patterns of activity. Nevertheless, the
analysis of repeated patterns could be done in future work with a larger recording of twitter activity,
containing different events appearing over several weeks.

To build the causal multilayer graph we use the activity over time (time series) and combine it
with the graph of followers (the social network) as follows.

**The graph** The graph is here the graph of followers given in the dataset. It is a directed network
where a user (source) is connected to another one (destination) if he is followed by him/her. The
graph is made of 456,626 nodes of users who have been active (retweet, reply, mention) at least once
during the recording and more than 14 million edges.

**The signal and the mask** From the retweet activity data, an activation signal is computed for
each user. The recording is cut in time-steps with a fixed length. We have chosen two scales: 1 minute
and 10 minutes. For each one, we produce time series of binary activations (the mask). A node (user) is active at a given time step $t$ if the corresponding user has retweeted within time steps $t$ and $t+1$. We only use retweet information because the available dataset only provides retweets, replies, mentions but not tweets. We do not include replies and mentions to be able to compare with [14] which focuses more on retweets than the other activities.

Figure 7: Graph of the largest activated component from the 10 minute sampling, colored by community. The graph layout has been generated using open ORD [26]. The different colors correspond to the different communities. However, the set of colors is limited and different communities may have the same color. Edges color correspond to the color of the source nodes color.

The analysis of the activated components The activated components are directed layered graphs of activity. Each node is a user active at a particular time-step. Only inter-layer edges have been kept for the construction the components.

On table[1] the largest activated components extracted from the causal multilayer graph are shown. The largest one appear on the 4th of July, the announcement day and cover most of the day. Moreover, the number of users involved is extremely large. It shows how the information has spread over the network. It starts before 8:00 a.m. as rumors were spreading before and increased as the announcement time was approaching. The other components are at least one order of magnitude smaller and last one to three hours each. They are distributed between the 2nd and 5th days of July. Bursts of activity may appear indistinctly during day or night as the event is of worldwide importance. The geo-localization tags are not available in the dataset and we could not verify whether components involves particular countries or regions.
Figure 8: **Graph of the largest activated component from the 10 minute sampling, colored according to time slices.** The different color corresponds to the different time slices. However, the set of colors is limited and different time-slices may have the same color. Edges colors correspond to the color of the source nodes they connect to.

The largest activated components obtained by the one minute sampling are displayed on Table 2. The largest components appear on the 4th as expected. The first component, 4704 different users are active and the activity spans over 114 minutes. This frenetic activity appears just before the official announcement. The second largest component is the consequence of the announcement, it starts at 8:01 a.m. and the information propagates quickly until 8:18 a.m. Most of the largest components last around 10-15 minutes and involves around a hundred users. The largest components take place on the 4th, where the activity is so frenetic that information can be retweeted in less than one minute and propagates to tens of users in only 10 minutes.

The number of layers (time spread) of the two different sampling rates are similar: the largest component in each is around 100 layers long and the others are around 15. The number of layers is the number of time-steps, so components of the 10 min sampling rate last 10 times longer than the 1 min sampling rate. The social spread (number of different users in the component) is less than 10 times larger between Table 1 to Table 2. Except for the largest component, it is only multiplied by 2 (roughly). It tends to advocate for a community-like activity where information is retweeted within communities, limiting the number of users involved.

In order to better understand the rumor spreading dynamics, it is interesting to focus on the largest activated component on the 10 min sampling rate, described on the first line of Table 1. This component cover the official announcement of the Higgs boson discovery. The activated component is a graph and we have run a community detection algorithm (Louvain method [10]) on it. The
| Nb of nodes | Nb of layers | Social spread | Start day and time | End    |
|-------------|-------------|---------------|--------------------|--------|
| 55037       | 108         | 36800         | 04, 03:10          | 04, 21:00 |
| 357         | 15          | 324           | 03, 17:40          | 03, 20:00 |
| 299         | 12          | 277           | 05, 11:30          | 05, 13:20 |
| 254         | 12          | 231           | 05, 05:00          | 05, 06:50 |
| 244         | 9           | 235           | 05, 00:00          | 05, 01:20 |
| 232         | 12          | 212           | 02, 16:20          | 02, 18:10 |
| 200         | 14          | 163           | 03, 21:00          | 03, 23:10 |
| 169         | 5           | 107           | 04, 14:40          | 04, 15:20 |
| 166         | 9           | 160           | 05, 10:30          | 05, 11:50 |
| 142         | 9           | 128           | 04, 20:50          | 04, 22:10 |

