World Wide Web
https://doi.org/10.1007/s11280-021-01001-2

Graph embeddings in criminal investigation: towards combining *precision, generalization* and *transparency*

Special issue on computational aspects of network science

Valerio Bellandi[1] · **Paolo Ceravolo**[1] · **Samira Maghool**[1] · **Stefano Siccardi**[1]

Received: 1 April 2021 / Revised: 17 September 2021 / Accepted: 22 December 2021

Abstract
Criminal investigation adopts Artificial Intelligence to enhance the volume of the facts that can be investigated and documented in trials. However, the abstract reasoning implied in legal justification and argumentation requests to adopt solutions providing high precision, low generalization error, and retrospective transparency. Three requirements that hardly coexist in today's Artificial Intelligence solutions. In a controlled experiment, we then investigated the use of graph embeddings procedures to retrieve potential criminal actions based on patterns defined in enquiry protocols. We observed that a significant level of accuracy can be achieved but different graph reformation procedures imply different levels of precision, generalization, and transparency.

Keywords Knowledge graphs · Enquiry protocols · Criminal investigation · Graph embeddings

1 Introduction

Criminal investigation and prosecution are complex procedures that have to deeply examine large documental sources to spotlight facts often unrevealed, denied, or deliberately

This article belongs to the Topical Collection: *Special Issue on Computational Aspects of Network Science*
Guest Editors: Apostolos N. Papadopoulos and Richard Chbeir

✉ Samira Maghool
samira.maghool@unimi.it

Valerio Bellandi
valerio.bellandi@unimi.it

Paolo Ceravolo
paolo.ceravolo@unimi.it

Stefano Siccardi
stefano.siccardi@unimi.it

[1] Department of Computer Science, Università degli studi di Milano, Via Celoria 18, Milan, Italy

withheld by criminal agents. Criminal agents also find out and make use of those assets that are less traced in the documental sources exploited by persecutors and law enforcement agencies. On the contrary, a successful investigation should result in exhaustive and documented proceedings delineating the facts and the responsibilities that comprise criminal actions. This must be done in accordance with the law and following policies that can guarantee fair, impartial, and efficient procedures. Artificial Intelligence (AI) [2] have been proposed in support of the great deal of work implied by criminal prosecution. AI can process large data sources and automatically identify relevant patterns to support the prosecutor with recommendations. It has been observed that automating the inspection of data sources can significantly impact the size of documents a prosecutor can bring in the trial [21] but the benefits of AI go beyond the volumes of facts that can be documented. AI can enhance the ability to identify criminal actions by flexibly matching the patterns defined by prosecutors and extending their scope [3], anticipating this way the ability of criminals in hiding their actions by layering the stages that bring to exchanges and revenues.

A basic requirement of AI is *precision*, that in information retrieval refers to the rate of relevant information within the set retrieved information. This requirement impacts how reliable will be considered AI predictions. However, high precision achieved at the cost of low generalization is not favorable, otherwise, the validity of the AI support will be very narrowed and over fitted to specific problem. This notion can be measured by the *generalization error* that rates how accurately an algorithm can predict outcome values for previously unseen data. Intuitively, larger is the domain of data that can be accurately handled by an algorithm better it generalizes. It has been observed that the generalization error may be significant when applying AI [19] to legal documentation. Either if the intelligence is achieved by expert systems, driven by explicit rules, by supervised learning, discriminating from large amounts of examples, or by the hybridization of these techniques, obtaining low generalization errors is challenging. Legal systems are, indeed, constructed on very abstract definitions and interpretations of facts, that can be hardly reduced to a restricted set of observations and that evolve over time following the concerns of the society. The juridical justifications that are provided to decide on a case are often constructed a-posteriori, based on the imputations to be supported and using a selective set of legislative provisions. This problem is, for example, observed in [26], where an average accuracy range from 58% to 68% is reported in predicting decisions of the European Court of Human Rights for future cases based on the cases from the past. As the authors note, the Court formulates the justifications in a way that is conducive to fit the conclusion. A same legal framework can be applied differently based on the conditions encompassing a case.

A prosecutor has to verify specific facts, enquiries must address information in the scope of the probed events, all related actions must be transparently documented. Therefore, another key requirement for AI in criminal investigation is *transparency*. Following [13, 25] we highlight transparency in algorithmic decision-making systems is more than accountability. An accountable software is software we can observe, verify the tasks it executed, the systems, and the users it interacted with [12]. A transparent AI has to support the retrospective analysis of the decision process followed by the algorithms, decomposing it into the main elements that determined the final decision and allowing to backtrack the decision steps followed.

In [3] we presented a method to support criminal investigation by operationalizing *Enquiry Protocols*. Prosecutors adopt protocols to identify the qualified sources, registers, and documents that can be exploited to pursue a crime, the information to be verified and integrated, and the formal stages to be followed during the prosecution. Using a set of data integration techniques, data sources can be organized in queryable *Knowledge Graphs*.

Prosecutors can express a protocol as a set of subsequent operations over the data sources, referring to the historical cases they addressed. Each operation can then be translated into exact queries over the knowledge graph. An user interface can guide prosecutors along a workflow that allow applying exact queries and examining their results. However, an intrinsic limit of exact specifications is that a small variation in the structure of a data source may result in an unmatched occurrence. Using AI the exact knowledge of prosecutors can be generalized supporting the identification of patterns similar to the ones identified from their experience but differing in some respect. For example, a protocol could define a suspected drug dealer as a person traveling with an unusual number of baggage pieces, missing the fact he/she could simply escort an unusual number of travelers.

An efficient and versatile method to perform AI techniques on knowledge graphs is using *Graph embeddings*, i.e. transforming the entities represented in the graph into vectors, the standard input format for many AI algorithms. Experimental studies have reported that graph embeddings techniques can get the highest standards in terms of *precision* in classification tasks [17]. Embeddings are projections of graph nodes and edges into a low-dimensional vector space [17], with positive impact on the processing speed of AI algorithms. At the same time, embeddings can effectively capture the higher-order structure beyond a graph letting emerging paths not otherwise visible for approaches based on exact queries. For example, thanks to graph embeddings, two graph entities with different local properties can be efficiently matched to obtain a rate of their coverage.

In this paper we assess the conditions a graph emebedding method has to meet in order to reach *high precision*, *low generalization error* and *retrospective transparency*. In AI solutions reaching high accuracy, these requirements cannot coexist [38]. Our experimental analysis, extending the findings in [3], confirms a single technique cannot reach full coverage of these requirements. The solution we propose is then obtained by combining different approaches. Embeddings of the exact queries exploited in a protocol input is a selection task that can identify sub-graphs similar to the sub-graphs returned by the queries in a protocol. This makes graph queries robust to variations in the structure of the queried knowledge graph reducing the *generalization error* imposed by the system or uncertainties in the retrospective data.

Graph embeddings techniques have, however, significant limits in terms of *transparency*. They allow comparing two objects with high precision but lack interpretability [17]. To explain the recommendations prompted to the prosecutors we cannot just match two objects with high accuracy we have to provide visibility to the discriminates used to find this match [40]. A suspected drug dealer escorting travelers instead of luggage must be identified thanks to the proximity between travelers and pieces of luggage and this proximity must be highlighted.

To overcome these limits, in [3], we proposed to perform the selection of discriminant points by a *hierarchical filtering* procedure. The proximity measures in the embedding space, which quantifies the neighborhood of a given node, filters out the dissimilar sub-graphs in two steps. In the first step, filters according to the structural similarity, while in the second step, the higher-order of neighborhood is considered. A coarse-grained *reformation* [6] of the knowledge graph is required to give grip at the filtering procedure. The knowledge graph is reformed into an undirected network of interconnected instances that preserves the structural characteristic of the sub-graphs depicted by the protocol eliminating redundant details. This reformation procedure, filtering out details unrelated to the protocol, improves identifying a set of sub-graphs with high similarity to the sub-graphs resulting from the queries of the protocol. At the same time, this procedure support transparency as the identified sub-graphs can be reported to the entities involved in the reformation

process. However, not all the reformation procedures have the same impact on *generalization error* and *transparency*. In our experimental study, we assessed different reformation procedures getting different levels of precision in detecting the criminal patterns contained in a synthetic dataset generated by augmenting a real-world example. The experimental section contained results of three different embedding algorithms, node2vec, VERSE, and proNE, used for extracting the proximity of entities in the latent space.

We compared the solutions presented in [3] with a method based on the classification of group of events labeled as normal/criminal event patterns. The classification task has implemented by Support Vector Classification (SVC) algorithm that the literature report it as highly accurate and actually achieved full accuracy in our experiments. Our results show the precision achieved in [3] and generalization are comparable. At the same time, our proposal demonstrates better transferability due to the less specific reformation stage we applied.

More specifically, the remainder of the paper is organized as follows: Section 2 provides an overview of Knowledge Graphs and Graph Embeddings. In Section 3, we describe the materials and methods adopted for this study and we present our methods. In section 4, a money-laundering scenario is introduced to guide the controlled experiment we run. The results of our evaluation come in Section 5. A discussion on the results concludes the paper in Section 6.

