Using Graph Neural Networks for Heterogeneous Event Classification

Valerio Bellandi[1], Stefano Montanelli[1], Darya Shlyk[1] and Stefano Siccardi[2]

[1]Università Degli Studi di Milano, Department of Computer Science, Via Celoria 18, Milano, Italy
[2]Consorzio Interuniversitario Nazionale per l'Informatica, Via Ariosto, 25, Roma, Italy

Abstract

Graph Neural Networks (GNNs) are specialized machine learning models designed for analyzing and making predictions based on graph-structured data. In this paper, we introduce a GNN-based approach for the analysis of heterogeneous event graphs. By heterogeneous event graph, we refer to a temporal graph where multiple types of nodes representing events as well as multiple type of edges among events are defined. A distinguishing aspect of the proposed approach lies in the specification of the subgraph of nodes for each event to classify, and in the use of embedding techniques for representing all the relevant features/attributes of the subgraph. The proposed approach is compatible with different GNN models; the use of PSCN, namely a CNN for subgraph classification, is discussed in the paper under two different settings. Preliminary results on two different case-studies, as well as a real application in the field of historical data classification are also discussed.

Keywords

Multi-layer Event Classification, GNN, Historical Data Analysis

1. Introduction

Graph Neural Networks (GNNs) are a class of machine learning models that have gained attention in the recent years for their ability to work with data structured as graphs. Graphs are mathematical structures that consist of nodes and edges, and they are used to represent a wide range of real-world systems, such as for example social networks, recommendation systems, and biological networks. GNNs are designed to address the unique challenges posed by these graph-structured data and have emerged as a powerful tool in various domains. Since their first description in the seminal paper [1], several models have been proposed, with the common characteristic of operating directly on graph data and capturing complex dependencies and interactions within the data. Many GNNs can propagate information between connected nodes in a graph and update the feature representations of nodes by aggregating information from their neighboring nodes. Often, low dimensional vector representations of nodes (embeddings) are used in order to make their features usable in downstream tasks, such as node classification,

SEBD 2024: 32nd Symposium on Advanced Database Systems, June 23-26, 2024, Villasimius, Sardinia, Italy
*Corresponding author.
†These authors contributed equally.
valerio.bellandi@unimi.it (V. Bellandi); stefano.montanelli@unimi.it (S. Montanelli); darya.shlyk@unimi.it (D. Shlyk); stefano.siccardi@unimi.it (S. Siccardi)
0000-0003-4473-6258 (V. Bellandi); 0000-0002-6594-6644 (S. Montanelli); 0009-0006-4051-6871 (D. Shlyk); 0000-0002-6477-3876 (S. Siccardi)
© 2024 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR-WS.org/Vol-3741/paper36.pdf

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

link prediction, and graph classification. Sometimes, GNNs propagate information through multiple layers of the network. A hierarchy is the result, capable to capture increasingly complex patterns and abstractions in the data.

In this paper, we propose an approach based on GNNs for node classification over a heterogeneous event graph. By heterogeneous event graph, we refer to a temporal graph where multiple types of nodes representing events as well as multiple type of edges among events are defined. In addition to specific application-dependent attributes, an event is associated with a timestamp denoting when the event occurs. This way, the approach can deal with a variety of event graphs, and it is capable to represent entities from many different domains, including social or physical system, that can be organized in a network. The approach consists in the specification of a subgraph of nodes for each event to classify, and in the use of embedding techniques for representing all the relevant features/attributes of the subgraph. A GNN is finally invoked to enforce event classification over each selected subgraph. Our approach can be employed with different GNN models, thus, for tests and experiments, we consider the use of two different solutions, that are both based on a CNN for subgraph classification called PSCN [11] under two distinct settings based different criteria for graph construction. We stress that a contribution of our proposed approach is about the application/specialization of PSCNto event graph classification with multiple types of nodes and edges (i.e., graph of heterogeneous events). In particular, when defining the subgraphs of the events to classify, we allow to specify application-driven constraints expressed through a time window in which events have to occur for being considered and filters to customize the type of nodes/edges to include. Two different case studies are discussed about i) blackmailing detection and ii) bitcoin transaction rating, respectively. For future experiments, we envisage a further application case study of the proposed approach to a historical dataset in the framework of the Tacitroots ERC H2020 project[1].

