Clustering Dynamic Graphs using Time and Text Content

Timothy La Fond, Eisha Nathan, Hannah Nyholm, and Van Henson

Lawrence Livermore National Laboratory, 7000 East Ave, Livermore, CA 94550, USA
{lafond1,nathan4,nyholm7,henson5}@llnl.gov

Abstract. Modern datasets can often be represented as dynamic graphs; for example, email communications, online forums, and network traffic all fall under this umbrella. As these datasets can be large in size, graph clustering techniques are frequently used to organize the data into manageable chunks. Clustering communications is particularly valuable for identifying meaningful interactions, tracking evolving discussions, and detecting behavioral patterns. However, past algorithms have primarily focused on using the graph topology and time dynamics of the data when clustering, organizing the edges into clusters representing coherent spans of activity – a short conversation between individuals, for example. This ignores other features in the graph that may be useful for clustering such as the text content of the messages being sent. A conversation should have a coherent topic of discussion, therefore clustering edges according to topical similarity in the text can lead to clusters more naturally aligned with human conversations. In this paper we introduce a new dynamic clustering algorithm which takes into account additional metadata alongside the topology and time dynamics, and demonstrate how our approach can find clusters that are coherent in terms of both time and text content, leading to performance improvements of up to 0.28 greater AUC on thread prediction tasks.

1 Introduction

Dynamic graphs are a form of data where each edge represents some interaction between two entities that occurred at a particular point in time. Such datasets could include emails and/or texts, forum posts, or computer network traffic. As these datasets can be extremely large, especially if they cover a long period of time, it is helpful to apply a clustering algorithm which can organize the messages into manageable clusters. Traditional graph clustering has been utilized on a diverse range of applications: to detect fraudulent activity in financial datasets [17], to identify emerging trends or misinformation campaigns in social media data [19], or to detect cyber attacks in computer network traffic data [11].

Although traditional graph clustering has many uses, when blindly applied to data with a temporal aspect the clusters often fail to represent the dynamic behavior of the graph. In particular, evolving behavior may be lumped together into clusters that span long ranges of time, obscuring those dynamics. A better approach is to use dynamic graph clustering algorithms which attempt to cluster using both time and topology simultaneously. This yields clusters which occur over a coherent span of time, which can detect dynamic behavior such as cascades of messages triggered by some event or a back-and-forth conversation taking place over a short span of time. The paper by Ostroski et al [12] approaches this problem by forming a line graph where the nodes of the line graph are the edges in the original dynamic graph, and those nodes are connected with edge weights inversely proportional to the time gap between the two edges in the dynamic graph. The logic here is that the line graph represents possible causal influence between edges in the dynamic graph, and quick turnaround times for sending a message after receiving one indicates stronger evidence of a relationship. This forms a hierarchical dendrogram of messages, which the authors then cluster using an algorithm inspired by HDBscan [2] to find clusters which large relative time gaps between them. We will refer to this algorithm as the Time Filtered Line Graph algorithm, or TFLG.

This approach is scalable and produces manageably sized clusters or "message bursts" which represent time spans where communication is happening at roughly the same overall rate. However, these bursts don't necessarily map exactly to true human conversations. Conversations naturally include breaks or pauses and they may be halted then resumed at a later time. This variability in time delays will be interpreted as break points by the TFLG algorithm and the conversations will be broken into multiple clusters. Likewise, if a person is juggling multiple conversations at once TFLG will not be able to distinguish them due to the overlapping time dynamics. Other features of the data must be considered alongside the time dynamics in order to handle these issues.

Our solution to these issues is to make use of the textual information attached to each edge. A conversation is often comprised of messages focusing on one or a handful of coherent topics. Thus clustering messages while taking topic similarity into account will produce bursts which more closely resemble human conversations. This also aids in distinguishing concurrent conversations as long as the topics of those conversations are sufficiently different.