2 Related works

In this section, we review the background work required to develop our solution.

Knowledge graphs Knowledge graphs [30] describe interlinked entities with deep representational power. Generally speaking, a knowledge graph $\mathcal{G}(\mathcal{V}, \mathcal{E}_R)$ consists of $\mathcal{V} = \{v_1, ..., v_n\}$ nodes, $R = \{r_1, ..., r_k\}$ edge types, and $\mathcal{E}_{r_h} = e_{ij}$ edges, with $i, j \in [1, n]$ and $h \in [1, k]$. With $\mathcal{G}_{r_h}$ we refer to a graph containing edges of type r_h only. Including edges and nodes of multiple types a knowledge graph supports data integration and reasoning capabilities for retrieving implicit knowledge rather than only allowing querying on explicit knowledge.

The adoption of knowledge graphs in criminal investigation has been extensively studied under various headings. In [11] a public dataset from Manchester, U.K., is considered, mapping all the entities relevant to criminal queries to four basic concepts: *Person*, *Object*, *Location*, and *Event*. Meaningful links between nodes of these types highlight criminal evidence or support the computation of useful statistics about the criminal activity in an area.

Mobile phone calls can be given a natural graph structure, linking callers to called people and involved cellular towers. This structure can be used to find associations between criminals and anti-social people. Several studies proposed to identify patterns of calls, people most often called by the suspects, and their communication with convicted criminals, see for instance [9, 23].

In [10] the authors analyzed the database of a factoring company, to generate risk profiles of clients using social network metrics. The period under consideration spanned nineteen months. Factoring companies buy accounts receivable from third parties at a discounted price, repaying the seller with working capital and taking the responsibility of collecting the debts. As the money paid is a "clean" asset, this kind of business can be a case of money laundering when invoices are fictitious. Using techniques from social network analysis, potentially suspect entities have been found among the actors of big or most frequent

financial transactions. Companies operating in several economic sectors and located in some peripheral regions emerged from the analysis. Links between different companies sharing the same owner or representative have also been highlighted using visual tools.

The paper [31] considers a criminal network as a system that needs to reach equilibrium, in the setting of General System Theory in Political Science. It is proposed to apply data mining to these networks of organized crime and social network analysis is then exploited to study the conditions bringing to this equilibrium.

In [7] the authors point out that, notwithstanding the evident benefits of knowledge graphs for criminal investigations, their practical application is often limited. The reason is that complex and time-consuming data integration procedures are necessary, due to the legacy technologies adopted by law enforcement offices, and by regulatory barriers.

Graph embeddings Knowledge Graphs are widely adopted for supporting data modeling and data analytic procedures. Their detail-oriented structural complexity is, however, an obstacle for eliciting the information they convey using ML methods, which are engineered for ingesting feature vectors. Features are measurable properties, distinguishing characteristics, of the phenomena being observed. In ML data are seen in terms of a vector of features, but the data encoded in an interconnected graph lay on multi-modal feature spaces. Regarding the subject matter, the observed phenomena could refer to the constituent entities of this graph, such as specific nodes or links (connections), or to the variations of this graph, such as creation or annihilation of links which could be translate as "events". Given a predefined dimensionality to the feature space, multiple vectors can be equally representative of the graph.

Graph Embeddings are techniques to project a graph in a d-dimensional space that preserves as much as possible the graph properties. Given a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, graph embedding properly provides the transformation of a graph $\mathcal{G}$ into $\mathbb{R}^d$ by a matrix $X \in \mathbb{R}^{x \times d}$ whose rows $\mathbf{x}_i \in \mathbb{R}^d$ are the vectors, otherwise said embeddings, that encode the graph properties. In the recent literature, the most common algorithms used for generating embeddings are Neural Networks (NN) [1] which have been proved to be very effective in classifying patterns. The strategies initially proposed focused on embedding the nodes by representing their proximity to the other nodes in the graph. Other strategies have focused on embedding the edges or the sub-graphs. Applications focusing on graph matching have brought to techniques for encoding an entire graph [37]. The proposed approached can also differ because of the data reformation procedure adopted and the intrinsic specification of the studied problem. On the other hand, the reformation process on the dynamic network could be implemented considering the temporal sequence of events. In this sense, the event embedding techniques could help in finding the proximity of events as nodes in latent space. In particular in this work we compare strategies based on *Entities embedding* and *Event embedding*.

i) Entities embedding:

For embedding entities (nodes and links) of a graph into the latent space, the $\mathbf{x}_i$ are generated using proximity measures that are qualified in the first- and higher-order, depending on the number of transformations applied on the adjacency matrix encoding the graph by the pairs of nodes that are adjacent [17]. The general goal is to keep the interconnected nodes closer in the embedding space, besides, efficient task-dependent transformation strategies are studied to meet the requirements of different ML algorithms.

As one of the prominent transformation strategies, we can name the Random Walk (RW) based strategies. These methods use random walkers to traverse the graph and explore it.

The proximity measure $s(i, j)$ is approximated by the probability that nodes i and j co-occur on a random walk over the network. The RW-based algorithms not only suggest a flexible stochastic definition of node similarity function, that incorporates both local and higher-order neighborhood, but also are efficient since during the training stage just the pairs of nodes co-occurring on a random walk are considered, not all pairs of nodes [18, 32]. By extending this idea, a graph can be viewed as a set of routed sub-graphs around every node, the embedding of a graph can then be interpreted as an aggregation of the embeddings of every node in the graph [29]. The mentioned properties of RW-based embeddings make these strategies eligible in exploring the variations of exact sub-graphs. In general, methods using higher-order proximity measures outperform other methods and get excellent performance with ML tasks such as clustering or classifying nodes and graphs, or predicting link creation and decay [17].

ii) Event embedding

Event embedding idea is a newly emerging, powerful method in the area of complex network studies [38]. The main focus of this method is the temporal network in which the network variations happen on a small time scale. In most cases, the embedding procedure of temporal networks into latent, low dimensional space contains redundant information about single entities. On the other hand, the event embedding method implies reforming the data transforming the edges and nodes expressing temporal conditions as running events on the network, instead of constituting entities of the network. Based on the specific event time dimension, a vector $\mathbf{x}_i$ will be assigned to the event i. The defined event-vectors could create an "event graph" which contains the *events* as nodes that are connected to the non-simultaneous adjacent events with one shared end respecting the time direction. A consequence of applying this approach is that the network reformation procedure to be executed to organized data as a sequence of events is significantly domain-dependent and an error in the specification of correct event vector may result in loosing the accuracy. In Section 4.3 we discuss this point in the context of the enquiry protocol encoded in our experiments.

3 Methodology

In this section, we present our methodology by illustrating its basic components.

Enquiry protocols Enquiry Protocols define the qualified sources and documents that can be exploited to pursue a crime, the information to be verified and integrated, and the formal stages to be followed during the prosecution [16]. It may be helpful to think about an enquiry protocol as a funnel filled with many sources of information and data. That data passes through the investigative filters that determine the possibilities, develop theories, and test those theories against known evidence and facts. Data narrows itself down until a reasonable grounds of belief is found, getting inculpatory or exculpatory evidence [14].

To operationalize enquiry protocols we defined them as a set of *data integration* procedures followed by a set of nested *graph queries*.

Data integration It can be stressed that individuals are normally involved in several relational networks and during their daily activities generate many different types of data. Often, relationships from one of these sources shed light on data from another; for instance, links between companies can be inferred by integrating links between individuals and companies, taken from the companies' records, and links between individuals, taken from social

networks message flows. As a consequence, when constructing a knowledge graph, it is important to identify all the relevant data sources and the data integration rules to apply. A tricky point may consist of defining identity criteria, to decide if an entity described in several data sources is the same individual or not. Actually, only in the most favourable cases, the relevant entities are attached to the same identifier in all sources. When an automatic identification is impossible, the ingestion programs should generate separate entities, linking them as "potentially the same", leaving the final decision to the user.

The focus of this paper is not on data integration, the interested reader may refer to [24] for further details. We assume here that at the end of the data integration step a *knowledge graph* containing the information relevant to the addressed investigation is available.

Graph queries Graph queries [20] can be used to extract sub-graphs that are qualitatively or quantitatively different from the queries constituting an enquiry protocol. Queries can spotlight nodes with unusual characteristics, e.g. the number of edges. When the edges have attributes, one can search for an unusual aggregated value of the edges connected to a node. When the graph has a temporal dimension, queries can detect nodes affected by an unusual number of events in a time interval of some meaningful duration. To make sense, these queries presuppose some knowledge of the statistics of the phenomenon, or at least some heuristics, in order to define what should be meant as "unusual". Queries can find specific sequences of some types of nodes and edges that are known to be found (or to be missing) when the entities represented are involved in some crimes. As an example, we can consider the process of purchasing an item, that usually consists of a purchase order linked to product delivery, an invoice, and a payment. Patterns lacking the invoice node can be considered suspicious.

