World Wide Web (2022) 25:2403–2428
https://doi.org/10.1007/s11280-021-01002-1

# Event prediction from news text using subgraph embedding and graph sequence mining

Recep Firat Cekinel[1] · Pinar Karagoz[1]

Received: 10 December 2020 / Revised: 19 November 2021 / Accepted: 22 December 2021 /
Published online: 28 February 2022

## Abstract

Event detection from textual content by using text mining concepts is a well-researched field in the literature. On the other hand, graph modeling and graph embedding techniques in recent years provide an opportunity to represent textual contents as graphs. Text can be enriched with additional attributes in graphs, and the complex relationships can be captured within the graphs. In this paper, we focus on news prediction and model the problem as subgraph prediction. More specifically, we aim to predict the news skeleton in the form of a subgraph. To this aim, graph-based representations of news articles are constructed and a graph mining based pattern extraction method is proposed. The proposed method consists of three main steps. Initially, graph representation of the news text is constructed. Afterwards, frequent subgraph mining and sequential rule mining algorithms are adapted for pattern prediction on graph sequences. We consider that a subgraph captures the main story of the contents, and the sequential rules indicate the subgraph patterns' temporal relationships. Finally, extracted sequential patterns are used for predicting the future news skeleton (i.e. main features of the news). In order to measure the similarity, graph embedding techniques are also employed. The proposed method is analyzed on both a collection of news from an online newspaper and on a benchmark news dataset against baseline methods.

**Keywords** Graph mining · Sequential rule mining · Frequent subgraph mining · Graph embeddings · News prediction

## 1 Introduction

In the literature, *event* is generally defined as an incident that happen at a certain time and place and attracts people's attention [3, 34]. Various studies consider variations of the basic definition such that the event may last during a time interval rather than a short incident

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

✉ Recep Firat Cekinel
rfcekinel@ceng.metu.edu.tr

[1] Computer Engineering Department, Middle East Technical University, Ankara, Turkey

associated with a time instance. Similarly, some other studies neglect the place of the event, specifically for the virtual cases such as a show broadcasted on television or on the web. In either case, event is a happening that attracks attention and effects the circumference and the behaviour of people.

Event detection and prediction are well-known information extraction problems. Earlier approaches on event detection focus on text processing, mostly text clustering [3, 19, 34], aiming to early detection of events from mostly social media posts. Event prediction is a comparatively more recent problem which focuses on predicting the features of the event, such as time, place, for a future time window. Many of the studies focus on event prediction for specific domains, such as social unrest, flood events, or business failures [47].

Advancements in the graph modeling and graph mining techniques open up new opportunities. By representing a text as a graph, more complex features and relationships among the terms and text can be captured [10, 37]. Such graph representation for text provides potential to capture and analyze events for effectively.

In this work, we focus on event prediction problem. Specifically, we aim to extract event patterns from news documents and use these patterns for predicting the future event. Some events can be considered as *unexpected*, such as the Covid-19 pandemic or a plane crash, since there may not be any pattern they take part in. In this work, rather than such unexpected events, we aim to predict events based on patterns extracted from event history. Financial events, political or commercial events resulting from earlier actions or status are feasible candidates for the cases that involve an *event pattern*. For instance, central bank committee meetings, or FOMC meetings in the USA, are organized periodically. The regular patterns such as the currency and stock price movements close to the meeting dates can be explored the associations of such cases can be examined.

In the proposed approach, we represent the core features of an event, extracted from news text, in the form of a graph. Hence an event pattern involves a sequence of graphs and a sequence rule extracted from graph sequences. On the basis of this, we consider event prediction as a *graph prediction* problem.

The proposed method includes the steps of modeling a text as a graph, extracting subgraphs denoting the cores of news (i.e., events), and sequence pattern extraction from subgraph collection. Lastly, the extracted patterns are used for event prediction. As the first step, a given news text is modeled as a graph structure. Afterwards, frequent subgraphs are discovered with the assumption that subgraphs are the core components of news and repeated subgraphs can be an indicator for an event.

Subsequently, we form subgraph sequences with respect to news timestamps and then apply sequence mining to discover sequential patterns. We use these patterns in order to predict the core structure of an event to happen in the future time window. To this aim, we compare the similarity between the subgraph structure in the current time window and the extracted patterns. Since we use graph structure as the primary data source, the similarity measurement is calculated over the graph embeddings.

In other words, in this work, we aim to find the patterns of temporal relationships among news events and predict the core structure of the news (event) that may follow

by considering the previous patterns.

Unlike the conventional approaches that work on the textual resources directly, we represent each news in the form of a graph and generate a news prediction model that accepts graph structure as the input and executes graph analytics algorithms to extract relationships between graphs.

The basic contributions of this work are listed below:

- We propose an event prediction method, which predicts the core structure of the event to happen in the future time window, by making use of graph sequence based event patterns.
- We propose a graph representation for a given news text describing an event. It is assumed that the generated graph represents the core structure of the event described in the news text.
- On the basis of the graph based event representation, we propose an event pattern model in the form of sequence of subgraphs extracted from news graphs.
- Several graph and subgraph embedding methods are elaborated on to find the similarity measurement approach that can best capture the human perception for news similarity. The result of this analysis is used for matching the current news graph structure with the extracted patterns.
- The proposed method is analyzed on both a rich collection of news from the website of a newspaper and a benchmark news dataset against baseline methods.

The rest of this paper is organized as follows. The mathematical and technical concepts of the proposed model are presented in Section 2. Our methodology, data model and experiments are presented in Sections 3 and 4, respectively. The related studies are summarized in Section 5. Finally, the overview of the study and the future work are given in Section 6.

## 2 Preliminaries and problem definition

In this section, technical background about the proposed method and the problem definition are presented.

### 2.1 Graph representation

In this work, we model news text as an unweighted directed graph such that nodes represent the terms in a news document with high (above the user-specified threshold) TF-IDF scores, Additionally we include a unique hyper-node for each news. For term filtering, we use TF-IDF score as it is often used in data mining tasks to indicate a word's importance in a corpus [31].

The graph includes two types of relationships, *Contains* and *Followed_By*. *Contains* relationship links the unique *News* hyper-node to the *Term* vertices. *Followed_By* relationship exists between consecutive term nodes. The direction of the *Followed_By* relationship indicates the order of the terms in the news text. A sample graph structure is presented in Figure 1.

### 2.2 Frequent subgraph mining

Frequent subgraph mining is a task that aims to find all the subgraphs within the collection of graphs that occur more than a user-defined support threshold (*minsup*) [4].

