Visualising the Evolution of Dynamic Communities in Social Networks using Timelines

Hugo Hromic and Conor Hayes
Insight Centre for Data Analytics @ NUI Galway
{first.last}@insight-centre.org
http://www.insight-centre.org/

Abstract. Real-world social networks from a variety of domains can naturally be modelled as dynamic graphs. However, approaches for detecting communities mostly focus on identifying them in static graphs. Researchers often consider the problem of tracking the evolution of communities in dynamic scenarios. In this work we present the Dynamic Community Viewer (DCV), a visualisation tool for tracking the life cycle of communities over time in a dynamic network, where each community is characterised by a series of significant evolutionary events. The DCV is capable of visualising the development of these events at different points in time using browsable timelines. Specifically, it can visualise the birth, death, split, merge, contraction, expansion and any user-defined attribute change (e.g. topics) as evolutionary events for sets of dynamic communities. Our tool is based on an established community tracking model that leverages a community-matching strategy for efficiently identifying and tracking dynamic communities.

Keywords: Social Networks · Dynamic Communities · Visualisation.

1 Introduction

Social network analysis is traditionally focused on the representation of graphs as static networks for specific time steps. This representation is widely adopted for the task of community detection, where the goal is to identify meaningful group structures in these networks. However, representing dynamic sources of data using such static networks can lead to group structures present in short periods of time to be difficult to identify or be completely unnoticed by the community detection approach. Discarding temporal information can lead to loosing the detail of the evolutionary behaviour of these groups.

One of the most interesting aspects of community detection for researchers and decision makers is the ability to visualise how the user communities under study are evolving in time. However, most community finding approaches are unable to generate such view directly, but instead focus on identifying static sets of communities at particular points in time. Fortunately, dynamic user community tracking algorithms exist [8] that can analyse arbitrarily discovered communities between consecutive time steps and identify similarities from a past set of communities to a current updated configuration.

Using the dynamic community tracking algorithm in [8] as a base, we build a visualisation tool capable of extracting eight types of evolutionary events from a set of communities discovered at consecutive time steps: (1) birth, (2) death, (3) split, (4) merge, (5) expansion, (6) contraction, (7) intermittence, and (8) attribute change.
Our proposed workflow can be seen in Figure 1. First, the user chooses a community detection approach suitable for the dataset under study. Then, sets of independent static communities $C_{t,i}$ are extracted for consecutive time steps $t$ using any desired granularity, e.g. hourly or daily. With these static step communities, the user applies the dynamic community tracking approach from Greene et al. [8], called TRACKER, to obtain dynamic timelines that relate step communities to each other in time considering the birth, death, split, merge and intermittence evolutionary events. Finally, the user applies our post-processing algorithm, TRACKER2Vis, to compute the remaining evolutionary events, i.e. expansion, contraction and attribute change, and generate the complete visualisation information necessary for the interactive user interface. It is important to note that our tool is able to visualise any dynamic timeline computed using TRACKER, which mainly utilises the Jaccard similarity metric to compare “communities” across time steps. This characteristic makes their tracking approach highly convenient and flexible. In consequence, any group-like structure from any clustering technique can be used and despite the visualisation being designed towards community analytics, it is not strictly restricted only to this use case.

To achieve all the above functionality, our work proposes three main contributions: (1) we build a post-processing algorithm on top of the work of Greene et al. [8] for extracting evolutionary events from dynamic communities, (2) we propose an additional event type (attribute change) to support visualisation of user-defined aspects such as topics, and (3) we develop a web-based visualisation tool, the Dynamic Community Viewer (DCV), that combines the above contributions into a concrete general purpose tool. The source code\footnote{https://github.com/hhromic/dynamic-community-vis} and an interactive demo of the DCV\footnote{https://uimr.insight-centre.org/dcv/} are both available online.