Graph reformation Data pruning and graph reformation technically point to several approaches to avoid overfitting in building decision trees [8]. In the presented work, by graph data reformation, we aim to pre-process the provided data, for the sake of performing efficient and targeted embedding tasks in large graphs. Therefore, we adopt filtering procedures to reconsider the knowledge graphs' instances as dynamic/static nodes interconnecting via fixed/temporal links. The goal is representing a structured knowledge graph $\mathcal{G}$ with multiple edge types by an undirected network $\mathcal{N}$ equivalent to a graph $\mathcal{G}_{r_h}$ totally determined by an edge type r_h. During these pruning and reformation steps, we strictly respected the semantics of the source data sets. The details of the pruning and reformation process applied in our scenario, are discussed in Section 4.2.

Graph embeddings As discussed in Section 2, the embedding techniques provide a probabilistic approximation of a similarity function which possibly incorporates the low/high-order structure of the graph. More specifically, after data pruning and graph reformation, we adopt the *node2vec* method [18], VERSE [36] and proNE [39], to map the nodes in an undirected network $\mathcal{N}$ to a latent space. The *node2vec* algorithm, known as one of the best-performing embedding methods [22], can be efficiently optimized using stochastic gradient descent [27]. A second-order random walk, considered in this method, takes the previously visited node into consideration allowing to encode the latent structure of the explored graph.

We have also considered VERtex Similarity Embeddings (VERSE), a method that explicitly learns any similarity measures among nodes by training a single-layer neural network. In its learning core, VERSE stands between deep learning approaches on the one hand and

the direct decomposition of the similarity matrix on the other hand by VERSE. It is possible to choose a vertex-to-vertex similarity measure, whose distribution is preserved by the embedding.

Another method of embedding we have implemented in this paper is the proNE. ProNE initialize network embeddings in an efficient manner and enhance the representation power of these embeddings. The first step is achieved by formulating network embedding as sparse matrix factorization; The second step is to leverage the higher-order Cheeger's inequality to spectrally propagate the initial embeddings with the goal of capturing the network's localized smoothing and global clustering information.

Temporal slicing While the real-life networks are evolving in time, following such temporal evolution in graphs is challenging due to the huge size of data to be extracted and represented. With data pruning and graph reformation, we can pursue balance between computational resources and huge size of data. Following the same perspective, in our experiments, we adopt two approaches for representing the temporal dimension. In the *pseudo-static approach* we model the extracted graph data within an approximately large time duration, so the entire information is conclusively presented in the graph assumed as a static one. In the *temporal approach*, the graph is sliced in temporal layers [4] considering the need of problem. The nodes relevant to a protocol are first identified in each separated layer to then follow the edges interconnecting them. This approach brings a significant reduction in computational cost since rather than embedding the accumulated data resulted from large time duration, we apply the embeddings on the temporal layers which are significantly lower in size. To get concrete results of temporal embedding, a reference sub-graph connect the embeddings of different layers in a hierarchical order depending on the temporal sequence (Figure 3).

Event graphs The system under consideration is a temporal network, consisting of entities that interact and interactions constituting an event happening at a definite time. It is an intrinsically dynamical system that can be reduced to a static one using temporal slicing as described above. Some other specific techniques have been also developed for node embedding in temporal networks (e.g. [5]); among these a particularly interesting one is the construction of an event graph [35], as it results in a static network preserving the main characteristics of the original one, and including all the events occurred, as no time-slicing is used. In event graphs, the nodes (events) are the links of the original network and relations are found considering non-simultaneous events that are adjacent, that is they share at least one end. Adjacent nodes are connected by links, respecting the direction of time. A weight may be attached at links, corresponding to the absolute time difference between events; in this case, we speak of weighted event graphs. With the described construction, for instance, events occurring at the same place or involving the same individuals are linked to each other in sequences. We use a model of this type to validate the precision achieved by our proposal, as described in Section 4.3.

4 Experimental design

To illustrate our methodology we present it in the context of a controlled experiment illustrated in the following.

4.1 A money laundering scenario

Money laundering represents a relevant section of criminal activities that can be detected using data analytics. Funds move on standard financial instruments and can be traced by looking at anomalous behaviour. Multiple data sources are, however, required for identifying these anomalies. Knowledge graphs represent therefore a valuable tool in this area. The scenario we addressed involves a protocol to detect organizations hiding their financial movements by fractionating transactions in small transfers. The final collection of the funds is organized by numerous cash withdrawals at ATMs, using multiple debit cards. Because, nowadays, most of the banking services notify withdrawals via SMS, an anomalous pattern can be identified from the records when this technique is applied. In fact, usually, the mobile phone connected to a debit card is located in proximity to the ATM used for withdrawal. When a large organization adopts this technique, a same mobile phone can be connected to an uncommon number of cards, and the SMS are notified to cells not in the proximity of the ATMs. This intuition can be summarised in an *enquiry protocol* defining the following conditions. Within brackets, we provide the threshold used in our example but, clearly, these values are parametric.

1. Some people are in charge of withdrawing a large amount of money using multiple debit cards (> 10).
2. They choose at random an ATM and withdraw the maximum allowed amount (250) with each debit card.
3. When the withdrawals are concluded they call their supervisor (within 5 minutes of the last withdrawal).
4. Then they move to another ATM and follow back to point 1.
5. After each withdrawal, the bank sends an SMS to the mobile phone associated with the debit card.
6. The supervisor holds all the mobile phones associated with the debit cards.
7. The supervisor follows the operations from a geographical area not in proximity to the withdrawals.

Our controlled experiment was executed on a proof of concept realized for Italian prosecutors. The dataset was generated augmenting a set of real-world examples provided by the Milan prosecutor's office. As a graph database management system we used the *Neo4j* desktop edition and the *Neo4j* browser. We integrated data from several sources; in some cases, we used public databases, in others, we did not have real data available for privacy reasons and we generated simulated data. Among the first, we quote public data about the purchases made by the Italian public administration, companies' websites, and geographical positions of the ATMs and the cellular radio towers. Among the latter, we quote the records of mobile phone calls, debit card details, and withdrawals.

Data were imported into the Neo4j databases using Cypher shells or python programs in particular we imported: 996 ATMs of the Milan urban area with their real longitude and latitude; 2104 cellular towers from the same area with their real longitude and latitude; 30000 mobile phones with program generated numbers and sim card identifiers; 10000 debit cards with program generated codes, each connected to a phone number; 891096 phone calls; 370035 withdrawals.

Phone calls and withdrawals have been generated assuming three criminal withdrawers and a supervisor were acting for five days from 9:00 AM to 5:00 PM using 10 debit cards each. Normal user withdrawals could happen at any time. Criminal and normal withdrawals

and phone calls have been generated separately. The generation of criminal withdrawals started placing the supervisor in a randomly chosen place in the Milan urban area. Then, a minimum of 5 ATM was assigned to each withdrawer, choosing among those located in a randomly chosen area. Waiting times between withdrawals were normally distributed with a mean of 2 minutes and a standard deviation of 30 seconds. We generated 1473 criminal withdrawals in groups of a maximum of 10 and a minimum of 3 withdrawals, with an average of 6.2. These withdrawals and phone calls were flagged as *"criminal"* in order to check the ability of the embeddings to spot them. A confirmation call to the supervisor was made on an average of 5 minutes after the last withdrawal, with a 1 minute standard deviation. Normal users' withdrawals have been generated at random times in the 24 hours each day, using the same ATMs. Withdrawal amounts were chosen in 25% of cases as 100 euros, in 50% as 150 euros, in 12.5% as 200 euros, and in 12.5% as 250 euros. We also generated phone calls for normal users, using the same cell towers. Call duration and time were chosen at random.

Exact resultants Figure 1a shows a sub-graph including a withdrawal of 250 euros for which the bank sent SMS to a cell phone not close to the ATM. The withdrawal is represented by the red node connected to the cell tower, in light blue, via the ATM, in orange. The withdrawal is connected on the left to the debit card node and the leftmost section of the graph shows that the message is sent to another cell tower, this means that the phone is far from the ATM. This behaviour reflects points 5 and 6 of the scenario, but it is not enough to detect a suspicious transaction, because it might happen in a number of normal cases. In order to get better evidence, we expect a large number of 250 euros withdrawals in a short time (points 1 and 2 of the scenario). Accordingly, we divide the total observation time into suitable intervals and select the pairs of ATMs and intervals with higher numbers of maximal withdrawals to look for the pattern in Figure 1a.

We then use the ATM and period codes to get a new pattern illustrated in Figure 1b in the right area of the graphs the big red circles represent 10 withdrawals of the maximum amount carried out in a short time from the same ATM. In the left area, the big blue circles are 10 SMS sent by banks to confirm the withdrawals: it is clear that all of them used the same cell tower, so the phones are close to each other and far from the ATMs.