Apriori-based subgraph mining algorithms generate candidate subgraphs just like the Apriori-based frequent itemset mining algorithms. They perform a level-wise search to find all frequent subgraphs. Therefore, a join operation has to be applied for joining two k-size subgraphs, which is a costly operation. On the other hand, pattern-growth based algorithms

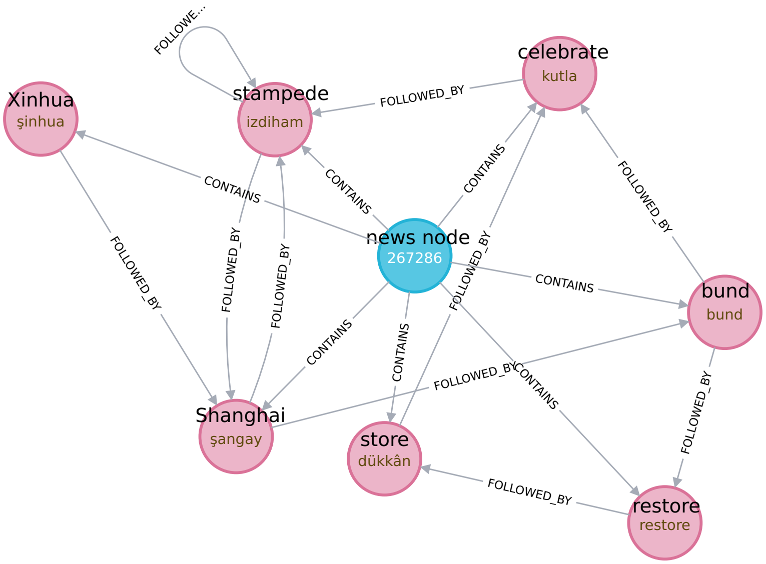


**Figure 1** Graph model of news text

do not need join operation. Instead, they add an edge to the existing subgraph at each iteration.

Due to its runtime efficiency, in this work, we use the gSpan [42] algorithm, which utilizes Depth First Search (DFS) codes and lexicographical ordering to eliminate redundant candidate generation and costly graph isomorphism test. The main purpose of using DFS codes is to express edges and vertices.

## 2.3 Sequential rule mining

Sequential rule mining aims to extract interesting rules from a sequence database, where a sequence is formed by a group of items (i. e., *sequence of itemsets*). Sequential patterns are subsequences that exist in a sequence database. A sequence consists of an ordered list of itemsets where each itemset has an unordered list of items. In other words, there is a time-based order between each itemset, but items within an itemset are not ordered [9].

A sequential rule is in the form of $A \rightarrow B$ where

both $A$ and $B$ are frequent sequences that are extracted as sequential patterns by a sequential pattern mining algorithm. The rule implies that $B$ is likely to happen after $A$, so a strong temporal relationship between $A$ and $B$ is implied.

The sequential rule mining can be decomposed into two subtasks:

1. **Sequential pattern mining:** This subtask discovers sequential patterns that exist frequently in a sequence database. In our work, we use the PrefixSpan [14] algorithm to extract frequent sequences. PrefixSpan [14] is a pattern-growth based frequent sequential pattern mining algorithm that extracts frequent sequences recursively starting from single-item sequences towards larger sequences using depth-first search.
2. **Sequential rule generation:** The second subtask performs the extraction of interesting rules by using the frequent sequences that were extracted in the first step. The rules are considered to be strong if the confidence value is above a user-defined *minconf*

threshold. One can also eliminate the rules with respect to additional criteria such as lift. In this work, we use the RuleGen [44] algorithm, which extracts all sequential rules having confidence above a threshold, *minconf*, that is adjusted by users.

## 2.4 Subgraph embeddings

We represent subgraphs as a lower-dimensional vector in order to measure the similarity between subgraphs. This also constitutes the basics for similarity calculation for subgraph sequences and the sequential subgraph rules. We use node embedding (*node2vec* [11]), subgraph embedding (*Sub2Vec* [1]) and graph embedding (*graph2vec* [28]) techniques to represent frequent subgraphs as vectors. In addition to vector based similarity calculation, there are also other methods such as graph edit distance [36] that measures the similarity of two graphs using minimum graph operations required to transform one graph into another. However, it is proven that computing the edit distance is NP-hard [45]. We can summarize the subgraph embedding techniques elaborated on in this study as follows:

- **One-hot Encoding**: Graph one-hot encoding is a bag-of-words model for graphs that represents each subgraph as a vector. It is a simple model and the vector size should be at least as large as the size of the vocabulary. In the vector, weight value of the terms that appear in the subgraph is set as 1. One-hot representation of two subgraphs are similar as long as they contain the same terms. Therefore, it is not able to capture semantic relationships between the terms.
- **Node2vec**: Node2vec [11] uses short random walks to learn node embeddings similar to DeepWalk [29]. The main difference between DeepWalk and node2vec is that DeepWalk uses pure random walks. However, node2vec uses biased random walks, and hence we can control randomness of each walk with an configurable parameter. Both node2vec and DeepWalk are inspired by the skip-gram model of word2Vec [24].

  Node2vec produces a vector for each node, so we take the sum of node embeddings that appear in the subgraph to calculate the subgraph's embedding. Note that the same node may have more than one vector representation in different news graphs. There can be more than one node2vec vector for the same subgraph with respect to its context.
- **Sub2Vec**: Sub2Vec [1] is another random walk based method that learns the latent representation of a subgraph. It can learn a representation for arbitrary subgraphs that preserve second-order proximity or structural equivalence. The second-order proximity is related with local connectivity of a subgraph, and the structural equivalence refers to the overall structure of the subgraph.
- **Graph2vec**: Graph2vec [28] is a neural embedding model that learns the representation of a whole graph using non-linear substructures, particularly rooted subgraphs. It is inspired by the idea of Doc2vec [20], a framework that learns the latent representation of a document by using word embeddings.

  Graph2vec views the whole graph as a document and rooted subgraphs as words of the document, and trains these vectors to learn the lower-dimensional representation of the graph structure. In our method, we use the graph2vec model to obtain latent representation of frequent subgraphs.

## 2.5 Problem definition

On the basis of the preliminaries, we can define the basic concepts of the proposed method as given below.

**Definition 1** (**News Graph**) A *news graph*, NG <V, E> is an unweighted, directed graph such that V is the set of vertices (nodes) and E is the set of edges. V includes a unique news node $n$, a set of nodes $t$ that correspond to the terms in the news. As described earlier, a filtering is applied on the set of terms with respect to TF-IDF score. An edge $e \in E$ denotes either the ordering relationships between two terms (*Followed_By*) or inclusion relationship between a term and a news (*Contains*).

**Definition 2** (**Event Skeleton**) *Event Skeleton*, $S$, is a frequent subgraph of news graphs that contain the essential information about an event. We consider such subgraphs as the base components describing events. Event skeleton refers to general event structure rather than a particular instance. For example, a particular case where the rise in price of a particular stock's price rises at a certain time with a certain price, event skeleton may include the nodes referring to the stock and increase in price describing a more abstract behavior.

On the basis of news graph and news skeleton definitions, we can describe the problem we challenge in this work as given below.

**Problem definition** Given a set of news text $\{T_i, ..T_n\}$ each with a timestamp, it is aimed to predict the event skeleton $S$ for the future time window.