The paper is organized as follows. In Section 2, we discuss the relevant references in the literature. Section 3 presents the proposed GNN approach for event classification. Experiments are illustrated in Section 4. Section 5 outlines a further case-study on the historical dataset of the Tacitroots project. Concluding remarks are finally given in Section 6.

2. Related work

Temporal Event Graphs, sometimes called Dynamic Graphs, have been described in [3], and they are characterized by nodes that represent events and links that connect an event and its nearest temporal neighbours. The paper also considers "motifs" which characterize pairs of events according to the involved nodes and the direction of their interaction. Events involve just two entities and they have no labels, that is just one type of events is considered. The work in [4] deals with the dynamics of diffusion-like processes on temporal networks and introduces weighted event graphs. The authors consider spreading agents that can be transmitted onward from a node only within some time δt. This time limit is the control parameter of a percolation problem and the structure of paths created within δt determines where the process can move. Weighted event graphs are static, weighted, and directed acyclic graphs whose nodes are the

[1] https://sites.unimi.it/tacitroots/.

events. Each event is linked to all events sharing an involved entity; links have a weight proportional to the difference of moments when the linked events happened in the timeline of the system. To apply a threshold to the event graph one can "cut" all links whose weight exceeds the threshold. The work in [5] considers the transformation of a temporal network to a weighted event graph similar to the one described above. An event embedding mechanism is defined by sampling the neighbourhood of events. The sampling is done according to probabilities determined by the weights of the links that connect the actual event to its neighbours. These samplings are passed as inputs to the Skip-Gram model, resulting in a d-dimensional vector space in which events are represented by Cartesian coordinates. Experiments indicate that these embeddings capture in large part the time ordering, and some mesoscale structures defined by temporal vicinity of events. An application is the study of the final outcome of the epidemic originated by a specific event.

Regarding Graph Embedding, in general terms, the embedding consists in projecting a graph in a $d-$dimensional space in such a way that the graph structure is preserved as much as possible. We quote, for instance, the *node2vec* method [6], VERSE [7], and proNE [8]. Entity embedding computation by node2vec is based on random walks traversing the graph and measuring the probability of observing a specific entity considering different staring points. On the other hand, VERSE learns embeddings by training a single-layer neural network, and ProNE enhances the embedding using spectral spaces techniques. The above algorithms do not distinguish link types or labels, they are concerned, in other terms, with single layer graphs. On the other hand, event graphs are multilayered, that is their nodes are connected by links of different types. In [9], a multilayered version of VERSE, MultiVERSE, can be found: a fast and scalable method to learn node embeddings from multiplex and multiplex-heterogeneous networks, based on Random Walks. In [10], a method based on the normalized graph Laplacian matrix and its spectrum is described, that is able to compute suitable embeddings for multi-layered graphs. Following a different path, in [11], the authors describe a construction for node neighborhood that can be directly applied to machine learning using a CNN network, dealing with a graph in the same way one would do with an image (PATCHY-SAN or PSCN). As a further option, in [12], the authors describe a method that we will call "ad hoc embedding" to derive an embedding specifically tailored for a query, based on suitable node counts in a temporal window around each event. Analogous methods are used in [13] and [14] for pattern matching in event graphs and to study correlations between events, respectively.

The Graph Neural Network paradigm was formalized in [1], that is a type of neural network that can act directly on graphs. The review paper [2] describes a taxonomy of GNNs, and it discusses some limitations of the original method and possible variants proposed to address the issues. It is worth noting that we use the term GNN in a broader sense than the model described in [1], that includes some models quoted in our previous section, e.g. PSCN. The GNN models have been extended to multilayer graphs, for instance in [15], where a multi-dimensional convolutional neural network model, mGCN, is proposed to capture rich information in learning node-level representations for multi-dimensional graphs. In [16], a general approach is described that is not task specific to extend GNNs to graphs with multiple edge types between nodes. With respect to the above solutions, the main contribution of our proposed approach is about the application and specialization of PSCN to event graph classification involving multiple types of nodes and edges, thereby forming a graph of heterogeneous events. Specifically, we

propose a solution for delimiting the subgraphs for event classification, and the specification of related application-driven constraints. These constraints are expressed through a time window in which events must occur to be considered, as well as filters that allow the customization of the types of nodes and edges to include.

3. The proposed approach

As described before we propose to extend the GNN model presented in [1] for being applied to heterogeneous event graphs. Our proposed approach consists of three steps described in the following, namely *event graph construction*, *event subgraph definition*, *subgraph analysis*.