4.2 Entities embedding method

Graph reformation Graph embeddings techniques accept in input an adjacency matrix that can capture complex graph structures using high-order proximity. High-order proximity is effective for organizing embeddings in a metric space but not for filtering them base on a specific node or edge values. For this reason, we implement data reformation steps that simplify the structure of the knowledge graph without changing the semantics of the resultants of our exact queries by pruning the redundant details. This reformation stage , will allow filtering patterns by filtering on a single node and have a positive impact on the computational efficiency of our method. The reformation stage we apply is composed of the following filtering steps:

- Ignore the debit cards and substituting them with the registered phone numbers in a branch of a bank.
- Change the modality of withdrawals (large red nodes), from "nodes" in the sub-graph of Figure 1a to "edges" in the simplified form, connecting the ATMs to phones numbers/debit cards.

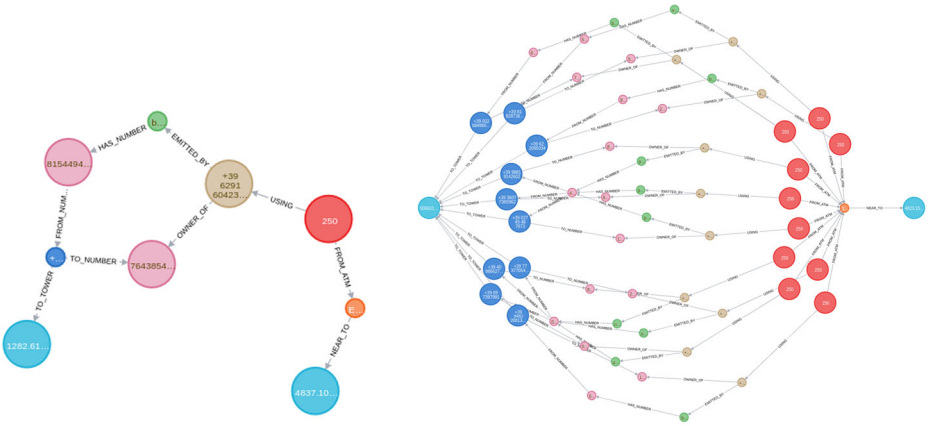


Figure 1 **a** Pattern of a withdrawal when the phone associated to the debit card is far from the ATM. **b** Pattern of a set of 10 withdrawals as described by our scenario

– Change the modality of phone calls/SMS (blue nodes), from "nodes" in the sub-graph of Figure 1a to "edges" in the revised form, connecting the phones/debit cards to towers.
– Omit any intermediary node, such as branches of bank or bank phone numbers.

Applying the aforementioned stage, we get the sub-graph of Figure 2 that highlights the main characteristics of a criminal pattern we aim to identify. The tower providing the service to an ATM is geographically far from the tower sending confirmation SMS to the phones

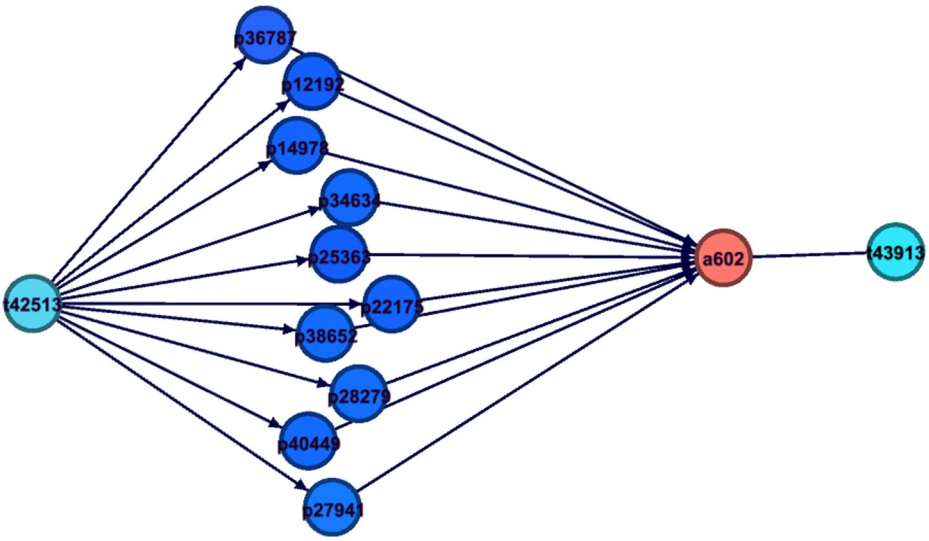


Figure 2 The schematic of reformed graph data, describing the specific characteristic of a criminal pattern. The tower in proximity to the ATM is different to the tower communicating with the phones held by the supervisor of this criminal organisation

connected to the suspicious withdrawals. Let's call the debit card/phone nodes as P, the ATM nodes as A, and the tower nodes as T. Considering this dual role for phones, at the same time counterpart in the $P-A-T$ and $A-P-T$ paths, we can radically simplify the graph $\mathcal{G}$ into an undirected network $\mathcal{N}$. This abstract coarse-grained form makes the implementation of embeddings on our resulted network more efficient in detecting the criminal sub-graphs out of the enormous original graph. The constituent nodes and edges of $\mathcal{N}$ can be encoded into an adjacency matrix $\mathbb{W}_{i,j}$ that:

$$\mathbb{W}_{i,j} = \begin{cases} 1 & \text{if a withdrawal or SMS happen between } i \text{ and } j \\ 0 & \text{otherwise} \end{cases},$$

where $i, j \in A|T|P$, are nodes presented in network $\mathcal{N}$.

Embedding procedure As discussed in Section 3, we compare two different procedures for embedding the network $\mathcal{N}$.

In the *pseudo-static approach*, the final configuration of the network $\mathcal{N}$, is the cumulative resultant data of 5-days of ingested data. By applying the *node2vec* embedding algorithm [18] on $\mathcal{N}$, specific d-dimensional vectors $\mathbf{x}_i$ to each node in $\mathcal{N}$ are assigned in the embedded space.

The proximity of nodes in the embedded space is evaluated using a cosine similarity function of their vectors of the given nodes i and j in the network:

$$s(i, j) \approx \frac{\mathbf{x_i}.\mathbf{x_j}}{|\mathbf{x_i}||\mathbf{x_i}|}. \tag{1}$$

Thereafter, we call n the number of the most-similar instances to a reported ATM involved in criminal withdrawals, for example, the ATM: "a602" in our sample graph data. These n-similar instances are chosen under the condition of $s(\text{"}a602\text{"}, j) = \mathcal{S}$ while $0 \leq \mathcal{S} \leq 1$ and are sorted by the larger assigned s-values. We should be noticed that, in order to get a complete list of similar ATMs, we need to choose n large enough since some of the phones and towers also may be returned as similar nodes to ATM: "a602". In our case, by trial and error, we have found out that choosing $n = [1000, 2000]$ almost all ATMs with $\mathcal{S} > 0$ are returned.

Even though F_1 score illustrated in Figure 5b indicates that leveraging the embedding method, efficiently returns a list of ATMs involved in criminal scenarios, these ATMs, by themselves, are not of much interest practically. From a prosecutor's point of view, criminal withdrawals by certain debit cards are required to be found out in order to operationally track a criminal scenario.

To this aim, we introduce some formal definitions to describe the procedure. First, We return a list of the ATMs similar to the known ATM: "a602", as $\mathcal{L}_{a602}$. In the same way, nodes in proximity to the i^{th} element of the list $\mathcal{L}_{a602}$, are inserted in list $\mathcal{L}^i_{a602}$, and then grouped in the list of lists $\mathcal{L} = [[\mathcal{L}^1_{a602}], [\mathcal{L}^2_{a602}], ...]$. Each criminal sub-graph, structurally, contains an ATM from the $\mathcal{L}_{a602}$ list and a tower named T_i, which should be located closely in the embedded space. According to our case study, the money laundering scenario, the supervisor of criminal withdrawals who holds the phones, does not change his place occasionally, so the number of nearby towers providing SMS service to phones are few. These towers frequently take part in different criminal withdrawals of different ATMs (different sub-graphs).

The procedure of recognizing the criminal debit cards/phones is as follow:

- For every single ATM presented in the list $\mathcal{L}_{a602}$, we return the $\mathcal{L}^i_{a602}$ and compose the list $\mathcal{L}$.
- The most-frequent "towers", $\{T_i\}$ in the list ($\mathcal{L}$) are returned. Each corresponding peers of $\mathcal{L}_{a602}$ and $\{T_i\}$ lists, for example ATM:A_m and tower T_m, are constituents of a criminal sub-graph.
- Referring to the network $\mathcal{N}$ and adjacency matrix $\mathbb{W}$, the phones (P_k) that enabled a path from the tower: T_m through the ATMs: A_m, could be recognized, where:

$$\begin{cases} \mathbb{W}_{A_m,P_k} = 1 & \text{and,} \\ \mathbb{W}_{P_k,T_m} = 1 \end{cases}$$

However, we have to notice the network $\mathcal{N}$ evolves over time. Although the $A - T$ edges are fixed, due to their physical position, the $A - P$ and $P - T$ edges (respectively, withdrawals and SMS) are constantly changing over time. We have then designed an embedding procedure that exploits this temporal dimension to further reduce the computational complexity of the embedding procedure.