Table 1: Largest activated components with their size and time of appearance for the 10 minutes time-steps.

| Nb of nodes | Nb of layers | Social spread | Start day and time | End    |
|-------------|-------------|---------------|--------------------|--------|
| 8593        | 114         | 4704          | 04, 05:17          | 04, 07:10 |
| 255         | 18          | 214           | 04, 08:01          | 04, 08:18 |
| 216         | 15          | 164           | 04, 04:51          | 04, 05:05 |
| 151         | 9           | 130           | 04, 05:09          | 04, 05:17 |
| 142         | 5           | 133           | 04, 13:41          | 04, 13:45 |
| 100         | 9           | 83            | 04, 07:13          | 04, 07:21 |
| 98          | 15          | 98            | 04, 13:22          | 04, 13:36 |
| 95          | 15          | 75            | 04, 15:08          | 04, 15:22 |
| 95          | 14          | 95            | 02, 19:57          | 02, 20:10 |
| 93          | 16          | 91            | 04, 14:12          | 04, 14:27 |

Table 2: Largest activated components with their size and time of appearance for the 1 minute time steps.

The modularity value of the graph is 0.82, showing evidence that it is made of well separated communities. The algorithm provides 56 different communities\(^2\) with 4 main ones containing a large number of nodes (11%, 9%, 7%, 7% of the total number of nodes respectively). A plot of the graph with the different communities is shown on Fig. 7. Large well connected communities can be seen. The communities are elongated as connections in the graph are only between successive layers: 2 nodes in the same community may not be directly connected but are connected through their neighbors at successive time steps. We recall that each node of the graph is a user active at a particular time-step. Hence several nodes can correspond to the same user and communities in the activated component do not necessarily represent communities in the twitter graph of followers. Communities in the activated components are sets of connected users active within a particular time interval.

A part of the dynamical activity of the component is plotted on Fig. 8. It is the same graph but each color corresponds to a time step within the activated component (a layer) instead of a community. As expected the elongated communities possesses several colors, having connections on several successive layers. Comparing with Fig. 7 some communities are active at the same time and some of them activate in a successive manner. The spreading mechanism made of bursts occurring between users seems to be the same as the one occurring on a larger scale, between communities: bursts of community activity are appearing within the activated component. This definitely deserves further investigation.

\(^{2}\)Due to anonymization of the data and the non availability of the geo-localization data we were not able to check if these communities correspond to particular location or group of people.
5.2 Visualizing crowd movements in a train station

Our second application visualizes and quantifies pedestrian movements in the train station of Lausanne, Switzerland. The dataset, gathered by [2], is composed of 42 millions points (x, y, t, pedestrian id) tracked by a connected network of cameras placed in the two main corridors of the station. The data collection spans over 2 weeks at the most crowded hours. Our study aims at characterizing repeated congestion patterns over time and understand their dynamics. Far from a purely academic interest, the Swiss Federal Railways are seeking to expand the corridors in prevision of an ever increasing traffic. This work constitutes a first step in the visualization of repeated crowd movements in the station.

The graph. Before the creation of the causal multilayer graph, a spatial graph of connected regions onto which the crowd moves needs to be created. The original coordinates of the pedestrians are too precise to be used as is (8 millions unique pair of points) since we are interested in global trends in the dataset. Instead of manually defining zones of interests, we propose an automated method to create a graph of regions of interest by clustering (via batch k-means) a random subset (in our case 30%) of the points. The nodes are centroids of the cluster and the edges are created by linking adjacent clusters. This graph is considerably better than a regular grid because it is fine-grained on congested zones and coarser where the traffic is less dense. This technique is very similar to the concept of superpixels used in some computer vision applications [1], see Fig. 9.

![Image](image-url)

Figure 9: Segmentation of the west corridor into regions of interests according to the density of pedestrians. On the corridor map, the 8 black boxes labeled as ‘P’ represent the platforms to access the trains. The letter ‘E’ stands for the two exits and ‘S’ is a shop. Each colored polygon is a small area of the corridor, represented by a single node on the graph. The main access to the station, on the right, is busier than the one on the left, giving smaller polygons.