The rest of this paper is organised as follows. In Section 2 a summary of related work is presented. In Section 3 we describe the evolutionary events we consider, how we extract them and detail our visualisation approach. Finally, in Section 4, we present conclusions and potential future directions for our tool.
2 Related Work

Finding communities in static graphs is a well-known problem in the literature [7, 13]. In recent years researchers have been more focused on the dynamic aspects of communities motivated by modern real-world social media. For example, extensions to the popular clique percolation method for evolutionary networks has been proposed [12]. This extension involves applying community detection to joint graphs of adjacent time steps. A similar life cycle model was proposed in [16], where the dynamic community finding task is formulated as a graph colouring problem. Another example is in [1], where a community event identification approach is described based on a matching strategy across time steps. Besides social communities, the more general problem of identifying clusters in dynamic networks has been studied extensively, e.g. [3, 4, 9, 5].

The visualisation and exploration of such dynamic communities has been presented extensively in previous work [17]. For example, the transition between time steps can be visualised using straight links in timelines [14] or splines [15]. The former is one of the few examples with an explicit sorting strategy. Moreover, communities or vertices have been depicted using changing nodes [6, 11]. However, to the best of our knowledge, no other work has previously used event drops in timelines for representation.

3 Tracking and Visualising Dynamic Communities

We present our visualisation approach as an extension to the dynamic community tracking algorithm proposed by Greene et al. in [8]. In their work, the authors propose a generalisation for the task of dynamic community finding focused on the life cycle of user communities in dynamic networks. In this paper, we refer to it as Tracker.

A dynamic network is represented as a set of time step graphs that provide snapshots of the nodes and edges of the network at successive intervals. The dynamic community finding task is then defined as identifying a set of \( k \) dynamic communities \( D = \{D_1, \ldots, D_k\} \) that are present in the network across one or more consecutive time steps with any desired granularity, e.g. every minute, hour or day. Furthermore, a step community is defined as a static community identified at a particular time step, representing a specific observation of a dynamic community at a given point in time. It is important to note here that the identification of such step communities is not restricted to any particular community detection method or algorithm. The definition of communities can be of any nature desired and is solely dependant on the chosen static community identification approach. The set of \( k_t \) step communities identified at each time \( t \) is denoted as \( C_t = \{C_{t,1}, \ldots, C_{t,k_t}\} \). Therefore, each dynamic community \( D_i \) can be represented by a timeline of its step communities, ordered by time.

We are interested in visualising the evolution of of this set of dynamic communities \( D \). For this we propose a two-fold approach. First, we compute the dynamic community timelines for sets of step communities using Tracker from Greene et al. Then, we post-process these timelines and compute explicit community evolutionary events (e.g. splits, merges, contractions and expansions) and their visualisation meta-data, e.g. the source/target steps for a split event or the grow ratio for an expansion event.
We aim for an intuitive visual representation of all the available evolutionary information at a glance and therefore we propose to use a modified *event drops* display\(^3\). In this type of visualisation, time is represented in the horizontal axis and dynamic communities \(D_i\) are stacked in lanes, with their step communities \(C_t\) represented using circles (drops) at each discrete time step \(t\) horizontally. This display can be zoomed in or out along the horizontal axis (time), and drops fuse or separate when they become close or distant enough. Originally, the event drops display is only capable of single disconnected drops per event. Thus, we extend it to include connecting lines between drops that represent the continuation, splitting, merging, contraction and expansion evolutionary events. An example view of our proposed visualisation can be seen in Figure 2.

\[ \text{Fig. 2. Example Visualisation of Dynamic Timelines for Synthetic Communities.} \]

In the Figure, four dynamic community timelines can be seen named from \(M_1\) to \(M_4\) that demonstrate all the considered evolutionary events. For each timeline \(M_i\), static step communities exist with a particular attribute represented by a colour at different time steps. An example of an attribute can be a dominant topic label for each step communities. These attributes can change in time and therefore change colour. The timeline \(M_1\) shows an intermittence. \(M_2\) shows an expansion from the fourth to the fifth step, represented by a green line on top. \(M_3\) shows both a contraction from step 2 to step 3, represented by a red line on top, and a split to \(M_4\) in step 2, represented by a dashed line from \(M_3\) to \(M_4\). Lastly, \(M_4\) shows a merge back to \(M_2\) also represented by a dashed line between both. In all cases, the continuation lines adopt the same colour of the attribute for the next step community in the same dynamic community.