In the *temporal approach*, considering the fact that the criminal withdrawals may have taken place in small time intervals, we divide the entire network into different layers corresponding to the time intervals of the occurring withdrawals. Figure 3 proposes a schematic view of the network partitioned into layers corresponding to specific time intervals. In our experiment, we size a span of 2 hours.

Given the *ATM:"a602"* as an ATM involved in a criminal sub-graph, we made an educated guess to consider this ATM and all of its first and second-order neighbors as omnipresent instances in all network layers (the orange-colored node and its emanating green edges in Figure 3). The choice of the first and the second order of proximity is directly concluded from the abstract form of criminal sub-graph (Figure 2), indicating that starting from known ATM, at least two order of proximity is needed to reach the service provider tower in the embedded space. With this assumption, we are still able to find similar sub-graphs to the given sub-graph, in each time interval of occurring withdrawals and SMS.

In the studied problem, computational complexity significantly reduced by applying our temporal approach, due to a radical reduction in the dimension of the research space.

4.3 Event embedding method

Graph reformation This method requires reforming the network in terms of sequences of events. We start by identifying two types of *top-level events* at time $\mathbf{t}$ withdrawals $\{\mathbf{w_t}\}$, and phone calls $\{\mathbf{p_t}\}$, where $\mathbf{t} < \mathbf{T}$ the duration of study. They may happen independently where $\mathbf{w_{t_1}}, \mathbf{p_{t_2}}$ with $\mathbf{t_2} - \mathbf{t_1} \gg \mathbf{0}$, or a withdrawal may trigger a phone call, $\mathbf{w_{t_1}} \longmapsto \mathbf{p_{t_1+\Delta t}}$ with $\Delta\mathbf{t} \to \mathbf{0}$ which indicates the $\mathbf{p_{t_1+\Delta t}}$ is triggered by $\mathbf{w_{t_1}}$. According to the enquery protocols presented in Section4.1, in detecting the criminal withdrawals, the geographical distance between the tower which provides the service to the ATM during the withdrawal process and the tower provides the phone call or SMS is a highly relevant parameter. Furthermore, the number of repetitive withdrawals from a single ATM, and the marginal amount of withdrawals are the other key points to b considered. Recalling these three key points, in the graph reformation step we build the vectors which contain the required spatio-temporal information. The reformation procedure is divided into two steps. In a pre-processing step, we build the *compound events*, $\{A, B, N, M\}$: $\{\{w_t\} \oplus \{\mathbf{p_{t+\Delta t}}\}$ with $\mathbf{0} < \mathbf{t} < \mathbf{2T}\}$, identifying the withdrawals and the triggered phone calls, including SMS if any, and assign a type

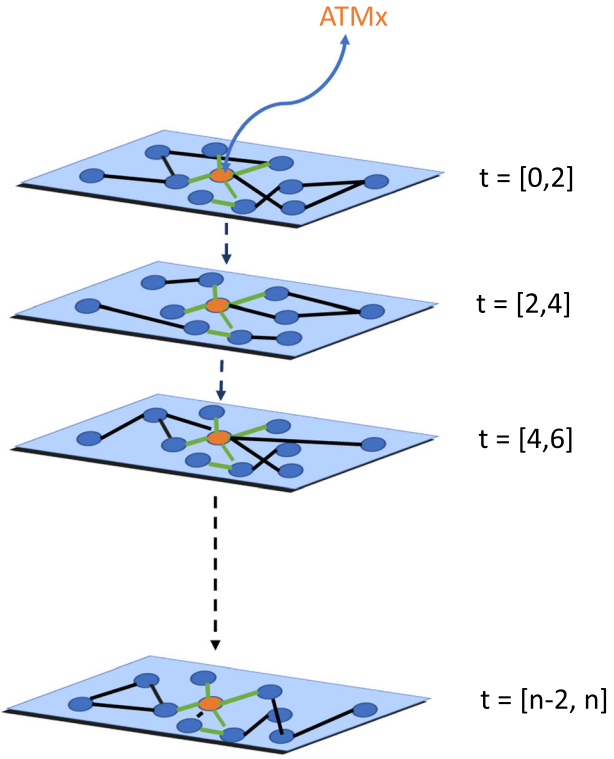


Figure 3 Converting the 5-days cumulative network into partial networks containing the temporal edges occurring in the corresponding time interval which is mentioned beside each layer. The orange node and green links represent the studied ATM and its related links, which is involved in a criminal sub-graph. This criminal sub-graph is present in all partial networks

to each of these events, depending on the characteristics of the withdrawal and the call. The characteristic of withdrawals could be specified by amount, origin ATM and nearby tower while the phone calls (SMS) are characterized by their providing towers. The embedding procedure builds a vector for each compound event, whose components are the type of the events and the numbers of events of each type that happened in a time window that includes the event itself. For simplicity, in the rest of this paper, we avoid the full term of "compound event" and call them just as "event". We have considered four main event types namely:

1. Type Ax withdrawals have *maximal* withdrawal amount, with SMS received at a tower far from the ATM; x is the receiving tower id and acts as a sub-type.
2. Type Bx withdrawals have *non-maximal* withdrawal amount, with SMS received at a tower far from the ATM; x is the receiving tower id and acts as a sub-type.
3. Type N withdrawals are withdrawals of *any amounts*, with SMS received at the tower *near* the ATM; they have no sub-types.
4. Type M withdrawals are withdrawals of *any amounts*, with no SMS received; they have no sub-types.

The embedding is built starting at each event labeled as above and considering events that happened at the same ATM during a time-span $T = 10$ minutes before and after the event itself. The vector columns are as follows:

- first 4 columns contain the number of type A, B, M, N events found in the time window;
- next 4 columns contain 1 if the event is of type A, 0 otherwise; the same for types B, M, N;
- next 2 columns contain the number of different sub-types of A events and B events found;
- next 2 columns contain the number of A events with the same sub-type as the event and the same for B events;
- the last column is the target label (criminal or normal withdrawal) for supervised learning.

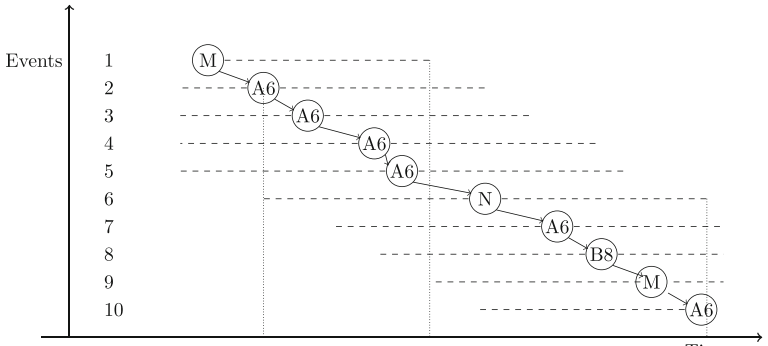
This structure is illustrated in Figure 4 where events are indexed from 1 to 10.

Embedding procedure The table in Figure 4 contains the results of the embedding procedure, where each row is the vector corresponding to an event. The columns are in the same order as in the vectors previously described, even if, the last column, about the target label, has been omitted. For instance, the embedding of the first event, that is of type M, is built in this way: we consider the time period spanned by the horizontal dashed line that starts at the event itself. Because it is the first event there is no dashed line before it. We see that four events of type $A6$ can be found in the time window, that is before the vertical dotted line. Accordingly, column "$n.A$" contains 4, and columns "$n.B$", "$n.M$" and "$n.N$" contain 0 because no events of these types are found in the time window. Of the following four columns, only the one labeled "Is M" contains 1, as the event is of type M. We have only a subtype, "6", for the 4 events of type A, so column "$St.A$" contains 1 and column "$St.B$" contains 0 (we have no events of type B, so no B subtypes at all). As the event is of type M, we have no type A or B events of the same subtype of the event under consideration and the last two columns contain 0 too. The other vectors are computed in the same way, but the time window moves, so that also events that happened earlier and are drawn above in the picture, must be taken into account. As an illustration, dotted vertical lines delimiting the window have been drawn for the sixth event, of type N.

Therefore, we end up with the "event vectors" containing the information of a specific event within the defined time window. Considering the money laundering scenario, we aim to detect criminal withdrawals out of a large number of withdrawals. Each of these vectors could be assumed as a 12-Dimensional vector and ingested into a classification algorithm. For this purpose, we use Support Vector Classification (SVC) algorithm by sklearn [33] and keras in TensorFlow [34]), which is effective in high dimensional space and memory efficient algorithm. The input layers receive the event vectors to processed. In this way, the event vectors, which encode the different types of withdrawals, will be labeled as criminal or non-criminal.