## 3 Proposed method for event prediction from news text

Given a set of news document, pipeline of the proposed method for event skeleton prediction is presented in Figure 2. As given in the pipeline, textual news data is preprocessed and

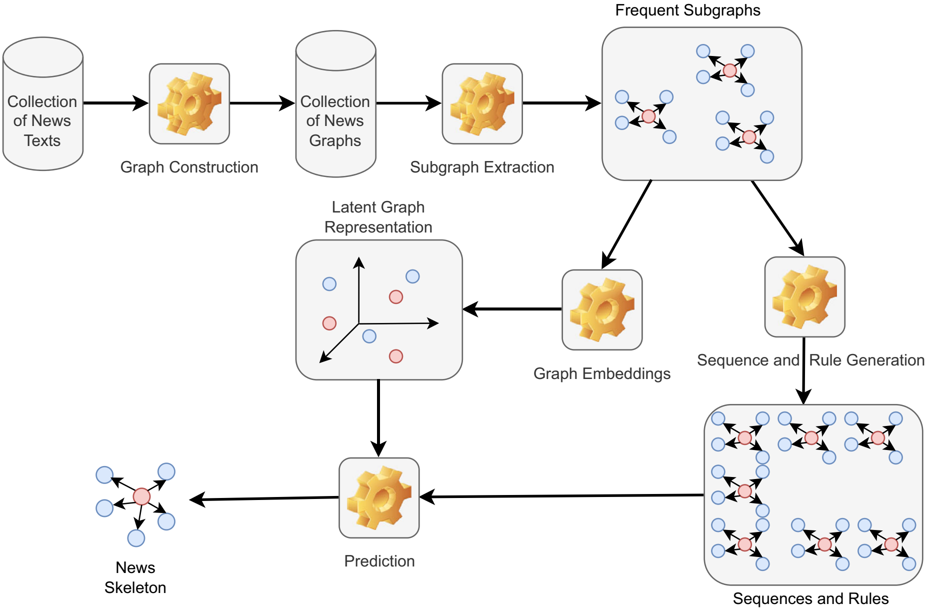


**Figure 2** Pipeline of the proposed model

a graph structure is constructed per news. Following this, graphs are grouped under predefined time windows, and the gSpan [42] algorithm is run for each time window to extract frequent subgraphs. The frequent subgraphs indicate the skeleton of important patterns that occur within these periods. In the next stage, to discover the temporal patterns in the events, subgraph sequences are generated. In these sequences, each subgraph can be considered as an item and subgraphs that happen on the same day form an itemset. The PrefixSpan [14] and the RuleGen [44] algorithms are applied to discover the sequential rules within the sequence transactions. Finally, in the prediction stage, given a set of recent news, their graph structures are compared with the antecedents of the event patterns in the rule set. At this point, we encode frequent subgraphs using graph embedding methods and measure the similarity between subgraphs through their embeddings. We consider the matching rule's consequent as the event skeleton prediction for the future time window.

Basically, our model predicts event in the form of subgraphs, denoted as *event skeleton*, where these subgraphs represent the core elements of an event. Therefore, our model discovers the upcoming events by extracting the sequential rules from news history.

## 3.1 News graph construction

In order to determine the graph model to be employed in the study, we elaborated on several alternatives such as including names entities as nodes, including all nouns and verbs in the text as nodes and filtering the terms to include as nodes based on TF-IDF values. On the basis of validation experiments conducted on a sample set, the TF-IDF graph structure is observed to capture the event context better than the others. Besides, both the noun-verb and named entity graph models output very similar subgraph patterns, which are not very discriminative and descriptive. Notably, the noun-verb model's recommendations contain many non-identifier terms failing to reflect the story of the event well. Therefore, in our method, we use the TF-IDF score filtering based graph model to represent news text.

## 3.2 Event skeleton extraction

Frequent subgraph mining is a task that finds all the subgraphs within a collection of graphs that occur more than user-defined support threshold, *minsup* [4].

At this step, we use the gSpan [42], which is a pattern-growth based frequent subgraph extraction algorithm. It utilizes DFS codes and lexicographical ordering to eliminate redundant candidate generation and costly graph isomorphism test. The main purpose of using DFS codes is to represent edges and vertices effectively.

We divided the news graph collection into configurable time windows and the gSpan [42] algorithm is applied for each time window separately to extract frequent subgraphs. By this way, we obtain the event skeletons per window.

## 3.3 Event pattern extraction

In order to extract event patterns, it is needed to capture the temporal ordering among the event skeletons. To this aim, as the first step, frequent subgraphs in the same time window are ordered with respect to the timestamps of the news documents they belong to. By this way we obtain itemset sequences such that an item is a subgraph, and an itemset is a set of frequent subgraphs within a time window.

As an example, let S = $< \{g_1, g_2, g_3\}, \{g_1, g_4\}, \{g_5, g_6, g_7\}, \{g_4, g_5, g_8, g_9\}, \{g_7, g_9\}, \{g_6, g_7, g_8\}, \{g_9\} >$ be a sequence of subgraph sets such that each $g_i$ is a frequent subgraph. There is a temporal ordering between the itemsets with respect to the order of the time windows. By this way, we obtain a sequence database. On this sequence collection, the PrefixSpan [14] algorithm is applied in order to find all frequent sequences. For sequential rule extraction, a modified version of the RuleGen [44] algorithm is used with user-defined *minsup* and *minconf* thresholds. Our implementation of the PrefixSpan and the RuleGen algorithms are based on the SPMF library [8]. Note that we modified the RuleGen algorithm such that the discovered rules are generated in the form of $A \rightarrow (B - A)$ (i.e., the itemsets of the antecedent and the consequence of a rule are disjoint).

**Algorithm 1** subgraphSim (subgid1, subgid2, matchAnte, emb).

```
input  : subgid1: Subgraph id of the first subgraph
         subgid2: Subgraph id of the second subgraph
         emb: indicator for the graph embedding technique
output: cosine similarity of subgid1 and subgid2
vec1 ← getEmbedding(subgid1, emb)
vec2 ← getEmbedding(subgid2, emb)
if emb == "NODE2VEC" then
    maxSim ← -1
    for K in range(len(vec1)) do
        for j in range(len(vec2)) do
            enc1 ← vec1[k], enc2 ← vec2[j]
            sim ← cosineSim(enc1, enc2)
            if maxSim < sim then
                maxSim = sim
            end
        end
    end
    return maxSim
end
return cosineSim (vec1, vec2)
```

### 3.4 Latent subgraph embedding extraction

For the prediction phase, we need to find the similarity between subgraphs of the recent news in the observation time window and the patterns generated in the form of sequence rules. To this aim, we generated a rule-based subgraph matching algorithm that discovers similar pairs of our rules and the observed sequences. For similarity calculation between two subgraphs, we represent subgraphs in a lower-dimensional space and assign a vector for each. We apply node embedding (i.e. node2vec [11]), subgraph embedding(i.e. Sub2Vec [1]) and graph embedding (i.e. graph2vec [28]) techniques to represent frequent subgraphs as vectors. The performance analysis details under each of these embeddings are given in Section 4.