The **event graph construction** is performed by defining a node for each event to classify. An event is characterized by a timestamp, featuring when the event occurred. Two event nodes are connected by an edge in the graph when i) the two events are *next* (i.e., successive) to each other in time, and ii) they have a *shared attribute*, namely a common property that describes the link between the two nodes, and it is used to label the edge. For instance, if the event nodes are phone calls, possible shared attributes are i) the calls have the same caller, ii) the calls have the same receiver, iii) one call is received by the caller of the other, and iv) one call is made by the receiver of the other. It is worth noting that a node is linked only to the next node/event with a certain shared attribute, not all the successive nodes. As an example, in case of calls with the same caller as shared attribute, a call/node is only linked to the successive call according to the timestamps. Further details about the event graph construction are provided in [14].

The **event subgraph definition** is about the extraction of an appropriate subgraph for each event to classify. A subgraph is targeted to a specific event node and it has to include all the nodes and edges that are potentially useful and pertinent to classify the target. Each subgraph can work as the receptive field of a CNN, and it is extracted according to the method described in [11]. A subgraph must contain a prefixed number of nodes, and a notion of *time window* is introduced to define a mechanism for choosing the nodes to include in the subgraph. Since a node of our event graph is associated with a timestamp, the time window is defined by setting a maximum time afterwards (or backwards) with respect to the initial target event node. Furthermore, the time window specifies the maximum distance of a node from the target for being included in the subgraph. Given an event node to classify (i.e., target node), the definition of the corresponding subgraph is performed as follows. The target node is added to the subgraph. Then, the neighboring nodes are added to the subgraph when i) the timestamp of neighbors falls in the expected time window, and ii) the prefixed size of the subgraph as well as the maximum distance from the target are not reached. The addition of neighboring nodes is repeated until the above conditions are not violated. Dummy nodes are added to the subgraph when the time window is passed and the graph contains less nodes than expected. Optionally, filters can be defined to apply application-driven constraints based on specific types of edges/nodes during subgraph extraction. A vector representation, namely an embedding, can be derived from the subgraph by listing all the nodes and edges, including any feature that can be relevant for classifying the target.

In Figure 1, we show an illustrative example of subgraph definition by considering a target node/event denoted by T. The graph contains different types of event each one denoted by a

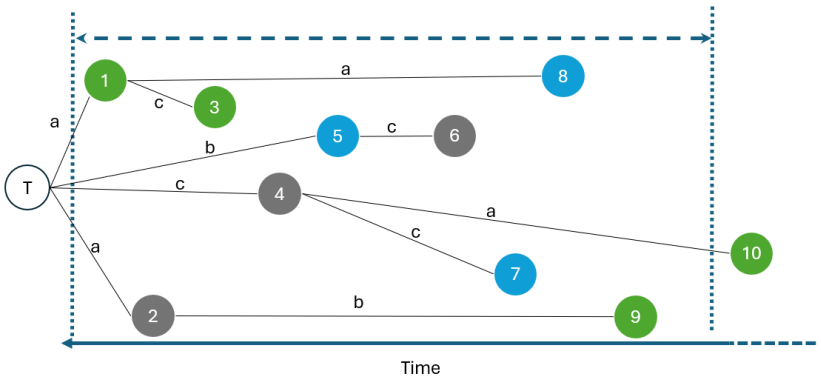

Figure 1: Example of subgraph definition.

specific color, and different types of edge, each one denoted by a specific textual label. For the construction of the subgraph of the node T, we assume that the time flows from right to left and we include nodes in the subgraph of T by considering i) a time back-window criterion, ii) a maximum of 7 subgraph nodes, and iii) a max distance of 2 from the target. As further application-driven constraints, we specify that only edges of type *a* and *c* are considered when the distance from the target is 1, while all the edge labels are considered at distance 2. As a result, the subgraph for the target node T contains the nodes 1, 2, and 4 considering the nodes at distance 1 from target; and the nodes 3, 7, 8, and 9 considering the nodes at distance 2. The node 5 is excluded since linked to the target by an edge labelled with *b* (distance 1). The node 10 is excluded since it is beyond the time window. The node 6 is not considered since the previous node 5 has been excluded. Since the expected number of nodes is inserted in the subgraph, no dummy nodes are added.