5 Evaluation

To evaluate our methodology we assess it in terms of *precision*, *generalization*, and *transparency* and compare the results of the three proposed methods, *temporal* and *static entities embedding* and *event embedding*. We calculate the precision as the ratio of correctly predicted positive observations to the total predicted positive observations and the recall (or sensitivity) as the ratio of correctly predicted positive observations to all positive observations in actual data. Here we have reported the f_1 score which is the harmonic mean of



Events	n.A	n.B	n.M	n.N	Is A	Is B	Is M	Is N	St.A	St.B	=A	=B
1	4	0	0	0	0	0	1	0	1	0	0	0
2	3	0	1	1	1	0	0	0	1	0	3	0
3	3	0	1	1	1	0	0	0	1	0	3	0
4	4	1	1	1	1	0	0	0	1	1	4	0
5	4	1	1	1	1	0	0	0	1	1	4	0
6	6	1	1	0	0	0	0	1	1	1	0	0
7	3	1	1	1	1	0	0	0	1	1	3	0
8	3	0	1	1	0	1	0	0	1	1	0	0
9	2	1	0	1	0	0	1	0	1	1	0	0
10	1	1	1	1	1	0	0	0	1	1	1	0

Figure 4 We consider 10 events at the same ATM. Time is running along the horizontal axis and the event displacement is proportional to the elapsed time. For the sake of clarity, the events have been displaced along the vertical axis too. The dashed horizontal lines represent the time window before and after each event, in which events were considered to build the embedding. Vertical dotted lines delimiting the time windows have been drawn for the first and sixth events only

precision and recall [15]. Therefore, this score takes both false positives and false negatives into account. The f_1 score is usually more useful than accuracy score, especially if we have an uneven class distribution, i.e. a large number of actual negative, for the binary classification task.

Concerning the generalization of an algorithm, one might have two view points. First, one could be considered as the success of a model in correctly prediction of unseen data set. For this purpose we compared f_1 and precision computed using increasing percentages of data for training plotted by learning curve. In this way we evaluate how the model can correctly extend predictions to a set of data that is much bigger of the training data set.

Secondly, we note that generalization can be understood in a broader sense, that is a qualitative evaluation of the effort that should be done to adapt a method to a different data set or context, that is a kind of transferability of the methods to other domains. In this sense, it is evident that the entity based methods are more general than the event based one, because they could be used with virtually no changes to classify data from other graphs. On the other hand, the event based method requires appropriate definition of the event types for the context at hand and, as a consequence, a reworking of the pre-processing phase.

With transparency we mean the property of an algorithm of being more or less easily understood by a human, so that one can realize how the model made decisions and arrived at

the results. We will discuss qualitative evaluations of this property, taking into account that the algorithms are composed by two distinct steps: embedding computation and embedding vectors classification. We will consider separately the transparency of each step to compare the overall transparency of the methods.

5.1 Results: entities embedding method

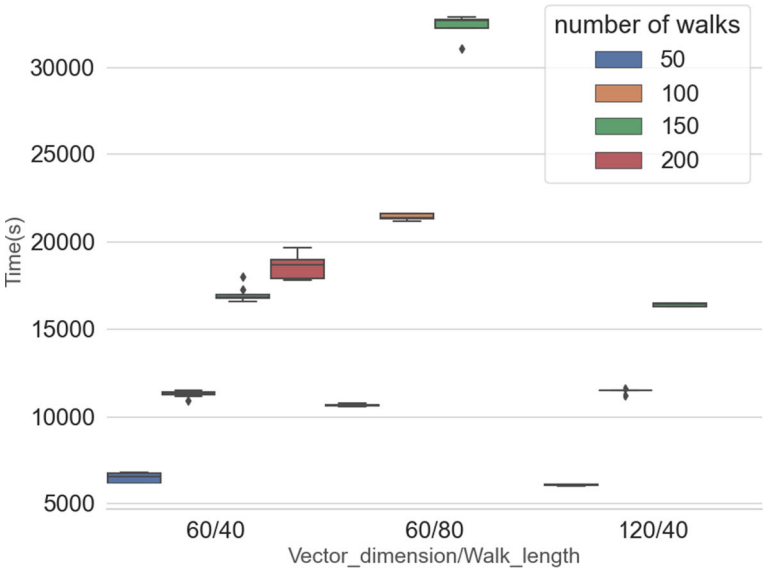
In Section 4.2, we discussed how the embedding algorithms help to find sub-graphs committed to our money laundering scenario by converting the network $\mathcal{N}$ of incorporating instances to vectors. The experimental results of testing parameters for the first method, the pseudo-static network using node2vec algorithm, is demonstrated in Figure 5. The figure contains two parts, the required modelling time, comes in Figure 5a, and the f_1 score obtained by using different parameters is presented in Figure 5b. Considering the hyper-parameters *vector-dimension* and *walk-length* sets on the horizontal axis, we witnessed the needed time for generating embeddings increases by increments in the number of walks. Figure 5b depicts the f_1 score of embedding models for different parameters on the basis of returning the correct number of ATMs involved in a criminal event. This plot represents that the method, in general, is able to correctly recognizes more than 90% of criminal withdrawals.

In the second method, reforming the network in temporal slices, we aimed to reduce the computational complexity in the embedding procedure. Figure 6a and b demonstrate the needed time and f_1 score of the model using different parameters. While the f_1 score for temporal network embedding, in comparison to the first method, decreases as we expected, the needed time of modeling sharply decreases. This reduction is a direct result of the temporal slicing procedure which may cause interruptions in the series of occurring criminal withdrawals and received SMS. However, it is worth mentioning that, we still observe high performances in the case with large length and number of walks. The Figure 7 compares the precision/ f_1 score of pseudo and temporal entity embedding methods.
Using the proNE and VERSE algorithms we confirmed the output results of node2vec algorithms comparing the pseudo-static and temporal embedding. Whilst these algorithms are significantly faster in modelling, the accuracy drastically decreased (For further details and plots of the modelling time and the f_1 score see the Supplementary material).

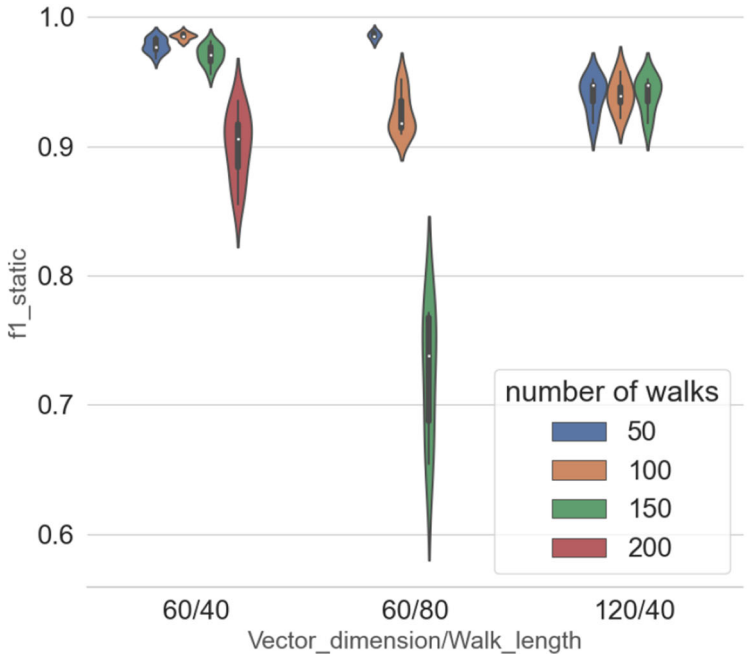
5.2 Results: event embedding method

The event embedding vectors are made according to the description provided in Section 4.3 regarding the money laundering scenario. Leveraging a classification algorithm, event vectors (withdrawals) that contain the spatio-temporal information of withdrawals could be labeled as criminal or non-criminal. To this aim, we feed the "event vectors" to the Support Vector Classifier for classification purpose.

In particular, Figure 8 shows the learning curve for the SVC used to classify the event embedding. The leftmost point uses the same amount of training data as Figure 7a and the rightmost uses the full data set for training as Figure 7b. Comparison of the three panels shows that the event embedding can make correct previsions using a smaller set of data than the entity embedding methods. We therefore conclude that event embedding get high generalization faster that entity embedding, however, all three methods have reasonably high and comparable f_1 and precision scores.