3.1 Extracting Evolutionary Events

In the dynamic community finding literature (e.g. \([12, 16, 1]\)) there is a consensus on the fundamental events that can be used to characterise the evolution of dynamic communities. Using the same notation as in the previous Section, we consider the following evolutionary events for visualisation:

\[^{3}\text{https://github.com/marmelab/EventDrops}\]
**Birth** is the emergence of a step community $C_{t,i}$ observed at time $t$ for which there is no corresponding dynamic community $D_i \in \mathbb{D}$.

**Death** is the dissolution of a dynamic community $D_i$ after not observing at least $d$ corresponding consecutive step communities for it in $C_t$. With $d$ being user-defined.

**Merging** occurs when two different dynamic communities $(D_i, D_j)$ observed at time $t = 1$ share the same single step community at time $t$. Consequently, the pair starts sharing common timelines.

**Splitting** occurs when a single dynamic community $D_i$ at time $t = 1$ observes two distinct step communities at time $t$. A branching occurs and a new dynamic community $D_j$ emerges at time $t$ with a new independent timeline.

**Contraction and Expansion** are the change in size, i.e. number of members, of a dynamic community $D_i$ from time $t = 1$ to $t$, that is under or above a user-defined threshold $\alpha$, e.g. $\alpha > 10\%$ for an expansion.

**Intermittence** occurs when a dynamic community $D_i$ is not observed from time $t = 1$ to $t$ but re-appears in a future user-defined time $t + n$ before its death event.

**Attribute Change** is the change of a user-defined attribute for a dynamic community $D_i$, e.g. its topic, from time $t = 1$ to $t$. This event was not previously considered by Greene et al. in their original work [8] and is an addition from ours.

The Tracker algorithm computes the timelines for dynamic communities $\mathbb{D}$ from an input set of step communities $\mathbb{C}_t$, however it does not explicitly identify the above events in these timelines. Therefore, we present Tracker2Vis, a post-processing algorithm that uncovers these evolutionary events from the plain Tracker timelines.

To extract the evolutionary events, Tracker2Vis uses both, the original input step communities and the output timelines generated by Tracker, to compute a set of $k'$ events $E = \{E_1, \ldots, E_k\}$ for the dynamic communities $\mathbb{D}$. An event $E_i$ is composed of a type, e.g. birth or expansion, and depending on this type, a set of associated event meta-data: for birth, death and intermittence, the step community $C_{t,i}$ of the event; and for merge, split, contraction, expansion and attribute change, the source and target step communities $(C_{t,i}, C_{t+1,i})$ of the event. For the contraction and expansion types we also record the shrink and grow ratio $r > \alpha$, and for attribute change its new value.

To compute the set of events $E$, we proceed as follows. From the step communities, we can trivially extract the expansion and contraction evolutionary events by comparing every adjacent step communities $C_{t,i} \in \mathbb{C}_t$ and $C_{t+1,i} \in \mathbb{C}_{t+1}$ of each dynamic community $D_i \in \mathbb{D}$, measuring their sizes against a predefined threshold $\alpha$. Then we record the source and target step communities for which this size changed above $\alpha$.

The rest of the evolutionary events are extracted from the timelines computed by Tracker. First, we find split events by searching dynamic communities pair-wise $(D_i, D_j) \in \mathbb{D}$ for those with common first step community. Then for each found pair, we iterate their common step communities until we find a distinct step, thus signalling a divergence in their timelines. We record the source and target step communities where the divergence happens and proceed to remove all the initial step communities of the dynamic community $D_j$ until this point. After this operation, it is possible that the timelines now contain duplicated dynamic communities, therefore we find and remove these duplicates. To find merge events, we follow a similar approach but in reverse order of step communities iteration, i.e. from back to front. At this point, it is now possible that some previously found split events are left orphaned, i.e. finding the merge events removed the step communities they referenced. Therefore we further scan the built split
events and remove those that are no longer relevant. Finally, we proceed to find the birth, death and intermittence events by iterating every single dynamic community $D_i \in \mathcal{D}$. For birth and death, we search for the first and last step communities respectively and record them as the event meta-data. For intermittence, we scan the timelines for non-consecutive step communities that are not the last and record this as the event.