**Algorithm 2** ruleMatching (sequence, ruleAnte, matchAnte, sim_thres, emb, day).

```
input  : seq = ([$S_1$, $S_2$], [$S_1$, $S_3$, $S4$], ..., [$S_n$]): subgraphs that appear on the same day
         are located at the same index.
         ruleAnte : either antecedent or consequent of a sequential rule
         matchAnte : it will be assigned to the matched part of the sequence
         sim_thres : cosine similarity threshold
         emb: indicator for the graph embedding technique
         day: points to the current itemset of the sequence
output:  Integer value that indicates whether there is a match or not
i ← 0 ;
while day < len(sequence): do
    itemset ← [], seqIdx ← 0, ruleIdx ← 0;
    while i < len(ruleAnte) and seqIdx < len(sequence[day]) and ruleIdx <
    len(ruleAnte[i]): do
        sim = subgraphSim(ruleAnte[i][ruleIdx], sequence[day][seqIdx], emb)
        if sim > sim_thres then
            ruleIdx += 1
            itemset.append(sequence[day][seqIdx])
            if ruleIdx == len(ruleAnte[i]) then
                matchAnte.append(tuple(itemset))
                i += 1
            end
        end
        seqIdx += 1
    end
    day += 1
    if i == len(ruleAnte) and ruleIdx == len(ruleAnte[i-1]) then
        return day;
    end
end
return -1
```

### 3.5 Subgraph similarity and rule matching algorithms

The event skeleton prediction relies on the assumption that there is an association between the recent events given in the news within the observation time window and the forthcoming event in the future time window. Therefore we search for sequence patterns in the observation window that are similar to the antecedent of a sequential rule, such that the consequence of the same rule can be used for prediction. Instead of working directly on the complex structure of the graphs, we represent them in a lower-dimensional space as vectors, and measure the cosine similarity between the embeddings of the subgraph sequences in the rule set and in the observation time window.

In the similarity calculation process, we calculate the similarity between two subgraphs as given in Algorithm 1. In the algorithm, if the embedding method is chosen as *node2vec*, the *getEmbedding* method returns a list of vectors for the given subgraph. *Node2vec* method may generate different embeddings for the same node with respect to the context. Consequently, a subgraph may have several embeddings resulting from variations in the embeddings of its nodes. We measure the similarity of two subgraphs by considering all vectors of each subgraph in a pairwise manner and return the highest similarity score as the result. For other embedding methods, since each subgraph has a single embedding, we directly use these values to measure the cosine similarity between subgraphs.

Since our sequences consist of several subgraph sets, we calculate the cosine similarity of subgraphs in each rule's antecedent and the subgraphs in the observation sequence one by one. If the similarity score is higher than the *sim_thres*, we consider them as similar subgraph pairs and continue to seek matching for the the rest of the sequences. By this way, we guarantee that a similar pattern in the sequence does not violate the rule's temporal order. The algorithm for similarity calculation for a pair of sequences is given in Algorithm 2. If we can match a rule antecedent with the sequence of observation window, then we present the rule's consequent as the prediction of the event skeleton for the future time window.

## 4 Experiments and results

### 4.1 Dataset

In this study, we use the news collection retrieved from the online Turkish newspaper *Dünya Gazetesi*[1] It mostly covers financial and business related news. Our dataset contains 31785 news in total collected from 01.01.2015 to 31.12.2016. Each record includes a unique news id, timestamp, a title, a short summary and full content. In our experiments, we utilize the entire textual content and the timestamp.

Additionally, we compared our model's performance with the baselines on the CC-News dataset which was released by Sebastian Nagel [25]. The original dataset contains 708241 news articles from January 2017 to December 2019. In our work, we used a subset of the collection for 2017. The subset used in our experiments includes 18104 news, which contains maximum 75 news per day.

### 4.2 Experimental setting

In the study, all experiments are run on a Windows 10 Pro machine with an Intel(R) Core(TM) i7-4700MQ processor running at 2.4 GHz with 12 GB of memory.

The proposed method's settings for the experiments are as follows. For news graphs, we used TF-IDF filtering based graph model. For the Turkish news dataset, we used the news from January 2015 to September 2016 for training and October 2016 to December 2016 for testing. The training dataset consists of 26067 news and the test set contains 5718 news.

The frequent subgraphs are obtained by gSpan algorithm for a collection of news per month. Hence the time window size is set as one month. Since the number of subgraphs per month varies, dynamic support values are determined for each month. The number of frequent subgraphs and support thresholds per month for Turkish news dataset are presented in Figures 3 and 4, respectively.

The gSpan algorithm may extract redundant subgraphs that are the mirror image of each other. In order to eliminate duplications and to reduce the number of subgraphs, we apply clustering on subgraphs. It is applied such that each subgraph is encoded as a one-hot vector and the vectors having cosine similarity higher than 0.8 are grouped together. Then, we take one sample instance from each cluster randomly.

[1] https://www.dunya.com/

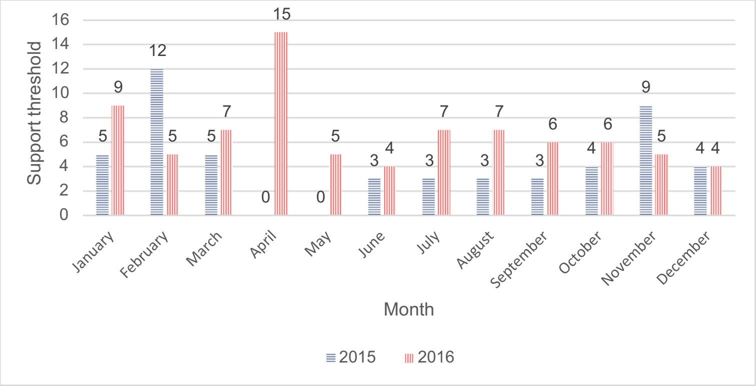


**Figure 3** Support thresholds in each month

Subgraphs whose timestamps belong to the same day form an itemset. The length of the sequences is defined as a parameter and it varies in different experiments. For extracting the graph sequence rules, the support threshold is set as 0.5 and the confidence threshold is set as 0.66. The threshold values are obtained through validation experiments. According to our observations, under these threshold values we can discover the optimum number of rules in a reasonable time period.

For the evaluation, we adapt the conventional precision, recall and f1-score metrics for our setting. To this aim, we define the terms *antecedent matching (ante_match)* and *consequent matching (cons_match)* as follows.

**Definition 3** (**Antecedent_Match**) It is the number of matchings in the test sequences to the antecedent of any rule.

**Definition 4** (**Consequent_Match**) It is the number of matchings for the consequent with the test sequences for those rules whose antecedent already matched with the test sequences.