The **subgraph analysys** is finally enforced. The goal is to classify the target event of each subgraph that has been previously defined. Different methods can be employed to perform subgraph analysis, and the most appropriate method can be invoked depending on the case study at hand. In the remaining of the paper, we consider two specific application case-studies with related experiments based on a CNN for subgraph classification that is PSCN.

4. Experiments

For experimentation, we exploits two different application case-studies with related datasets. The first case-study is called **blackmailing** and the goal is to recognize when a node/event can be considered a request for money from blackmailed people according to the following pattern: somebody calls or writes to a number of persons in a short time; afterwards each of the persons withdraws (using his/her debit card or bank account) some money; then the caller deposits on his/her bank account an amount that is approximately the sum of the withdrawn amounts. The possible events in the graph are phone calls, email messages, debit card withdrawals, and bank account movements. We are aware that the one above is not a realistic blackmailing

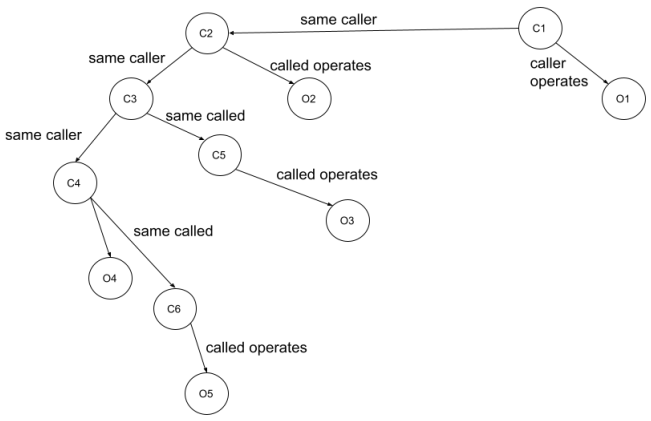

Figure 2: Example of sub graph for a (simplified) pattern of the blackmailing dataset.

pattern; however, in this case-study, our goal is to show that our proposed approach is capable to correctly recognize a given pattern even when the graph is noisy and the target is confused by spurious event nodes.

Synthetic data for experimentation was generated as follows:

- records for 1000 persons were generated and assigned phones, email addresses, bank accounts, debit cards and "friends", that is a vector of other persons' identifiers
- based on adjustable parameters, the program generated records for phone calls, email messages, debit card withdrawals and bank account moves at discrete times
- a number of persons generated the specified pattern, with events marked as targets

To test the sensitivity of the methods, we generated two sets of data with increasing confusion between the targets and the other events. For all the sets the simulated period was 200 days:

1. average interval between i) non-target phone calls is 24 hours, ii) email messages is 4 days, iii) bank account movements is 8 days, iv) debit card withdrawals is 6 days; v) minimum interval between target events is 2 hours, the maximum is 6 hours;
2. average interval between i) non-target phone calls is 6 hours, ii) email messages is 12 hours, iii) bank account movements is 8 days, iv) debit card withdrawals is 6 days; v) minimum interval between target events is 4 hours, the maximum is 12 hours.

For each set of parameters, two datasets were generated: one for training and one for testing. In particular, the datasets with the parameter set 1 are called BM1a and BM1b, while the datasets with the parameter set 2 are BM2a and BM2b. Figure 2 shows the sub graph that one expects to find for a target pattern. The rightmost node, labelled O1, is the final deposit. It is linked to the nearest phone call, C1, made by the operator. This call is not necessarily one of the calls made

Table 1
Event Graphs for the blackmailing case-study

Code	N.Nodes	N.Edges	N.Deposits	N.Targets
BM1a	338,621	1,739,669	16,158	90
BM1b	338,617	1,739,578	15,998	90
BM2a	920,102	5,176,665	16,080	90
BM2b	920,200	5,177,204	16,145	90

to solicit money (target calls), but it is linked to a previous one made by the same caller, until a target call is found. This can be linked to a financial operation like in case of C2 and O2, if the called person did not receive any other call before withdrawing money. However, it may happen that the financial operation is linked to another call, that can be found following the edges linking calls received by the same person, as for the nodes C3, C5 and O3. Both situations may occur, as for the node C4, if several financial operations were made by the called person in the same period.