The signal and mask. We naturally chose the pedestrian’s density in the station as the signal. The sampling rate is 5 seconds and the value at each time step is the number pedestrian that have crossed the area within this 5s duration. We normalize the signal by subtracting the mean congestion value over time at each node. This allows to track the crowd movement even in busy areas of the station where the signal could “saturate”. Eventually, we create the congestion mask by empirically setting a threshold to cast the traffic activity in a congested / non congested state.

Analysis of the activated components. The causal multilayer graph is made of thousands of components naturally split by the activation threshold. The basic statistics of each component, such as width and spatial spread, are useful to quantify the impact of a event (e.g a train departure) in the whole station. The width gives the duration of an event and the spatial spread the number of regions impacted by it. Note that the congestion event is tracked over time, the width and spatial spread take into account all of the congestion pattern even if it moves along the corridors.

The k-means clustering of thousands of components of similar shape creates average activated components correlated with the departures and arrivals of trains in the station. Two examples are given on Fig. 10 (left and right). Each one represents an average dynamic trajectory of pedestrians.
inside the corridors. On these average components, a node represents an activated (crowded) area of the corridor, the node color represents the time dimension, from the start of the component (blue) to its end (red). The two examples display mean congestion patterns evolving in time as the crowd moves along the corridor. It demonstrates the ability of our causal multilayer graph model to track the crowd movement and extract relevant information from it.

**Figure 10:** Two examples of average activated components plotted on the same figure. Each circle represents a crowded area at some point in time. The color scale gives the time of appearance of a congestion at a given location. It ranges from dark blue (start of the congestion) to yellow, then to red for the end of the component. On the right, an average activated component giving a repeated congestion pattern between one platform and the main exit. At the beginning of the component, pedestrians go on the platform (blue dots under the yellow), just before a train departure. Then the crowd coming out of the train can be tracked, from yellow (earlier time) near the platform access to red (later time) as it approaches the main exit. The second activated component on the left shows the crowd coming both from the left entrance and from a shopping store and entering the main corridor. This component ends inside the corridor as the crowd then spreads to different platforms and reduce the congestion rate below the threshold.

The average activated components (associated to each cluster) can be used as a basis for the analysis of the most recurrent events in the station. It provides information such as the most frequently crowded areas, the largest congestion in time, in space, highlighting any traffic "bottleneck”. In addition, unusual events (delay, accident, etc.) in the station causing a congestion can be detected by comparing its activated component to the average one of each cluster. A large dissimilarity with all the clusters is considered abnormal and may require a human intervention in the station.

### 5.3 Analyzing thousands of collaborative audio playlists

Up to now, the presented applications have extracted and analyzed activated components from causal processes modeled as time-series on a graph. In this application, we use the causal multilayer graph approach to analyze another kind of dataset, without a given graph, without time series but with causal relations. We first create a graph from the data and model the causal process as vectors associated to the graph nodes. This is an example showing the wide range of applications that can be modeled by our method. In this section, the focus is on the method (how to construct a model from causal data, how it scales) and not on the results, there are probably better ways to build recommender systems for music.

The Art of the Mix dataset originally crawled by McFee in [28] regroups 101,343 collaborative mixes from 1998 to 2011. A mix is a special kind of playlist where songs are chosen to have meaningful transitions between them. In other words, songs are put in a specific order in a mix because it exists a causal relationship between them. The position inside the mix is also important as a mix possesses a global evolution. The types of songs (their mood, energy, degree of danceability) in the first part of it are often different from the ones in the middle or at the end.

In this application, our objective is to show how to use activated components to create a playlist...
A recommender system based on what Bonnin et al. describe as frequent pattern mining in [11]. For illustration purposes we visualize groups of common listening patterns of users.

**The graph.** The graph $G$ is constructed by doing the union of all the playlists in the dataset. An edge between two songs only exists if at least one playlist contains this particular sequence of songs. The number of nodes, the number of unique songs matched by artist and title, is over 159,000.