We will build upon the line graph approach set by the TFLG algorithm but allow for local graph features such as text to be used alongside the time dynamics when deciding cluster boundaries. We demonstrate this approach by utilizing the text content as the local feature to create a variant of TFLG that we call `c-TFLG`, but other types of metadata can also be used in this framework. We will show that this `c-TFLG` algorithm is more effective in recovering conversations from dynamic graph data than approaches that only utilize time and topology.

2 Related Work

Apart from the TFLG approach which forms the starting point for this work, a number of other papers tackle the clustering of dynamic graphs [16]. Of these, many use a windowing or time slicing approach where the dynamic graph is broken into multiple static graphs representing fixed blocks of time [3] [8]. The advantage of this approach is that it allows for traditional graph clustering to be applied without modification; however, the selection of the windows significantly affects the output. Other papers factor in neighboring time windows in an attempt to smooth the clustering process and mitigate the effect of time window boundaries [1].

Another approach is to discretize time to form the data into a tensor and then use tensor factor analysis to find correlations between time and graph structure [7]. Like the time slicing approaches this also requires *a priori* decisions about how to discretize the data into meaningful time blocks. In addition, the factorization can be computationally intensive for large datasets.

Some papers use a streaming approach to dynamic graph clustering where initial clusters are updated when new edges arrive in the graph [5] [10]. These algorithms often focus on having quick computation times in order to be used on active streams of data and sometimes sacrifice accuracy for this purpose, making them only appealing in the streaming setting.

Conversation detection has been attempted on threads of email messages using features of the emails like text content [6]. However, the techniques used rely heavily on the subject matter line of those emails, beginning with an initial conversation set consisting of emails with identical subjects and then further refining them using content similarity and/or time dynamics. In data without a field as reliable as a subject line it would be difficult to obtain an appropriate initial clustering.

3 Time Filtered Line Graph Algorithm

3.1 Minimum Spanning Tree and Original Edge Weights

The original TFLG paper [12] presents a method for linking messages in a message stream using a minimum spanning tree (MST) and then clustering the line graph to form communities of messages. TFLG is designed for directed dynamic multigraphs $G(V, E, T)$ where each edge $e \in V \times V$ is a directed edge that also has a time t representing when it occurred. The line graph for G is $L_G(E, F, W)$ with $F \in E \times E$ being the line edges, with the restriction that those edges are directed and pointing in the forward direction of time, i.e. no line edge points from a message to an earlier message. These line edges also have weights $W \in \mathbb{R}^+$ given by function $f(e_1, e_2) = \delta t = |t_2 - t_1|$ where t_1, t_2 are the timestamps of the messages.

The task of organizing the messages into short-duration bursts then becomes equivalent to finding clusters in the line graph with minimal edge weight. This is done by calculating a MST on the line graph, creating a hierarchical clustering dendrogram of that MST, then selecting clusters from the dendrogram that minimize the internal edge weights in a manner similar to HDBScan. The result is an algorithm which clusters messages into coherent groups in time with minimal input parameters, only requiring a minimum valid cluster size M.

4 Local Feature TFLG Framework

We will now introduce a framework to perform TFLG-style dynamic clustering while utilizing local metadata to assist in determining cluster boundaries. First, recognize that an equivalent algorithm to the clustering step in TFLG is to apply a merge criterion

$$F(C_{parent}, \mathbb{C}) = \Lambda_{C_{parent}} > \sum_{C_{child} \in \mathbb{C}} [\Lambda_{C_{child}}] \tag{1}$$

to each non-leaf node and their children, selecting the merged cluster C_{parent} only if the inequality holds true.

We can then control the clustering behavior of the TFLG algorithm by replacing this function with an alternative F^* which includes values other than cluster stabilities. An example of an F^* which considers text content within the possible clusters will be given later in equation 2. This deviates from TFLG's original objective of optimizing the stability function alone in order to accommodate other cluster properties.