We created the event graph linking each node with the previous ones of each type, sharing an agent, that is: phone calls to phone calls with the same caller, same receiver, the receiver as caller and the caller as receiver; analogously for links to to emails (caller as sender, etc.); to the previous deposit made by the caller and by the receiver; to the previous account move made by the caller and the receiver. Analogous links where created for the other event types. A total of 49 link types were created. The derived event graphs features are presented in Table 1. This is a case-study of binary classification on the nodes of type Deposit: the goal is to detect target and non-target deposits according to the above pattern of blackmailing.

The second dataset is called **Bitcoin-OTC**[2] and it is commonly used for this kind of experiments (e.g., [18, 19]). It consists of 35,592 records, that are edges between two users of Bitcoin OTC. Each edge represents a rating the first user gave (called *source*) of the second (called *target*) at a specific time, in a scale from -10 (total distrust) to +10 (total trust) in steps of 1, with 89% positive values. We note that normally this dataset is processed as an edge classification task. Considering edges as event nodes, we can trace it back to our event classification task. We created the event graph linking each node to the previous ones having: the same source (ss), the same target (tt), the source as target (st), and the target as source (ts). The resulting graph had 124,145 edges and 35,592 nodes, corresponding to the edges of the original graph.

4.1. Results for the blackmailing dataset

In the experiments, the PSCN method has been considered with two distinct settings characterized by different criteria used for subgraph definition. Results are summarized in Tables 2 in terms of True Positives, False Negatives, True Negatives, and False Positives. We used a time window of 70,000 sec. (around 20 hours), a requested number of 50 nodes for BM1a and BM1b, and 100 training epochs; a time window of 100,000 sec. (around 28 hours), a requested number of 150 nodes for BM2a and BM2b.

[2] http://snap.stanford.edu/data/soc-sign-bitcoin-otc.html.

The first setting of PSCN is *generic*, in the sense that we do not consider constraints on the shape of target subgraphs. Constraints on the distance of nodes from the target have been not specified. Edge labels were filtered, so that only 16 edge types have been selected in the subgraphs, in order to select phone calls and emails originated by the operator of deposits and bank movements, withdrawals made by people who received a phone call or an email and chains of phone calls and emails by the same person.

Table 2
Blackmailing Event Graphs, PSCN results

Code	TP	FN	TN	FP	Code	TP	FN	TN	FP
BM1a train	90	0	16,068	0	BM1b train	90	0	15,908	0
BM1b test	85	5	15,889	19	BM1a test	73	17	16,060	8
BM2a train	90	0	15,990	0	BM2b train	90	0	16,055	0
BM2b test	21	69	16,045	10	BM2a test	29	61	15,964	26

The second setting of PSCN is *specialized*, in the sense that we define a maximum distance of nodes from the target (i.e., max distance is 3), and constraints on the node/edge types to consider are specified at each distance. At distance 1, that is for neighbours of the target node to be classified, only links to phone calls and emails made by the person who made the deposit were considered. At distance 2 and 3, only emails and phone calls made by the same person as above, and financial operations by people who received emails and phone calls were considered. An illustration of the resulting subgraphs can be found in Figure 3 where an example of target (on the left) and non-target (on the right) graphs are provided. Both graphs start with a deposit (red) and contain chains of emails and phone calls (blue) as well as financial operations (withdrawals and account moves, yellow). Results are reported in Table 3.

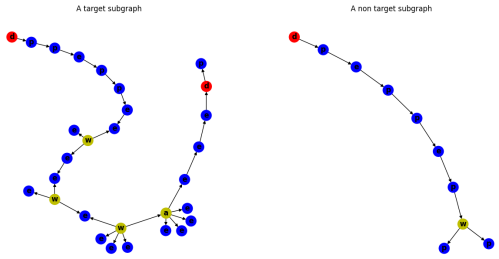

Figure 3: Blackmailing dataset: example of a target (left) and a non-target (right) sub graph.

We note that the specialized setting of PSCN provides very interesting results, especially on the second, more complex, group of tests. Considering that in all the cases the task consists in finding a small number of events, we see that the number of false positive is always fairly low. However, the general method fails to detect approximately two thirds of the targets in the complex case, while the specialized model is able to find always more than 85%.