**The signal and mask.** As a signal on the graph, we use the relative popularity of a song in the dataset. To compute it, we count how many times a particular song has been placed at a particular position for all playlists. For example, a song A has been placed 3 times in first position of a playlist and 5 times at the third position. Thus, in this example, the vector on the node A has 2 non-zero values (3, 0, 5, 0, ...). The number of entries of each vector is equal to the number of songs in the longest playlist. Similarly to the previous application, we normalize the signal by subtracting the mean popularity for each song. This allows the activation of less popular songs resulting in coherent activated components of very popular songs linked together by less popular ones through causal edges. This coherence and the difference of popularity make activated components good candidates to create a playlist recommender system.

**Playlist recommender system.** We propose an algorithm to generate playlists based on music “moods” using average activated components obtained by $k$-means clustering. As an outcome of our method we will see that different music moods are associated, in a totally unsupervised manner, to the clusters. Moods are “Electroish”, “Metallic”, “Rocky” (see Fig. 12 for example) and can be viewed as a meta-genre of music regrouping related music genres such as Rock, Indie, Alternative, etc.

We extract “activated components” of songs from the causal multilayer graph by thresholding the normalized popularity vector (keeping the largest peaks of appearance of songs in playlists). Each activated component is a group of songs fitting nicely together and respecting a precise order within the playlists. These activated components are then clustered together using $k$-means clustering. Each cluster represent now a music ”mood”. The value of $k$ defines the diversity of the mood, a small $k$ creates a small number of clusters of large clusters having a more diverse set of songs. We can label each cluster with a mood a posteriori by analysis the music genres present within the clusters.

Once the mood has been chosen by a user, the algorithm selects the average activated component associated to the mood. To generate a playlist, a seed song, the first of the playlist, is selected in the first layer of the component. In its simplest form, the selection is done at random. However, many criteria such as user history, ratings, time since last played, can be used to select a starting point. Then, the rest of the playlist is constructed by doing a random walk on the causal edges of the component. The familiarity versus discovery ratio can be tuned by modifying the edge weights according to the popularity of each node. The random walker would have more or less chances to reach a popular song depending on the user’s will.

A good playlister should alternate between familiarity, discovery and smoothness in the transitions between songs [27]. An average activated component is a coherent weighted subgraph of songs, where each node and edge are weighted by their likelihood of appearance at that particular position. The most popular songs, appearing in many activated components, have a large weight, filling the contract for the familiarity part. Less popular songs will also be clustered together giving choice for the discovery part. Finally, the smoothness of the transitions is guaranteed by the graph $G$: each song to song transition has been created by a human and is appealing to at least one of them.

Keeping the original ordering of songs in a playlist (successive positions of several songs, not just 2) has been shown to be crucial when designing recommender systems as shown in [27] and [11]. Following the causal edges of an average activated component takes into account the ordering of several successive songs with their positions within playlists, unlike a random walk on the graph $G$ which considers only one to one coupling between songs.
Figure 11: **Average activated component of the cluster grouping most of the Jazz songs.** For a clearer visualization, we have discarded nodes with a low probability of appearance. The horizontal axis represent the layers: the position in the playlist (limited to the 8 first layers). The nodes and edges are colored from light to dark according to their weights, i.e. their likelihood of appearance (larger is darker).

Figure 12: **An average activation component for Death Metal.** While the genre diversity is a bit higher than in the Jazz example, all genres belong to the same meta-genre: Metal.

Figure 13: **An average activation component for Hip-Hop.** The Heavy Metal nodes are not outliers but are indeed connected to Hip-Hop with songs from famous artists such as Korn, Limp Bizkit, Public Enemy, etc.

**Visualization** The activated components can also be used for exploring and visualizing the dataset. Since each activated component has a large spatial spread, we group songs on each layer by genre as it drastically reduces the dimensionality and exhibits interesting insights on how users create playlists. Note that genres are here to validate the methodology and have not been used to cluster the activated components together. Like in the previous applications the method is completely unsupervised.

While the dataset is biased towards Rock, Alternative and Indie (more than 40% of all the songs),
An average activated component for the pair Rock - Alternative. The diversity of the genres is much higher than in previous examples but stays coherent.