Using this clustering decision function allows for consideration of properties that are undefined at the edge level and could not be represented by changing the distance function for the line graph edges. For example, the set of participating individuals shared by two clusters would be undefined at the edge level. Our new framework allows for maximum flexibility when adapting the algorithm to accommodate various graph features. All of these modified TFLG algorithms have HDBScan's minimum cluster size M as an input parameter in addition to their own input parameters. In the initial aggregation step, messages are greedily combined using the shortest time deltas until each initial cluster is at least size M. These initial clusters become the leaves of the hierarchical clustering dendrogram $\mathcal{D}(\mathcal{C}, \mathcal{R})$, which is a rooted tree where the nodes $C \in \mathcal{C}$ represent clusters of graph vertices $C \subset V$. These clusters are related via parent child relationships such that $(C_{parent}, C_{child}) \in \mathcal{R}$ if and only if $C_{child} \subset C_{parent}$. The root of this tree is a cluster containing all vertices of the graph.

The modified algorithmic steps are outlined in Alg. 1 and Alg. 2. The steps to produce hierarchical clustering dendrogram D are identical to the original TFLG paper and will be omitted; the modified algorithms introduced here only change the process of selecting the final clusters from D. The initial cluster selection set S is the set of leaf nodes in D, those initial clusters obtained by greedily merging until each is at least M in size. Then, a recursive selection algorithm is applied to each node C_{parent} in the hierarchical dendrogram which is a parent of some leaf nodes $\mathbb{C}$. If the clustering criterion function $F^*(C_{parent}, \mathbb{C})$ returns true when applied to that node and its children, the children are removed from the selection set and the parent is added to the selection set S (i.e. the clusters represented by the leaves are merged into the cluster represented by the parent). If $F*$ returns false then no merge occurs and the children nodes remain part of the selection set S. This process repeats until no potential merge passes the F^* criterion, at which point the current nodes in S are returned as the final clusters.

Algorithm 1 Select clusters from hierarchical dendrogram

```
INPUT : Dendrogram $D$
OUTPUT : Selected clusters $S$
  Add all leaf nodes in $D$ to $S$
  for each height 1 non-leaf node in $D$ do
      Apply recursive selection algorithm to node
  return $S$
```

Algorithm 2 Recursive selection algorithm

```
INPUT : Node $C_{parent}$ and its children $\mathbb{C}$

  if $F^*(C_{parent}, \mathbb{C})$ is true then
      Remove all clusters in $\mathbb{C}$ from $S$
      Add $C_{parent}$ to $S$
      Apply recursive selection algorithm to $C_{parent}$'s parent
```

4.1 c-TFLG

`c-TFLG` uses the text content of the edges contained in each cluster to determine whether to merge. If T is a text embedding method, m_{parent} and m_{child} the concatenated text content of edges in a parent and a child cluster respectively, $CosSim$ the cosine similarity function, μ_c an "inflection point" parameter, and w_t, w_c as relative weight parameters, then the merge criterion F_c^* for `c-TFLG` is defined as

$$F_c^* = 1 < w_t e^{1 - \frac{\sum_{C_{child} \in \mathbb{C}} [\Lambda_{C_{child}}]}{|\mathbb{C}| \Lambda_{C_{parent}}}} + w_c \frac{\sum_{C_{child} \in \mathbb{C}} [CosSim(T(m_{parent}), T(m_{child})) - \mu_c]}{|\mathbb{C}|} \tag{2}$$

If the content in each of the child clusters is similar, the content in the merged cluster should also be similar to each child, tipping the equation towards merging the clusters. The text embedding method T can be any one of a number of embedding approaches detailed in a later section. The parameters w_t and w_c control how strongly the clustering decision is weighted between the temporal dynamics and content similarity respectively, with the temporal dynamics included via the original stability function $\Lambda = \frac{1}{\delta t}$. The selection of these parameter values will depend on the user's desire to balance temporal and textual features in the data.