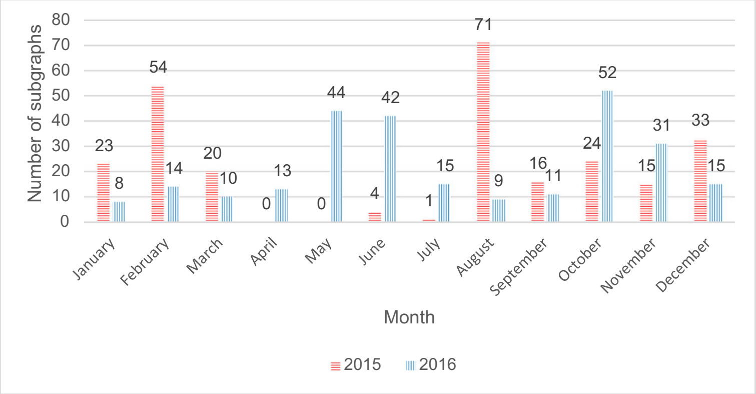


**Figure 4** Number of frequent subgraphs in each month

By using these definitions, precision, recall and f1-score metrics are adapted as given (1), (2) and (3).

$$precision = \frac{Consequent\_Match}{Antecedent\_Match} \tag{1}$$

$$recall = \frac{Consequent\_Match}{Total\ Sequences\ (in\ the\ Test\ Set)} \tag{2}$$

$$f1 - score = 2 * \frac{precision * recall}{precision + recall} \tag{3}$$

### 4.3 Experiment 1: user study on graph embeddings for subgraph similarity

In order to determine the most applicable graph embedding techniques to represent news graphs, we conducted a user study with the participation of six reviewers. We provided 20 news subgraphs selected randomly from our data set, and the reviewers are asked to select the most similar subgraph from the provided choices. We provided four subgraph options that were the most similar news subgraphs of one-hot encoding, graph2vec, sub2vec, and node2vec embeddings, determined under cosine similarity. We conducted the survey by two groups. The reviewers in the first group did not see the content of the news. On the other hand, the second group was allowed to see the news text for each question.

The evaluations of the reviewers are presented in Table 1. According to the responds of both the first and second groups, one-hot encoding and node2vec embeddings generate more similar outputs with the given news subgraps. Thus, in the rest of the experiments, we use on the one-hot encoding and node2vec embeddings.

### 4.4 Experiment 2: analyzing the effect of sequence length on the prediction accuracy

In this experiment, we analyze the effect of the sequence length of the extracted patterns on the prediction accuracy. To that end, we formed sequences of lengths 5, 7, and 10, which we denote as *Seq5*, *Seq7*, and *Seq10*, respectively. Note that, we extract the frequent subgraphs for each month and group them per day. Hence the longest sequence we can get has the size

**Table 1** Graph embedding survey results

| Survey | | One-hot | Graph2vec | Sub2vec | Node2vec |
|---|---|---|---|---|---|
| Blind review w/o news content | Reviewer 1 | **14** | 7 | 12 | 10 |
| | Reviewer 2 | **17** | 7 | 12 | **17** |
| | Reviewer 3 | **17** | 7 | 11 | 16 |
| Review with news content | Reviewer 4 | 11 | 4 | 9 | **15** |
| | Reviewer 5 | **13** | 3 | 9 | 10 |
| | Reviewer 6 | **18** | 9 | 12 | **18** |
| Average | | **15.0** | 6.2 | 10.8 | **14.3** |
| Percentage (%) | | **75** | 31 | 54 | **71.5** |

**Table 2** Sample *Seq5*, *Seq7*, and *Seq10* with varying granularities

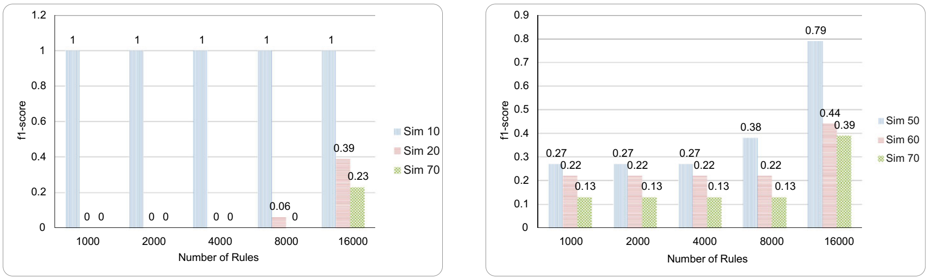
| Sequence | Seq5 days | Seq7 days | Seq10 Days |
|---|---|---|---|
| S1 | 1 2 3 4 5 | 1 2 3 4 5 6 7 | 1 2 3 4 5 6 7 8 9 10 |
| S2 | 4 5 6 7 8 | 5 6 7 8 9 10 11 | 6 7 8 9 10 11 12 13 14 15 |
| S3 | 7 8 9 10 11 | 9 10 11 12 13 14 15 | 11 12 13 14 15 16 17 18 19 20 |
| S4 | 10 11 12 13 14 | 13 14 15 16 17 18 19 | 16 17 18 19 20 21 22 23 24 25 |
| S5 | 13 14 15 16 17 | 17 18 19 20 21 22 23 | 21 22 23 24 25 26 27 28 29 30 |
| S6 | 16 17 18 19 20 | 21 22 23 24 25 26 27 | 26 27 28 29 30 31 |
| S7 | 19 20 21 22 23 | 25 26 27 28 29 30 31 | |
| S8 | 22 23 24 25 26 | | |
| S9 | 25 26 27 28 29 | | |
| S10 | 28 29 30 31 | | |

as many as the number of days in a month. However, this single sequence is too long and too few to extract effective event patterns.

While constructing the sequences, we shift the sequence windows according to the sequence parameter. The sequence structures with varying sequence lengths are shown in Table 2. The reason for using varying amount of shifts for different sequence lengths is to prevent repetition bias towards overlapping days. For *Seq5*, the shift parameter is set as 3, whereas it is set as 4 for *Seq7*, as 5 for *Seq10*. If we had selected a lower granularity value for *Seq10*, itemsets for the days in the middle of each month would appear more than twice, which may cause a bias towards these itemsets during rule generation.

Once the sequence rules are extracted, we further filter the rules and retained the topmost 1000, 2000, 4000, 8000 and 16000 rules according to the interestingness metric. The reason for putting a limit on the rules is to examine the effect of sequence length closely. We measured the interestingness of a rule by using (4). As given in the equation, interestingness score of a rule is calculated through confidence of the rule and probability/support (Pr) of the rule consequent. The rules with high-interest values (above 0.5) are considered to be promising [31].

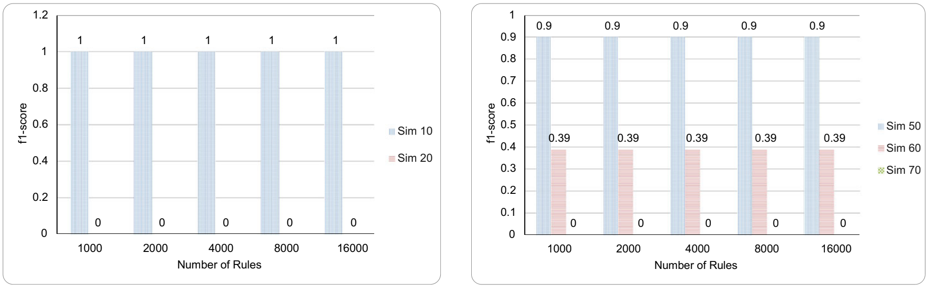
$$Interest(I \rightarrow j) = |confidence(I \rightarrow j) - Pr[j]| \tag{4}$$