Table 3
Blackmailing Event Graphs, PSCN results with the specialized version

Code	TP	FN	TN	FP	Code	TP	FN	TN	FP
BM1a train	90	0	16,067	1	BM1b train	90	0	15,908	0
BM1b test	90	0	15,908	0	BM1a test	90	0	16,064	4
BM2a train	87	3	15,988	2	BM2b train	90	0	16055	0
BM2b test	77	13	16,049	6	BM2a test	83	7	15,982	8

4.2. Results for the Bitcoin-OTC dataset

We considered subgraphs whose events shared either the source or the target user with the event to classify. We considered a time window of 4,000,000 sec., and a maximum distance of three steps from the target node. Figure 4 shows an example of subgraph for a 10 labelled event, and one for a -10 labelled event. Labels from -10 to -5 are drawn in red, from -5 to -1 in yellow, from 1 to 5 in blue and greater that 5 in green. It can be seen that in this dataset, the subgraph structures may be not so different, and the values of the labels may be the discriminant feature. The obtained results can be compared to those of [20] and [18]. In [20], the problem is

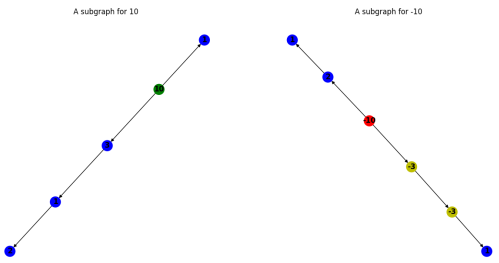

Figure 4: Example of a 10-labelled (left) and a -10-labelled (right) sub graph.

expressed in terms of edge classification and it reports a F1 micro average scores in a figure, with values above 0.85. In [18], the problem is presented in terms of edge-weight prediction and it reports as measures the Root Mean Square Error (RMSE) and the Pearson Correlation Coefficient (PCC) of predictions vs true values. The best results quoted are 0.31 RMSE and 0.49 PCC for a single method; 0.26 RMSE and 0.67 PCC using predictions of several algorithms in a supervised classification. Scores where normalized to the $[-1, 1]$ interval for this computation. We note that RMSE seems to be a better measure for the quality of results, than f1 score. Actually, a real user would be more interested in an approximation of the values, that are ratings in an interval, than exact guesses. Some older work are presented as edge sign prediction (e.g. [21]), and a comparison between their results and ours cannot be done, as they used other datasets. However, we report, only for test data, a *binary* f1 score referred to edge signs. Results are shown in Table 4. We performed experiments under two settings. In the first one, we used

Table 4
Bitcoin-OTC Event Graphs, PSCN results

Case	f1	RMSE	PCC	Binary f1	Case	f1	RMSE	PCC	Binary f1
50% train	0.610	0.295	0.621		33% train 1st	0.603	0.297	0.612	
50% test	0.608	0.304	0.595	0.966	66% test 1st	0.608	0.305	0.591	0.966
33% train 2nd	0.609	0.290	0.613		33% train 3rd	0.608	0.293	0.630	
66% test 2nd	0.603	0.301	0.587	0.966	66% test 3rd	0.600	0.301	0.595	0.964

half of the data for training, and half for testing. In the second setting, we used one third for training, and the remaining for testing, by also considering three three different compositions of the training set.

5. Application of the approach to historical data

In the following, we envisage a real application case-study where the proposed approach could be effective. The case-study is based on a dataset of historical events in the context of the Tacitroots ERC H2020 project. The goal of Tacitroots is to study an extensive corpus of unpublished documents covering several decades of scientific work in the European continent. The collected data includes thousands of letters exchanged between scholars throughout Europe in the 16th century, discussing open scientific issues, and documenting various experimental findings. Our idea is to consider the exchange of letters between scholars as events and to build a corresponding event graph according to the approach described in Section 3. In particular, an event node in our graph represents a letter sent by a scholar to a colleague at a certain moment in time. An event is associated with the sending date, the sender name, and the receiver name. Furthermore, every event node is associated with a topic that is discussed in the letter, such as medicine, astronomy, or history. Additionally, a node/event can be associated with further attributes when it mentions the execution of an experiment with related observations, as well as an instrument description. Adjacent nodes can be linked by edges of different type featuring a shared attribute. Possible examples of edge type are: common sender/receiver, or cross-reference between sender and receiver of the linked letters. Further edge types are about common experiments or instruments mentioned in two letters.