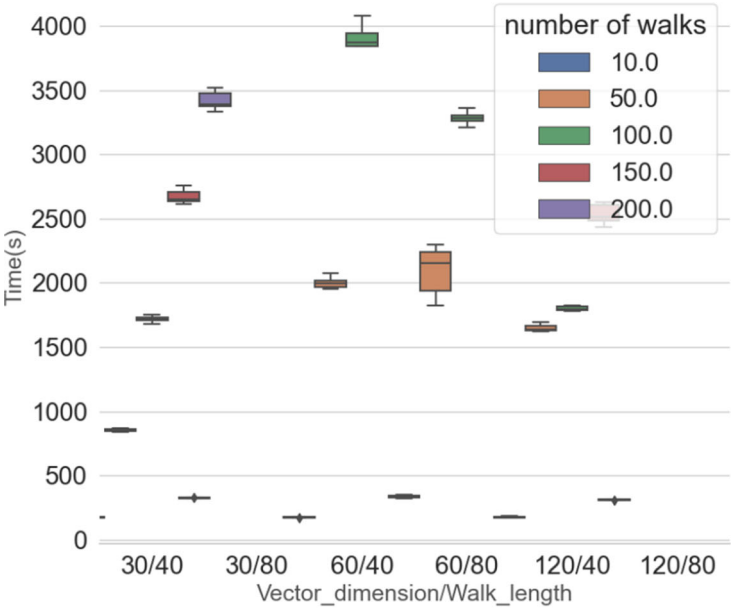


(a)

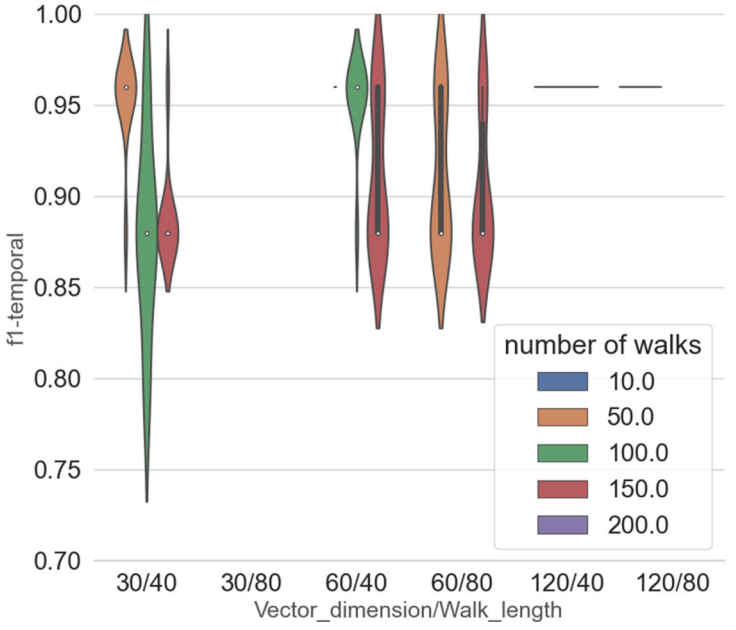


(b)

Figure 5 **a** The experimental results for the needed time of modelling the pseudo-static network is plotted. **b** The f_1 score in detecting correct number of ATMs involved in criminal withdrawals. Different values of parameters in *node2vec* algorithm are tested



(a)



(b)

Figure 6 **a** The experimental results on the time used for modeling Temporal Embeddings is plotted. **b** The F_1 score in detecting the criminal withdrawals at each layer of the network. Different values of parameters in *Node2Vec* algorithm are tested

6 Discussion

In this paper, we have discussed two methods with different perspectives of the embedding. In the *Entities embedding* method, the constituting instances of a knowledge graph are embedded in the latent space according to their sub-graphs structural similarity. In this method, both spatial and temporal dimensions are considered as two different attributes of entities. Considering the results of the pseudo static and temporal approaches, our findings indicate that the spatial dimension of the network could serve the pattern recognition and sub-graph matching task. This structural robustness could help to progress the studies for a large period of time without an obsessive focus on the time dimension and provide an evaluation of the distance between the predicted criminal instances and the original enquery protocol.

On the other hand, in the event embedding method, the phenomenal point of view of the network is much of interest. In this method, the time dimension plays a crucial role while the higher-order structure of the network will not be preserved anymore. This point makes this method precise and fast in recognizing the favorite events (in this study, the criminal withdrawals).

On the other hand, this method requires a domain-dependent pre-processing procedure in "events" definitions before initiating the classification task, which is time-consuming and may cause a high generalization error in the embedding computation procedure.

In Figure 7, a summary of the precision and f_1 score of the proposed methods in terms of relevant hyper-parameters, for our specific case study, are provided. These results indicate the high precision and f_1 score for the entity embedding methods. Instead, Figure 8 demonstrates a learning curve for the event embedding classification task using SVC which has high but at the same time comparable f_1 and precision scores for the different training examples.

Concerning the transparency required in the criminal investigations, as stated in Section 5, we consider separately the steps of embedding computation and embedding vectors classification.

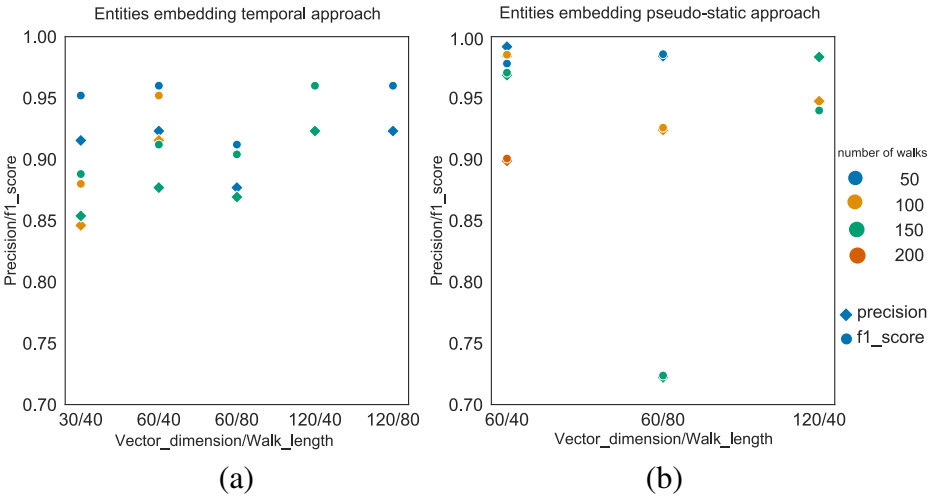


Figure 7 The *precision* and F_1 score of proposed methods. The diagrams represent respectively: **a** the result of *Entities embedding* for temporal approach; **b** the result of *Entities embedding* for pseudo-static approach in terms of a vector of dimension/walk length for different number of random walkers

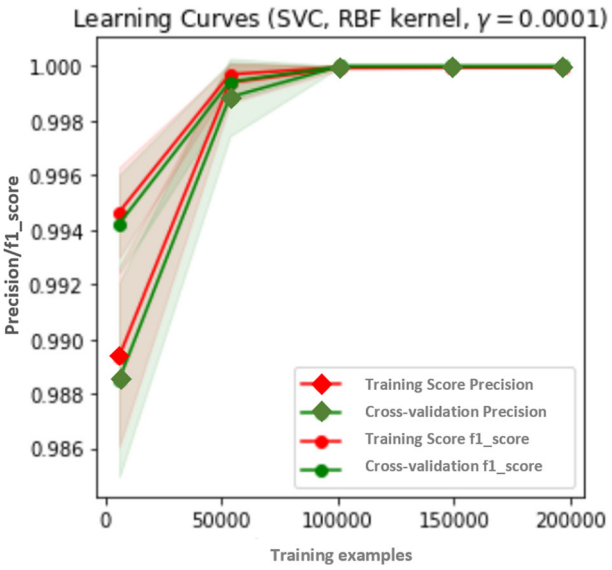


Figure 8 The learning curve of Support Vector Classifier leveraged for the classification task of event embedding algorithm. The leftmost data points demonstrate the f_1 and precision of 3% of training data set comparable with the partial data used for modelling temporal embedding. The rightmost points represent the 100% of the training example data equal to the used data for the pseudo-static entity embedding algorithm

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 which could be intuitive by human.

On the other hand, VERSE learns embeddings by training a single-layer neural network, so that its transparency is lower. The way ProNE enhances the embedding using spectral spaces techniques seems also hardly transparent.

Event embedding computation consists in counting events of some types happening in a time window before and after the event to embed. It is a perfectly human understandable operation as, probably, is what a human agent would do if he/she had to manually find event patterns.

Entity embedding classification is performed using cosine similarity, that can be considered quite an intuitive concept, so easily understood in general.

Event embedding classification may be done using a choice of machine learning methods, such as decision trees, support vector machines, neural networks and so on.

Table 1 Comparing the transparency levels of the proposed methods in terms of transparency in the computation of embedding vectors and classification task. We used a scale: 0 (not transparent), 1 (partially transparent), 2 (fully transparent)

Methods Embedding	Transparency levels
Entities embedding-node2vec	1
Entities embedding-VERSE	0
Entities embedding-ProNE	0
Event embedding-Event series	2

Table 2 Evaluation of Precision, Generalization (that is separated in Generalization err. Transferability), and Transparency of proposed methods. Each method is ranked as their performance in evaluation

		Generalization		
	Precision	Generalization err.	Transferability	Transparency
Entities embedding-node2vec	3°	1°	1°	2°
Entities embedding-VERSE	2°	1°	1°	3°
Entities embedding-ProNE	2°	2°	1°	3°
Event embedding-Event series	1°	2°	2°	1°

Accordingly, the transparency of this step may vary considerably. In our implementation we used a SVC, that can be considered half way in transparency, as one can compute and plot an estimate of the sensitivity of the model to each feature [28].