(a) One-hot encoding

(b) node2vec

**Figure 5** F1-scores vs. top k interesting rules for *Seq5*



(a) One-hot encoding

(b) node2vec

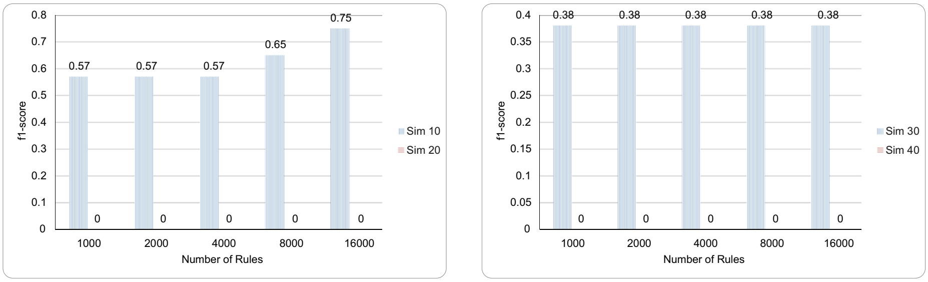
**Figure 6** F1-scores vs. top k interesting rules for *Seq7*

The event skeleton prediction performance of the proposed method in f1-score under varying number of rules for *Seq5*, *Seq7*, and *Seq10* are given in Figures 5, 6 and 7, respectively. We measure the change using different cosine similarity thresholds. For instance, the legend *Sim 20* in the figures indicates the case where the cosine similarity threshold is set as 0.20. According to the experiments, f1-scores of both one-hot encoding and node2vec models are improved as the number of rules increases. Besides, as we decrease the cosine similarity threshold, the f1-score increases.

According to the experiment results, the highest f1-scores are achieved with *Seq5*. Both the one-hot encoding model and the node2vec model performed the best when we used the top 16000 interesting rules. Additionally, we conducted another experiment by using all sequential rules. The results of this experiment are presented in Figure 8. According to the results, *Seq10* outperformed the other models under both one-hot encoding and node2vec embeddings. Note that, we extracted more rules with *Seq10* than the other configurations. It appears that rule size plays a crucial role, but it is not the only factor in the prediction performance. When we restricted the number of rules, *Seq10* performed the worst, whereas Seq5 was the best performing sequence configuration.

### 4.5 Experiment 3: prediction accuracy comparison against the baseline methods

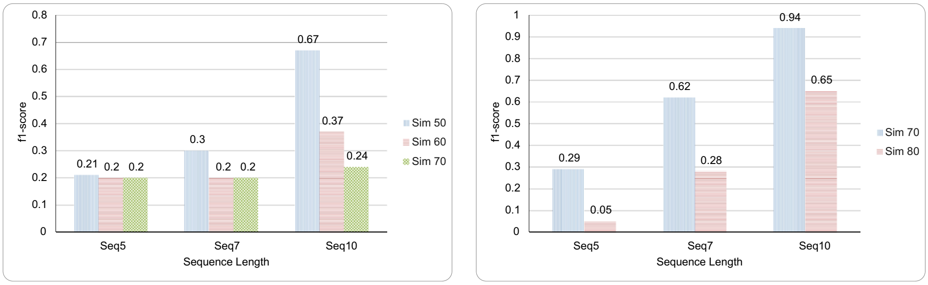
In this experiment, we compare the proposed method's prediction performance against three baseline methods.



(a) One-hot encoding

(b) node2vec

**Figure 7** F1-scores vs. top k interesting rules for *Seq10*



(a) One-hot encoding (b) node2vec

**Figure 8** F1-scores vs. sequence length for various similarity thresholds using all rules

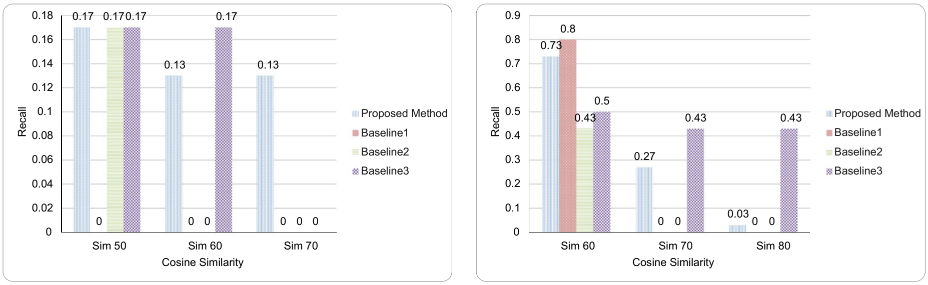
*Baseline1.* The first baseline method is using the most frequent subgraphs as the prediction.

*Baseline2.* The second baseline method is based on presenting the most frequent subgraphs that occur in the most recent month of the training data set. In our case, the last month of the training data set is September 2016.

*Baseline3.* The final baseline method is using the most frequent sequences within the most recent month of the training data set.

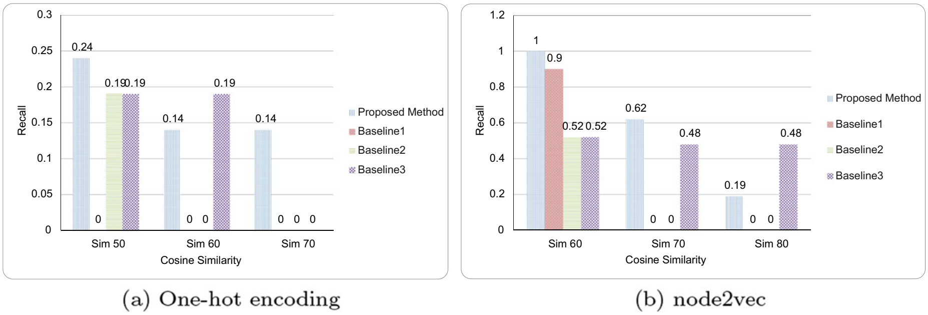
Since the baseline methods do not involve any sequence rule, the precision metric is not applicable for them. For this reason, we use the recall metric for performance comparison. In the experiments, we use one-hot encoding and node2vec embeddings with varying cosine similarity thresholds. Experimental results for *Seq5*, *Seq7* and *Seq10* are presented in Figures 9, 10, and 11, respectively.

Experiments show that proposed method provides higher prediction performance with *Seq10* than the baselines for both one-hot encoding and node2vec embeddings for all similarity thresholds, except for 0.60 where there is an equality with *Baseline3*. With *Seq7*, it also perform better than *Baseline1* and *Baseline2* in all configurations. However, *Baseline3* outperform with *Seq7* using node2vec embeddings under Sim 80 and using one-hot encoding under Sim 60. The findings imply that when we increase the sequence length, the matching rate also tends to rise. This is basically due to mechanism devised in the rule matching algorithm. The algorithm searches for a matching for each subsequence which means that the increase in the sequence length may also enhance chance of the match.



(a) One-hot encoding (b) node2vec

**Figure 9** Recall vs. cosine similarity for Seq5 against the baselines



(a) One-hot encoding (b) node2vec

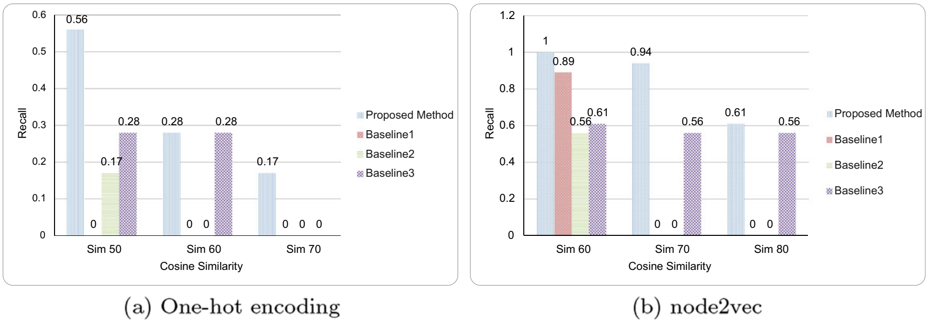
**Figure 10** Recall vs. cosine similarity for Seq7 against the baselines

Moreover, the number of unique rule antecedents and rule consequents may also affect the prediction success since our pairwise algorithm tries to find a similar pattern using rule antecedents. The chance of finding a similar pattern rises as we increase the number of unique rule antecedents.