The parameter μ_c controls the inflection point for content similarity: if the cosine similarity between two clusters is greater than μ_c, then merging is encouraged, but if the similarity is less then merging is discouraged. For many bodies of text and embedding methods the cosine similarity between random pairs of content may not be 0 on average, hence the introduction of this inflection point parameter to ensure that the function is not biased towards or against merging in general. An easy way to find a starting μ_c value is to calculate the average similarity between random selections of text in the data, establishing a baseline for cosine similarities in the data. Though depending on the data this value may be too low if conversations adjacent in time are slightly more similar than completely random pairs.

The authors of the original TFLG use distributed memory and parallel processing techniques to ensure scalability, all of which apply to `c-TFLG`. The only additional computation needed is embedding the text content, which can also be done in parallel.

5 Experiments

5.1 Datasets

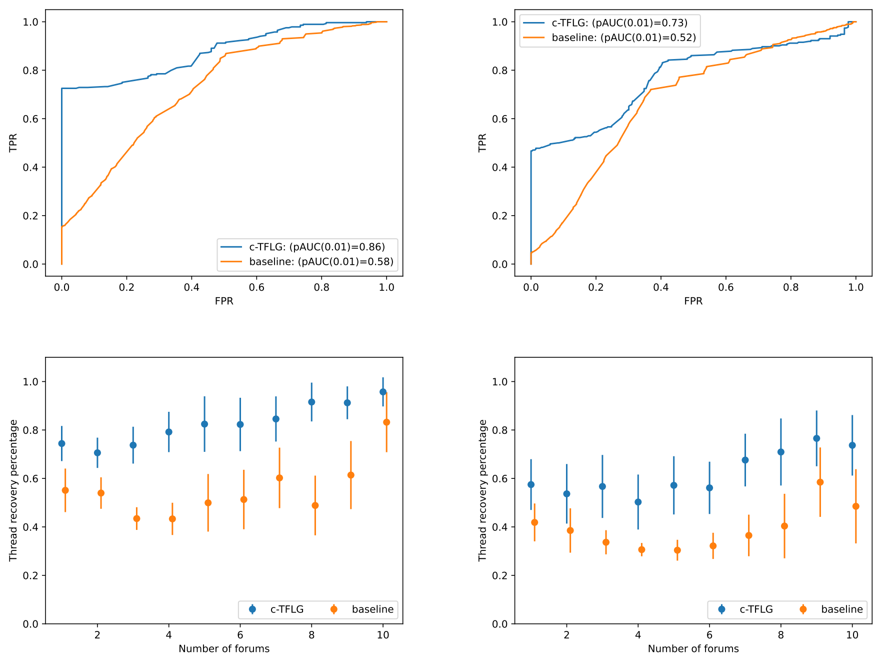
We test our methods on two different datasets: Stack Exchange and Reddit. The Stack Exchange dataset is a collection of data from ten Stack Exchange forums (`https://archive.org/details/stackexchange`), including Stack Overflow, Mathematics, Linux & Unix, Economics, and others. It includes questions, answers, comments, tags, and other related data from these sites. For our experiments we use the questions, answers, and comments. Each question can have multiple answers and each answer can have multiple comments. For the purpose of directed conversations, we induce an edge from a question to each of its answers and from each answer to each of its respective comments. The forums ranged in size from 47,325 edges to 234,472 edges.

For the Reddit dataset we use data scraped from `https://pushshift.io/`. Each subreddit in Reddit consists of threads consisting of an original post and the comment tree attached to it. We interpret the connection between a post and a comment on that post as pointing towards the comment. The 16 subreddits ranged in size from 9,805 edges to 1,115,935 edges.

Each forum of Stack Exchange has had its user IDs anonymized independently, making it impossible to determine which users post in multiple forums and making each forum dataset disconnected from the others. We synthetically induce shared authorship amongst forums by re-mapping the authors of disparate forums. Newly created authors are mapped to old authors using a power law distribution, where the majority of authors will only participate in 1 forum, with fewer participating in 2 forums, and even fewer in 3, and so on. Authors participating in a large number of forums are rare, which mimics the order of natural systems.