The spatial spread of the different average activated components presented in Fig. 11, Fig. 12, Fig. 13, Fig. 14. As one could expect, popular genres have a bigger spatial spread and are easily mixed with other genres.

the clustering still achieves to extract relatively pure patterns of related genres, or music “moods”, as it shown in Figs 11, 12, 13. As music experts could have expected, songs of popular genres are more volatile: they can easily be mixed with other genres and have a higher spatial spread, see Fig 14 and 15. On the contrary, songs of “connoisseur genres” such as Metal, Jazz, Hip-Hop or Classical stay clustered in their universe.

6 Conclusion

In addition to the applications developed here, we take part in an ongoing work aiming at using the causal multilayer graph for the analysis of the brain activity. As stated in the introduction, the analysis of dynamical activity within the brain is the object of tremendous interest. The cortex is modeled as a graph whose vertices are small pre-defined regions highly inter-connected in an anatomical way [32, 33]. The brain activity can be recorded over time (for example using functional MRI) giving a time-series
associated to each brain region. Particular activity patterns called “resting state networks” appear regularly during rest [12]. These patterns made of several brain regions are dynamic, evolving both spatially and temporally. As in the transportation problem, we can detect these repeated patterns and record the full activation “path” in the network over time.

From a general point of view, we have presented a new approach to extract and analyze sparse repeated patterns involving a graph and a causal process. It is based on a new graph model called the causal multilayer graph that account for causality in events appearing in graph structured datasets. This kind of model has demonstrated its efficiency in 3 different real world applications and looks promising for a wide range of other problems such as the spreading of epidemic outbreaks, social network activity, brain EEG recordings or any type of sensor networks. Applications where time series have been recorded on the vertices of a network are numerous, present in many fields of science such as engineering, social, biological, physical or computational science and keep increasing with the actual data deluge. Besides, our method can also be applied to problems actually modeled as temporal networks, allowing a different approach and an additional degree of model complexity.

This novel method possesses two important advantages for applications, especially large scale ones: i) the construction of the causal multilayer graph, the extraction of the activated components and their analysis are computationally efficient and scale gracefully with the dataset size ii) the proposed detection of patterns is robust to noise. The Higgs dataset is larger than the usual toy models used in research and our method handles it without difficulties. This is a first step towards larger datasets and the world of “Big Data” foreseen in future work.

On top of the adaptability of the generalized version of the causal multilayer graph, additional flexibility can be brought to the method. The clustering of repeated patterns rely on the \( k \)-means algorithm as it is driven by computational efficiency at large scale. However, depending on the distribution of the data, different clustering algorithms such as hierarchical clustering, Gaussian Mixtures or DBSCAN [15] could be used. Moreover, any machine learning algorithm can be used for the classification of the activation components by using the feature vectors associated to them. The Graphlab and Spark software both contain machine learning methods that can replace \( k \)-means without loosing any computational efficiency. One can also introduce additional steps in the process. For example, if large activated components are present, a community detection algorithm such as the one presented in [10] could be used to cut them into smaller pieces before clustering them.

References

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and Sabine Süssstrunk. SLIC Superpixels Compared to State-of-the-Art Superpixel Methods. IEEE Transactions on Pattern Analysis and Machine Intelligence, 34(11):2274–2282, nov 2012.

[2] Alexandre Alahi, Vignesh Ramanathan, and Li Fei-Fei. Socially-Aware Large-Scale Crowd Forecasting. In 2014 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, jun 2014.

[3] Sean Eron Anderson. Bit twiddling hacks. http://graphics.stanford.edu/~seander/bithacks.html, 2005.

[4] Per Bak, Chao Tang, and Kurt Wiesenfeld. Self-organized criticality: An explanation of the 1/f noise. Physical review letters, 59(4):381, 1987.

[5] Albert-Laszlo Barabasi. The origin of bursts and heavy tails in human dynamics. Nature, 435(7039):207–211, 2005.

[6] John M Beggs and Dietmar Plenz. Neuronal avalanches in neocortical circuits. The Journal of neuroscience, 23(35):11167–11177, 2003.
[7] John M Beggs and Nicholas Timme. Being critical of criticality in the brain. *Frontiers in physiology*, 3, 2012.

[8] Kirell Benzi. Code repository for the analysis of the Higgs boson dataset. https://github.com/epfl-lts2/higgs, 2015.

[9] Kirell Benzi. Code repository for the causal multilayer graph. https://github.com/kikohs/sptgraph, 2015.