Figure 5 illustrates an example of event graph built with the Tacitroots datasets and it represents a sequence of three letter exchanges. Letter 1 and Letter 2 refer to letters written by Vincenzo Viviani on 19th and 21st March 1664 to Giovanni Borelli and Carlo Rinaldini, respectively. Both events discuss thermology and mention telescope, and they are therefore connected by SAME SENDER and SAME INSTRUMENT relationships. Letter 3 represents the response of Carlo Rinaldini to Vincenzo Viviani addressing astronomy and holding a notice of observation on comets. This later event is linked to the previous one via RECEIVER = SENDER and SAME INSTRUMENT relationships. Considering the above example of the event graph, we would like to address the following node classification task. Given a target event node, describing a letter exchange between two scholars, the objective is to predict the topic addressed by the letter from the recent history of correspondences involving the sender and the receiver of the target event. We believe that integrating available node-level descriptive features, including instrument

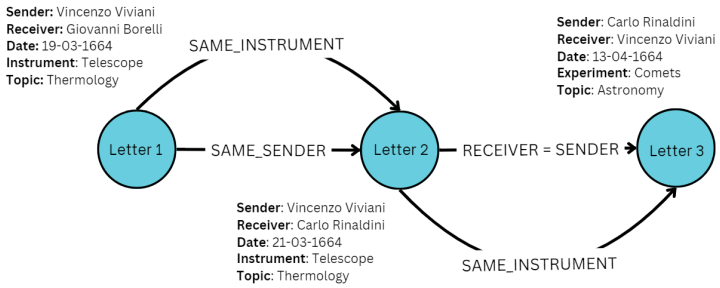

Figure 5: Fragment of the historical event graph based on the dataset of the Tacitroots project.

and experiment, can enhance classification accuracy. For this reason, we plan to consider all these features along with edge labels when computing the vector-based representation of target nodes and related neighbors. Ultimately, with this case-study we aim to showcase the applicability of our methodology to a broader field of historical studies, and its potential for uncovering valuable insights into the relationships and topics discussed among scholars.

6. Concluding remarks

In this paper, we presented an approach for heterogeneous event classification through GNN. The preliminary experimental results on two different case-studies discussed in the paper provides interesting scalability performances. For instance, run times for the BM2 datasets are roughly increased of 1/3 compared to run times for the BM1 datasets; on the other side the BM2 datasets are approximately three times larger than BM1. Another interesting point is that results are stable even when the training set size is reduced by more than 30%.

As a further result, we note that the proposed approach is quite flexible, and it can be used to represent very different event graphs. The Blackmailing graph has several types of nodes (with no attributes) and several types of edges. The Bitcoin graph has only one type of nodes with an attribute and just two types of edges. The first gives raise to relatively more complex subgraphs than the latter, as Figure 3 and 4 show. A further case-study is also sketched-out based on real historical data, with single type of nodes and multiple types of edges. Test and experiments on this case-study are currently ongoing.

As a final remark, we note that the choice of the features to use for defining the subgraph requires some domain knowledge of the underlying phenomena. Also the choice of the most appropriate GNN can depend on the domain and features of the considered events. Experiments with further GNN models are currently ongoing. For instance, the KONG method (Kernels for Ordered-Neighborhood Graphs) [17] has been experimented on both the paper case-studies, but results are less interesting than those obtained with PSCN, thus requiring deeper investigations.

Acknowledgments

Research supported, in parts, by *i)* Università degli Studi di Milano under the program "Piano di Sostegno alla Ricerca". *ii)* project MUSA - Multilayered Urban Sustainability Action - project, funded by the European Union - NextGenerationEU, under the National Recovery and Resilience Plan (NRRP) Mission 4 Component 2 Investment Line 1.5: Strengthening of research structures and creation of R&D "innovation ecosystems", set up of "territorial leaders in R&D" (CUP G43C22001370007, Code ECS00000037), *iii)* project SERICS (PE00000014) under the NRRP MUR program funded by the EU - NGEU. Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the Italian MUR. Neither the European Union nor the Italian MUR can be held responsible for them.

References

[1] Scarselli, F., Gori, M., Tsoi, A., Hagenbuchner, M. and Monfardini, G., The graph neural network model, IEEE Transactions on Neural Networks 2009, vol. 20, no. 1, pp. 61-80.

[2] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, Maosong Sun, Graph neural networks: A review of methods and applications, AI Open, Volume 1, 2020, Pages 57-81, ISSN 2666-6510, https://doi.org/10.1016/j.aiopen.2021.01.001.