5.2 Metrics

Burst classification We first measure performance of our methods as a function of burst classification. For a single burst, we classify it as a “true positive” (label 1) if we either A) capture all the messages from a particular thread within it or B) the burst consists of exclusively messages from a single thread. Otherwise, bursts are labeled with a 0. To score a burst, we define the metric “thread recovery percentage" as follows. Recall a subreddit (Reddit) or forum (Stack Exchange) consists of several threads (each thread is on the topic of the subreddit/forum, but is typically focused on its own related topic). A simple metric would be to calculate the purity of each burst based on the percentage of messages originating from the same thread. However, this metric captures the precision of the method without measuring the recall. If the original thread is split apart into many small bursts, each burst would have perfect purity scores, but the full thread would not have been successfully reconstructed. For the purposes of this thread recovery experiment, we modify this simple purity metric and instead calculate how much of the actual thread we are able to reconstruct. To do so, for a single burst, we 1) calculate the "mode" of the threads of the messages present in the burst (the thread that appears the most in our burst), then 2) of the total number of messages in the maximally appearing thread, we calculate how many of those landed in the burst of interest.



(a) *stack-exchange* (b) *reddit*

Fig. 1: Top: ROC curves for both datasets for burst classification. `c-TFLG` results are shown in blue while baseline TFLG results are in orange. Bottom: Thread recover results for both datasets.

Thread recovery Recall we induce common authorship (as explained in Section 5.1). Here we present results over 20 trials, as we continually increase the number of forums that participate in each trial. As each forum has induced author overlap with the others, more forums means more "node confusion." Our goal in this experiment is to test how well our methods can recover a full thread-tree in Reddit or Stack Exchange given increasing node confusion. For a single trial: we start with 1 randomly selected forum, obtain bursts from baseline TFLG and `c-TFLG`, and measure performance given our modifications. Next, we add a second randomly selected forum and re-run both `c-TFLG` and baseline TFLG using the dataset of the mixture of both forums, including the induced author overlap. We continue in this fashion, iteratively adding a new randomly chosen forum until we reach the maximum number of forums. Each forum gets 10 randomly sampled threads, all of which participate in our dataset and threads are all sampled from the same randomly selected time window to ensure overlap.

5.3 Experimental Results

The results of this experiment are shown in Figure 1. Using the original forum/subreddit as ground truth, we obtain ROC curves as seen in the top row of Figure 1 for the Stack Exchange and Reddit datasets respectively. We plot the false positive rate (FPR) on the x-axis versus true positive rate (TPR) on the y-axis where we classify/score bursts in this fashion using the BERT embeddings. These values are aggregated over 100 trials for a varying number of forums and partial area under the curve (pAUC) values for a FPR rate of 1% are given for each method in the legend. Note that we see approximately a 40% increase in the pAUC by integrating content into the burst clustering. The bottom row of Figure 1 shows the performance in terms of thread recovery. The x-axis is the number of forums that are present in a single trial of this experiment and the y-axis is the previously described quality metric of thread recovery. The plot shows the mean of the 20 trials per forum with error bars plotted as the standard deviation across the 20 trials. Results from `c-TFLG` are shown in blue with the baseline TFLG in orange. In both cases `c-TFLG` performs better on the Stack Exchange data than on Reddit. We theorize that this is due to the nature of the datasets themselves; in Reddit, people are more likely to veer off topic, and/or respond with much shorter messages (that semantically will not embed close to the topic of the thread at hand). Stack Exchange messages are typically longer and tend to be more on topic. Regardless, we see significant improvement in bursts obtained when integrating content into the clustering as opposed to ignoring the semantic nature of communications.

6 Conclusions

The local dynamic clustering algorithm we have described offers a means to cluster dynamic graph data such as message streams according to both time dynamics and text content. In this paper we demonstrated its effectiveness when recovering human conversations from dynamic graph data. This framework can be easily expanded to use other local features for its clustering decisions in datasets that lack text metadata or have other useful metadata.