[10] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *J. Stat. Mech.*, 2008(10):P10008, oct 2008.

[11] Geoffray Bonnin and Dietmar Jannach. Automated Generation of Music Playlists: Survey and Experiments. *ACM Computing Surveys (CSUR)*, 47(2):1–35, nov 2014.

[12] D. M. Cole, S. M. Smith, and Ch. F. Beckmann. Advances and pitfalls in the analysis and interpretation of resting-state FMRI data. *Front. Syst. Neurosci.*, 2010.

[13] Vittoria Colizza, Alain Barrat, Marc Barthélemy, and Alessandro Vespignani. The role of the airline transportation network in the prediction and predictability of global epidemics. *Proceedings of the National Academy of Sciences of the United States of America*, 103(7):2015–2020, 2006.

[14] Manlio De Domenico, Antonio Lima, Paul Mougel, and Mirco Musolesi. The anatomy of a scientific rumor. *Scientific reports*, 3, 2013.

[15] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[16] Sergio Gomez, Albert Diaz-Guilera, Jesus Gomez-Gardeñes, Conrad J Perez-Vicente, Yamir Moreno, and Alex Arenas. Diffusion dynamics on multiplex networks. *Physical review letters*, 110(2):028701, 2013.

[17] Petter Holme and Jari Saramäki. Temporal networks. *Physics Reports*, 519(3):97–125, oct 2012.

[18] U. Kang, Charalampos E. Tsourakakis, and Christos Faloutsos. PEGASUS: A Peta-Scale Graph Mining System Implementation and Observations. In 2009 Ninth IEEE International Conference on Data Mining. IEEE, dec 2009.

[19] Márton Karsai, Kimmo Kaski, Albert-László Barabási, and János Kertész. Universal features of correlated bursty behaviour. *Scientific reports*, 2, 2012.

[20] Matt J Keeling and Ken TD Eames. Networks and epidemic models. *Journal of the Royal Society Interface*, 2(4):295–307, 2005.

[21] M. Kivela, A. Arenas, M. Barthelemy, J. P. Gleeson, Y. Moreno, and M. A. Porter. Multilayer networks. *Journal of Complex Networks*, 2(3):203–271, jul 2014.

[22] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.

[23] S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, mar 1982.

[24] Yucheng Low, Danny Bickson, Joseph Gonzalez, Carlos Guestrin, Aapo Kyrola, and Joseph M. Hellerstein. Distributed GraphLab. *Proceedings of the VLDB Endowment*, 5(8):716–727, apr 2012.
[25] M. Donald MacLaren. The Art of Computer Programming, Volume 2: Seminumerical Algorithms (Donald E. Knuth). *SIAM Rev.*, 12(2):306–308, apr 1970.

[26] Shawn Martin, W Michael Brown, Richard Klavans, and Kevin W Boyack. Openord: an open-source toolbox for large graph layout. In *IS&T/SPIE Electronic Imaging*, pages 786806–786806. International Society for Optics and Photonics, 2011.

[27] Brian McFee and Gert RG Lanckriet. The Natural Language of Playlists. In *ISMIR*, pages 537–542, 2011.

[28] Brian McFee and Gert RG Lanckriet. Hypergraph Models of Playlist Dialects. In *ISMIR*, pages 343–348. Citeseer, 2012.

[29] Peter J Mucha, Thomas Richardson, Kevin Macon, Mason A Porter, and Jukka-Pekka Onnela. Community structure in time-dependent, multiscale, and multiplex networks. *Science*, 328(5980):876–878, 2010.

[30] Gert Sabidussi. Graph multiplication. *Mathematische Zeitschrift*, 72(1):446–457, 1959.

[31] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Process. Mag.*, 30(3):83–98, may 2013.

[32] Olaf Sporns. *Networks of the Brain*. MIT press, 2011.

[33] Olaf Sporns. Contributions and challenges for network models in cognitive neuroscience. *Nature neuroscience*, 17(5):652–660, 2014.

[34] Olaf Sporns, Dante R. Chialvo, Marcus Kaiser, and Claus C. Hilgetag. Organization, development and function of complex brain networks. *Trends in Cognitive Sciences*, 8(9):418 – 425, 2004.