3.2 Visualising Timelines and Evolutionary Events

After extracting a rich set of evolutionary events from dynamic community timelines, we can now visualise them using our proposed modified event drops display.

Summarising the example in Figure 2, the expansion and extraction events are visualised using green and red lines respectively between the two involved step communities $C_{i,t}$ and $C_{i,t+1}$. If the users of a static step community $C_{i,t}$ split, this is represented by a dashed line from their original community to a new dynamic community in a different lane. Likewise, if the members of a step community $C_{i,t}$ in a dynamic community $D_i$ join another existing dynamic community $D_j$ in a consecutive time step $C_{j,t+1}$, the merge is represented by a dashed line from $C_{i,t}$ to $C_{j,t+1}$. Lastly, every community can contain an arbitrary user-defined attribute whose values are represented by a colour mapping in both, the circle (drops) representation and the continuation lines. This provides a clear visualisation of attribute change in the dynamic communities.

We further illustrate our proposed tool using a real-world use-case. In the context of online social awareness and recommendation, in previous work [2] we captured Twitter data related to TV programmes being broadcasted live in Ireland by the national television broadcaster, RTE. We then constructed sets of step communities $C_t$ discovered using the OSLOM algorithm [10] which is based on the modularity of the groups. We applied the Tracker algorithm and our Tracker2Vis post-processing to obtain a set of dynamic community timelines $\mathcal{D}$ and their set of evolutionary events $\mathcal{E}$. For the user-defined community attribute, we used a topic label defined as the name of the TV programme most frequently referenced by the community members. The extraction of these topic labels is not restricted to any particular method. For example, topics could also be defined from the top three most frequent hashtags in the community. Figure 3 shows an example visualisation of the DCV tool applied to this data, where only dynamic communities with at least three step communities are displayed for clarity.

The DCV shows that some dynamic communities, such as $M_3$, $M_27$ and $M_{29}$ are steady and stable in time, while others such as $M_{12}$, $M_{14}$, $M_{21}$ and $M_{26}$ result in quite intermittent dynamic communities. After manual examination of these cases, we discovered that they were very short-lived communities, as identified by the OSLOM algorithm, however they tend to re-emerge later as new dynamic communities. This observation is recurrent in other dynamic communities in the dataset and is not a desired behaviour since it can be misleading for a decision maker, who for example could conclude that these short-lived communities are not performing well, while in fact they are just not being detected as the same community. While the OSLOM algorithm is reportedly a good community detection method for classic social media datasets [10], for the case of Twitter data it seems to be less effective.

Overall, examination using our visualisation tool then suggests to the analyst that an alternative detection method might be necessary. In particular, one that is more robust to the highly dynamic nature of the microblogging platform.
4 Conclusions

In this work we presented the Dynamic Community Viewer (DCV), a tool for visualisation of dynamic community timelines and their evolutionary events. To achieve this, we (1) built a post-processing algorithm, TRACKER2VIS, on top of the dynamic communities tracking algorithm TRACKER [8], for extracting common evolutionary events from the literature such as birth, death, split, merge, contraction, expansion, and (2) proposed an additional event type attribute change to further support the visualisation of user-defined aspects of the communities such as their topic. We believe that this visualisation tool is a valuable resource for researchers, decision makers and professionals in the field of dynamic networks and community analysis. We also showcased an example use-case from our past work in the Twitter microblogging scenario.

For future work, we consider two main lines for improvement: (1) a method for sorting and/or grouping the dynamic timelines, e.g. one could wish to see dynamic communities that split/merge more often together or dynamic communities with similar topics sorted by usage frequency, and (2) real-time updating of the visualisation. While the latter is an ambitious goal, it can be interesting to adapt our visualisation for automatically refreshing according to live step communities being identified on the fly.

References