In Table 2 we summarize the discussion providing the evaluation of three key points in this work, *precision*, *generalization*, and *transparency* of our proposed methods. We split the generalization into two columns, the first according to Figure 7 takes into account the ability of the model of getting low generalization error with a small training data set; the second, that we call transferability, considers the amount of work needed to adapt the model to other contexts.

We consider the temporal and pseudo-static entity embeddings as partial and complete training data set respectively for comparing generalization of entity and event embedding algorithms.

7 Conclusions

Using knowledge graphs and graph queries to integrate criminal data and retrieve operational inquiries that verify the facts according to protocols, is an advancement in justice systems. Graph queries properly return the events in exact correspondence to the criminal patterns specified in the protocols but fails in identifying even small deviations from these patterns.

With this paper, we have suggested two innovative approaches to crime detection using the embedding techniques to retrieve criminal patterns reasonably closed to the patterns specified in the protocols.

To this aim, in the first method, we have applied a hierarchical filtering procedure. According to this procedure, the proximity measures in the embedding space, which quantifies the neighborhood of a given node, filters out the dissimilar sub-graphs in two following steps. At the first step, the structural similarity and at the second step, the higher-order of neighborhood, are considered. Our method demonstrates robust accuracy with different hyper-parameters. It also provided good results in terms of transparency.

As a second method, we have considered the event embedding mechanism in order to detect the favorite series of events in a pre-defined time interval. Even though this method demonstrated high accuracy and robustness, it has the drawback of requiring high computation resources for training and high amount of work to transfer the computed model to other contexts.

References

1. Abiodun, O.I., Jantan, A., Abiodun, E.O., Dada, K.V., Nachaat, A.M., Arshad, H.: State-of-the-art in artificial neural network applications: A survey. Heliyon **4**(11), e00938 (2018)
2. Ashley, K.D.: Artificial intelligence and legal analytics: New tools for law practice in the digital age. Cambridge University Press (2017)
3. Bellandi, V., Ceravolo, P., Maghool, S., Siccardi, S.: Graph embeddings in criminal investigation: Extending the scope of enquiry protocols. In: Proceedings of the 12th International Conference on Management of Digital EcoSystems, MEDES '20, pp. 64–71. Association for Computing Machinery, New York (2020)
4. Benzi, K.M.: From recommender systems to spatio-temporal dynamics with network science. Technical report EPFL (2017)
5. Béres, F., Kelen, D.M., Pálovics, R., Benczúr, A.A.: Node embeddings in dynamic graphs. Applied Network Science (2019)
6. Bertalan, T., Wu, Y., Laing, C., Gear, C.W., Kevrekidis, I.G.: Coarse-grained descriptions of dynamics for networks with both intrinsic and structural heterogeneities. Front. Comput. Neurosci. **11**, 43 (2017)
7. Bjelland, H.F., Dahl, J.Y.: Exploring criminal investigation practices: The benefits of analysing police-generated investigation data (2017)
8. Breslow, L.A., Aha, D.W.: Simplifying decision trees: A survey. Knowl. Eng. Rev. **12**(1), 1–40 (1997)
9. Catanese, S.A., Fiumara, G.: A visual tool for forensic analysis of mobile phone traffic. In: MiFor '10: Proceedings of the 2nd ACM workshop on Multimedia in forensics, security and intelligence, pp. 71–76. ACM (2010)
10. Colladon, A.F., Remondi, E.: Using social network analysis to prevent money laundering. Expert Syst. Appl. **67**, 49–58 (2017)
11. Depeau, J.: Announcing the neo4j crime investigation sandbox. Technical report, Neo4j. https://medium.com/neo4j/announcing-the-neo4j-crime-investigation-sandbox-c0c3bd9e71b1 (2018)
12. Eriksén, S.: Designing for accountability. In: Proceedings of the Second Nordic Conference on Human-Computer Interaction, pp. 177–186 (2002)
13. Felzmann, H., Villaronga, E.F., Lutz, C., Tamò-Larrieux, A.: Transparency you can trust: Transparency requirements for artificial intelligence between legal norms and contextual concerns. Big Data & Society **6**(1), 2053951719860542 (2019)
14. Gehl, R., Plecas, D., et al: Introduction to criminal investigation: Processes, practices and thinking. Justice Institute of British Columbia. https://openlibrary-repo.ecampusontario.ca/jspui/handle/123456789/348 (2018)
15. Goutte, C., Gaussier, E.: A probabilistic interpretation of precision, recall and f-score, with implication for evaluation. In: European Conference on Information Retrieval, pp. 345–359. Springer (2005)
16. Govende, D.: The criminal investigation: principles and practices. Servamus Community-based Safety and Security Magazine **112**(11), 31–33 (2019)
17. Goyal, P., Ferrara, E.: Graph embedding techniques, applications, and performance: A survey. Knowl.-Based Syst. **151**, 78–94 (2018)
18. Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 855–864 (2016)
19. Hildebrandt, M.: Law as computation in the era of artificial legal intelligence: Speaking law to the power of statistics. University of Toronto Law Journal **68**(supplement 1), 12–35 (2018)
20. Holzschuher, F., Peinl, R.: Querying a graph database–language selection and performance considerations. J. Comput. Syst. Sci. **82**(1), 45–68 (2016)
21. Irons, A., Lallie, H.S.: Digital forensics to intelligent forensics. Fut Int **6**(3), 584–596 (2014)
22. Junior, S.B., Ceravolo, P., Damiani, E., Tavares, G.M.: Evaluating trace encoding methods in process mining. In: Bowles, J., Broccia, G., Nanni, M. (eds.) From Data to Models and Back, pp. 174–189. Springer International Publishing, Cham (2021)

23. Kumar, M., Hanumanthappa, M., Suresh Kumar, T.V.: Crime investigation and criminal network analysis using archive call detail records. In: 2016 Eighth International Conference on Advanced Computing (ICoAC), pp. 46–50. IEEE (2017)
24. Leida, M., Ceravolo, P., Damiani, E., Cui, Z., Gusmini, A.: Semantics-aware matching strategy (sams) for the ontology mediated data integration (oddi). International Journal of Knowledge Engineering and Soft Data Paradigms **2**(1), 33–56 (2010)
25. Lundberg, S.M., Erion, G., Chen, H., DeGrave, A., Prutkin, J.M., Nair, B., Katz, R., Himmelfarb, J., Bansal, N., Lee, S.-I.: From local explanations to global understanding with explainable ai for trees. Nature Machine Intelligence **2**(1), 56–67 (2020)
26. Medvedeva, M., Vols, M., Wieling, M.: Using machine learning to predict decisions of the european court of human rights. Artificial Intelligence and Law **28**(2), 237–266 (2020)
27. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv:http://arxiv.org/abs/1301.3781 (2013)
28. Nalbantov, G., Bioch, J., Groenen, P.: . Solving and interpreting binary classification problems in marketing with svms **566–573**, 11 (2005)
29. Narayanan, A., Chandramohan, M., Venkatesan, R., Chen, L., Liu, Y., Jaiswal, S.: graph2vec: Learning distributed representations of graphs. arXiv:http://arxiv.org/abs/1707.05005 (2017)
30. Paulheim, H.: Knowledge graph refinement: A survey of approaches and evaluation methods. Semantic Web **8**(3), 489–508 (2017)
31. Peroncini, R., Pizzi, R.: Values for some: How does criminal network undermine the political system? a data mining perspective. In: Systemics of Incompleteness and Quasi-Systems), pp. 267–282 (2019)
32. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 701–710 (2014)
33. scikit-learn, machine learning in python
34. Tensorflow
35. Torricelli, M., Karsai, M., Gauvin, L.: weg2vec: Event embedding for temporal networks. Scientific Reports (2020)
36. Tsitsulin, A., Mottin, D., Karras, P., Müller, E.: Verse: Versatile graph embeddings from similarity measures. In: Proceedings of the 2018 World Wide Web Conference, WWW '18, pp. 539–548. Republic and Canton of Geneva CHE (2018). International World Wide Web Conferences Steering Committee
37. Wang, Q., Mao, Z., Wang, B., Guo, L.: Knowledge graph embedding: A survey of approaches and applications. IEEE Trans. Knowl. Data Eng. **29**(12), 2724–2743 (2017)
38. Wischmeyer, T.: Artificial intelligence and transparency: Opening the black box. In: Regulating Artificial Intelligence, pp. 75–101. Springer (2020)
39. Zhang, J., Dong, Y., Wang, Y., Tang, J., Ding, M.: Prone: Fast and scalable network representation learning. In: IJCAI, vol. 19, pp. 4278–4284 (2019)
40. Zhang, M., Wang, Q., Xu, W., Li, W., Sun, S.: Discriminative path-based knowledge graph embedding for precise link prediction. In: European Conference on Information Retrieval, pp. 276–288. Springer (2018)