We also evaluate the proposed method's performance on the CC-News dataset against the baselines. We sampled at most 75 news documents for each day to mitigate the bias towards specific days having too high news. The news published between January 2017 and September 2017 are used for training and the news published in the next three months are used for testing. The training set contains 13774 news instances and the test set includes 4330 news instances. We perform the experiments using node2vec embedding and cosine similarity to measure the similarity between sequential rules and subgraph sequences.

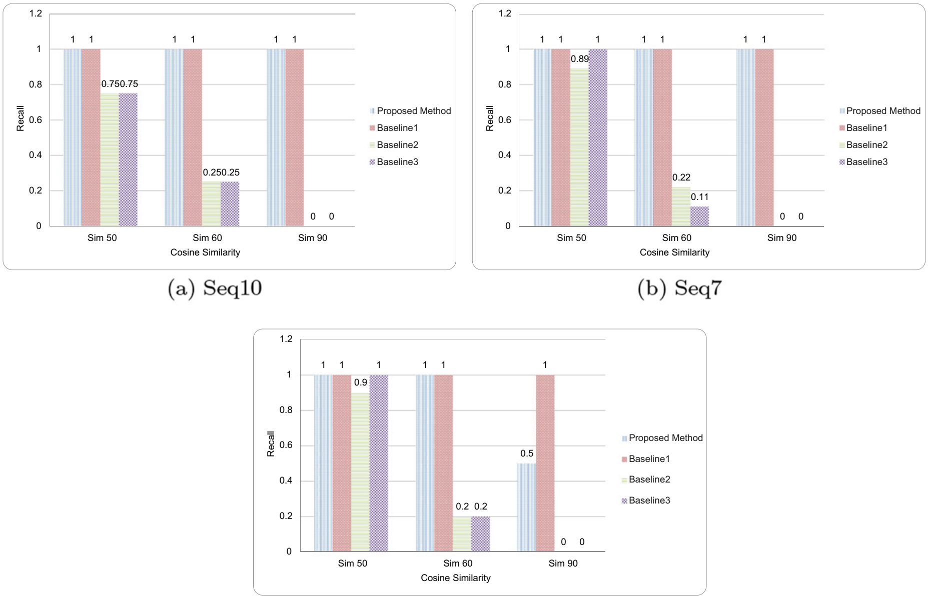
The experimental results against the baselines are presented in Figure 12. According to the results on the CC-News dataset, the proposed method finds matching rules for all test sequences with *Seq10* and *Seq7* under the cosine similarity threshold of 0.9. Having a different performance for this data set, *Baseline1* has the same performance as the proposed method with *Seq10* and *Seq7*. This is possibly due to the high overlap between the most frequent subgraphs and the rule antecedents.

On the other hand, recall value with *Seq5*'s is less than the *Baseline1* when we use the same cosine similarity threshold. It is an expected result since the pairwise similarity algorithm checks whether there exists a similar pair for the given sequential rules. Therefore as the sequence size increases, the possibility of finding a similar pair also increases.



(a) One-hot encoding (b) node2vec

**Figure 11** Recall vs. cosine similarity for Seq10 against the baselines



(a) Seq10

(b) Seq7

(c) Seq5

**Figure 12** Recall vs. cosine similarity results on the CC-News

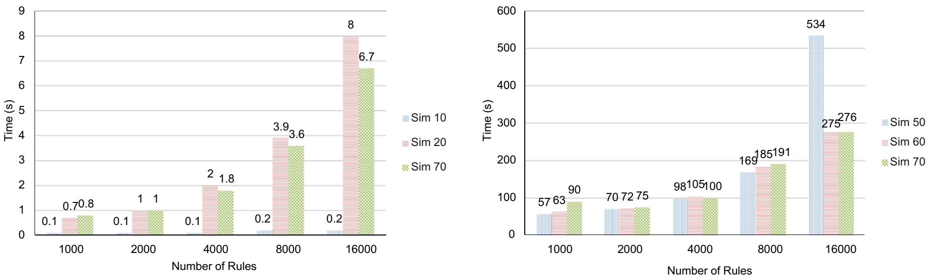
### 4.6 Experiment 4: time performance analysis of the proposed model

In this experiment, the effect of the number of extracted event patterns on the proposed method's time performance is analysed. For this purpose, we measure the prediction time with top-k interesting rules. We use 1000, 2000, 4000, 8000, and 16000 as the k value. Table 3 shows how many unique rule antecedents exist for each top-k setting which is essential since the time cost of rule matching depends on the number of unique rule antecedents to be searched. The k value for top-k interesting rules only indirectly affect the time cost as it is expected to have more unique rule antecedents as the number of rule increases.

Figures 13, 14, and 15 present the execution time of the proposed event prediction method for *Seq5*, *Seq7*, and *Seq10*, respectively. According to the figures, as the number of rules rises, the duration of the pairwise similarity and prediction processes increase. However, there is an exception for Seq7 where the time required for the prediction module did not change significantly, since the number of unique rule antecedents remained almost same

**Table 3** Number of unique rule antecedents within event pattern set for varying sequence lengths and event pattern set size

| Sequence length | Top-k rules | | | | |
|---|---|---|---|---|---|
| | **1000** | **2000** | **4000** | **8000** | **16000** |
| **5** | 11 | 20 | 38 | 74 | 145 |
| **7** | 9 | 11 | 11 | 11 | 11 |
| **10** | 1000 | 2000 | 4000 | 8000 | 16000 |



(a) One-hot encoding (b) node2vec

**Figure 13** Time vs. Number of Rules for Seq5

as the number of rules increased. Besides, it is seen that the similarity threshold affects the duration of the prediction process. This is due to that as the similarity threshold increases, the amount of search to find a matching subgraph increases.

In all the experiments, the method with node2vec embedding takes more time than one hot encoding model. This is an expected result since in our setup, node2vec may generate more than one embedding per node, and the pairwise similarity calculations per embedding vector increases the duration.