[3] Mellor, A. The Temporal Event Graph. Journal of Complex Networks (2017)

[4] Kivelä, M., Cambe, J., Saramäki, J. et al. Mapping temporal-network percolation to weighted, static event graphs. Sci Rep 8, 12357 (2018). https://doi.org/10.1038/s41598-018-29577-2

[5] Torricelli, M., Karsai, M. Gauvin, L. weg2vec: Event embedding for temporal networks. Sci Rep 10, 7164 (2020). https://doi.org/10.1038/s41598-020-63221-2

[6] Grover, Aditya and Leskovec, Jure node2vec: Scalable feature learning for networks, Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining (2016)

[7] Tsitsulin, Anton and Mottin, Davide and Karras, Panagiotis and Müller, Emmanuel VERSE: Versatile Graph Embeddings from Similarity Measures. Proceedings of the 2018 World Wide Web Conference (2018). https://doi.org/10.1145/3178876.3186120

[8] Zhang, Jie and Dong, Yuxiao and Wang, Yan and Tang, Jie and Ding, Ming ProNE: Fast and Scalable Network Representation Learning. IJCAI (2019)

[9] Léo Pio-Lopez, Alberto Valdeolivas, Laurent Tichit, Élisabeth Remy, Anaïs Baudot. MultiVERSE: a multiplex and multiplex-heterogeneous network embedding approach. 2020. ffhal-02997038

[10] Xiaowen Dong, Pascal Frossard, Pierre Vandergheynst, and Nikolai Nefedov Clustering on Multi-Layer Graphs via Subspace Analysis on Grassmann Manifolds (2014). IEEE TRANSACTIONS ON SIGNAL PROCESSING, VOL. 62, NO. 4

[11] Mathias Niepert, Mohamed Ahmed, Konstantin Kutzkov Learning Convolutional Neural Networks for Graphs. Proceedings of the 33rd International Conference on Machine Learning (2016)

[12] Bellandi, V. and Ceravolo, P. and Maghool, S. and Siccardi S. Graph embeddings in criminal

investigation: towards combining precision, generalization and transparency. World Wide Web 25, 2379–2402 (2022). https://doi.org/10.1007/s11280-021-01001-2

[13] Bellandi, V. and Ceravolo, P. and Maghool, S. and Pindaro, M. and Siccardi, S. Pattern Matching Algorithm in Event Graphs, Intl Conf on Cyber Science and Technology Congress, CyberSciTech, Falerna, Italy, 2022, pp. 1-7, doi: 10.1109/DASC/PiCom/CBDCom/Cy55231.2022.9927860.

[14] Bellandi, V. and Ceravolo, P. and Maghool, S. and Pindaro, M. and Siccardi, S. Correlation and pattern detection in event networks, IEEE International Conference on Big Data (Big Data), Orlando, FL, USA, 2021, pp. 4103-4112, doi: 10.1109/BigData52589.2021.9671512.

[15] Yao Ma and Suhang Wang and Chara C. Aggarwal and Dawei Yin and Jiliang Tang. Multi-dimensional Graph Convolutional Networks. Proceedings of the 2019 SIAM International Conference on Data Mining (SDM). 2019 https://doi.org/10.1137/1.9781611975673.74

[16] Grassia, M. and De Domenico, M. and Mangioni, G. mGNN: Generalizing the Graph Neural Networks to the Multilayer Case. arXiv:2109.10119

[17] Draief, Moez and Kutzkov, Konstantin and Scaman, Kevin and Vojnovic, Milan. KONG: Kernels for ordered-neighborhood graphs. Advances in Neural Information Processing Systems. Vol. 31. 2018

[18] Kumar, S. and Spezzano, F. and Subrahmanian, V.S. and Faloutsos, C. Edge Weight Prediction in Weighted Signed Networks. IEEE International Conference on Data Mining (ICDM), 2016.

[19] Kumar, S. and Hooi, B. and Makhija, D. and Kumar, M. and Subrahmanian, V.S. and Faloutsos, C. REV2: Fraudulent User Prediction in Rating Platforms. 11th ACM International Conference on Web Searchand Data Mining (WSDM), 2018

[20] Pareja, A. and Domeniconi, G. and Chen, J. and Ma, T. and Suzumura, T. and Kanezashi, H. and Kaler, T. and Schardl, T.B. and Leiserson, C.E. EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs, arXiv:1902.10191, 2019

[21] Leskovec, J. and Huttenlocher, D. and Kleinberg, J. Predicting positive and negative links in online social networks, WWW, 2010.