One limitation of this framework is the need for conversations to be somewhat coherent in time. If a conversation is halted then resumed after a long time gap (e.g. a group meets weekly and resumes an old conversation) the clusters will not be connected in the dendrogram and it will not be possible to merge the two halves of the conversation. Post-hoc analysis would be required to recognize the similarity between the clusters. A limitation of `c-TFLG` is the length of the text associated with each edge. Short, content-less messages such as one-word responses will have no meaningful topics on the messages regardless of the text model used. In this case the surrounding context must be used to determine if those short messages belong to the conversation or not.

Next steps in this research include improved methodology for selection of parameters. It may be possible to automatically select some parameters: for example, the inflection point could be set to the average similarity between messages belonging to the same ground truth, so that messages with less similarity would be penalized. Or, given ground truth examples it may be possible to fit optimal parameters for the dataset. Both approaches require some form of ground truth examples, however, and choosing parameters for data where no examples are available is an open question.

References

1. Pramod Bhatotia, Umut A Acar, Flavio P Junqueira, and Rodrigo Rodrigues. Slider: Incremental sliding window analytics. In *Proceedings of the 15th international middleware conference*, pages 61–72, 2014.
2. Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 160–172. Springer, 2013.
3. Michael S Crouch, Andrew McGregor, and Daniel Stubbs. Dynamic graphs in the sliding-window model. In *Algorithms–ESA 2013: 21st Annual European Symposium, Sophia Antipolis, France, September 2-4, 2013. Proceedings 21*, pages 337–348. Springer, 2013.
4. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.
5. David Ediger, Rob McColl, Jason Riedy, and David A Bader. Stinger: High performance data structure for streaming graphs. In *2012 IEEE Conference on High Performance Extreme Computing*, pages 1–5. IEEE, 2012.
6. Shai Erera and David Carmel. Conversation detection in email systems. In *Advances in Information Retrieval: 30th European Conference on IR Research, ECIR 2008, Glasgow, UK, March 30-April 3, 2008. Proceedings 30*, pages 498–505. Springer, 2008.
7. Laetitia Gauvin, André Panisson, and Ciro Cattuto. Detecting the community structure and activity patterns of temporal networks: a non-negative tensor factorization approach. *PloS one*, 9(1):e86028, 2014.

8. Timothy La Fond, Geoffrey Sanders, Christine Klymko, and Van Emden Henson. An ensemble framework for detecting community changes in dynamic networks. In *2017 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–6. IEEE, 2017.
9. Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.
10. Andrew McGregor. Graph stream algorithms: a survey. *ACM SIGMOD Record*, 43(1):9–20, 2014.
11. Steven Noel, Eric Harley, Kam Him Tam, Michael Limiero, and Matthew Share. Cygraph: graph-based analytics and visualization for cybersecurity. In *Handbook of statistics*, volume 35, pages 117–167. Elsevier, 2016.
12. Michael Ostroski, Geoffrey Sanders, Trevor Steil, and Roger Pearce. Scalable edge clustering of dynamic graphs via weighted line graphs. *Journal of Complex Networks*, 13(3):cnaf006, 2025.
13. Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.
14. Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.
15. Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.
16. Giulio Rossetti and Rémy Cazabet. Community discovery in dynamic networks: a survey. *ACM computing surveys (CSUR)*, 51(2):1–37, 2018.
17. Andrei Sorin Sabau. Survey of clustering based financial fraud detection research. *Informatica Economica*, 16(1):110, 2012.
18. Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.
19. Huajie Shao, Shuochao Yao, Andong Jing, Shengzhong Liu, Dongxin Liu, Tianshi Wang, Jinyang Li, Chaoqi Yang, Ruijie Wang, and Tarek Abdelzaher. Misinformation detection and adversarial attack cost analysis in directional social networks. In *2020 29th International Conference on Computer Communications and Networks (ICCCN)*, pages 1–11. IEEE, 2020.