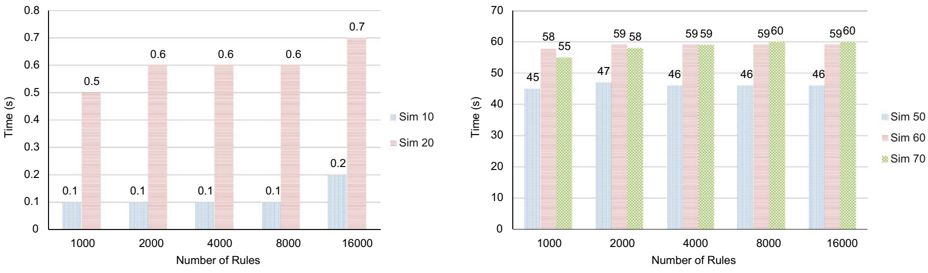
### 4.7 Case analysis

In this section, we analyze the event prediction performance of the proposed method on a case such that a set of news from from December 2016 in Turkish news dataset is matched against event patterns from January 2015 to September 2016 to be able to predict event skeleton for the rest of December 2016. The event patterns are constructed with sequence length *Seq7* and matched under node2vec embedding and *Sim60* cosine similarity threshold.

The subgraph sequence to be analyzed for the case study is given below:

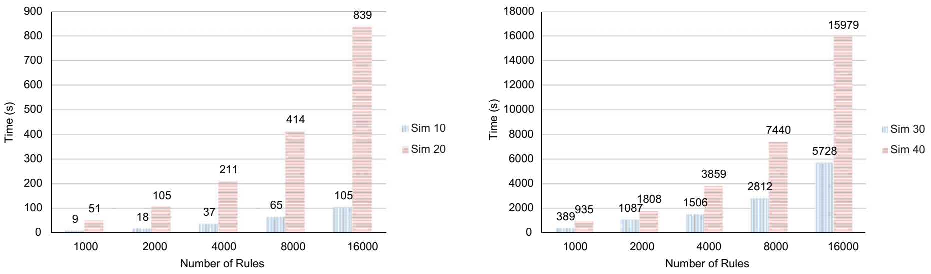
– {510,**514**,516,523}, {511,515,520,521,522}, {511,**513**,514,515,521,522}, {515,516,520,521,523,524}, {516,520,523,524}

Here, the subgraphs are shown with their identifiers. Among them, the first set matches with the following event pattern: {395} → {393}



(a) One-hot encoding (b) node2vec

**Figure 14** Time vs. Number of Rules for Seq7



(a) One-hot encoding (b) node2vec

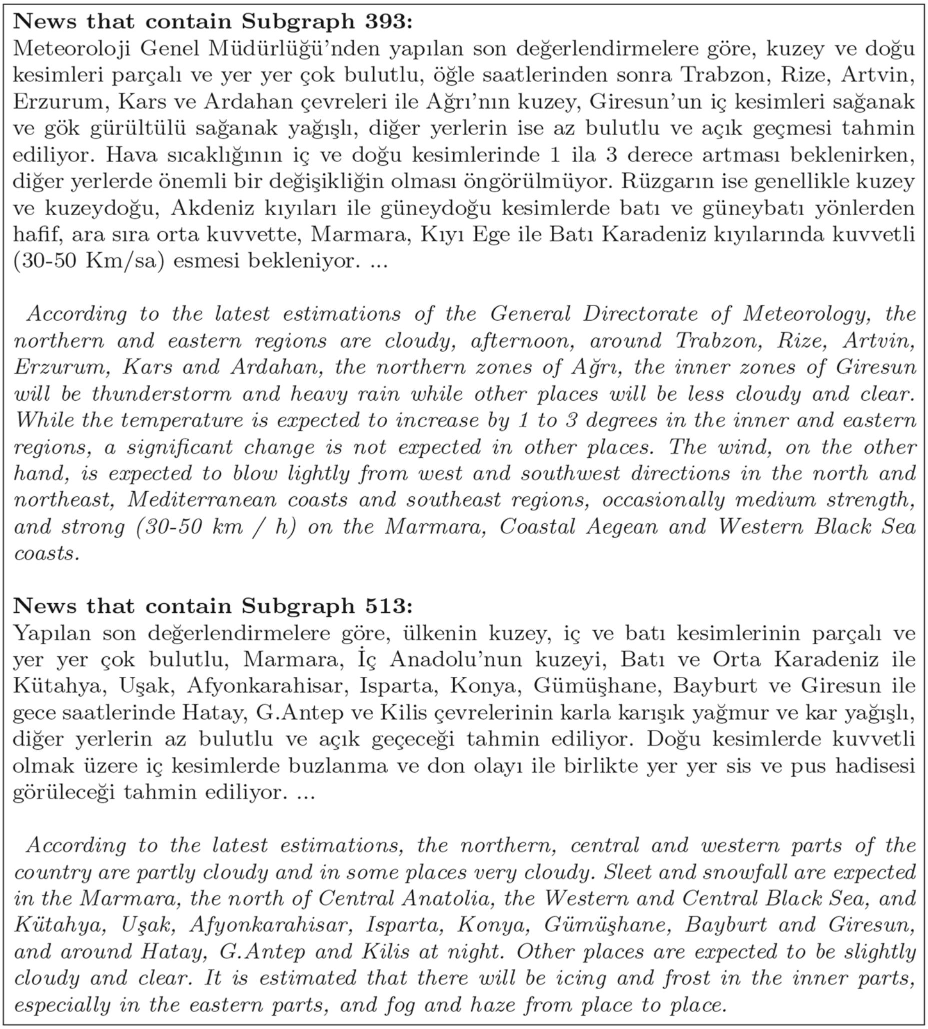
**Figure 15** Time vs. Number of Rules for Seq10

The model's prediction is an event skeleton that is considered as similar to Subgraph 393. The pairwise similarity algorithm determined that in the test set, Subgraph 513 is similar to Subgraph 393. Therefore, the model's prediction is considered as a successful prediction. According to the results, even though the similarity threshold was selected moderate, content of the predicted subgraph and the appeared subgraph can be considered as similar, since the contents of the news are related to the weather conditions of cities in Turkey and it is predicted that unfavorable weather conditions will arise (in subgraph 393), as also happens in the time window to be predicted (subgraph 513). We present first paragraph of the articles that contain these subgraphs in Figure 16. Additionally, subgraph visuals for the Subgraph 393 and Subgraph 513 are presented in Figure 17.

## 4.8 Scalability of the proposed method

The proposed method consists of several modules that work in sequence. In general terms, time cost of each one can be described as follows

- **News graph construction module.** In the first module, each of the news in the news text collection is represented as a graph. Given N news documents, this module calls for scanning each piece of text once to count the words and to calculate TF-IDF values for TF-IDF based filtering of terms. Hence, given N news documents, the time complexity of TF-IDF is O(NLlog(NL)), where L is the average length of each document.
- **Frequent subgraph extraction module.** In the second module, N news graphs are partitioned into k time windows, and freqent subgraph mining algorithm is applied for each partition. The gSpan [42] algorithm, which is used in this phase is efficient as it is based on pattern growth rather than candidate generation, and it uses canonical labeling to avoid duplicate subgraphs. We can consider that all graphs in each of the partitions are scanned to construct a simpler representation for further processing. Hence on the overall N graphs are scanned at this step, resulting with $M$ frequent subgraphs. Note that, at this step the number of subgraphs are further reduced through clustering.
- **Subgraph sequence construction module.** Sequence construction is based on sorting the subgraphs with respect to their timestamps. Assuming that each of the k partitions include L subgraphs, this phase approximately has complexity of $O(k * L * logL)$. The number of resulting sequences is based on the sequence length parameter.
- **Event pattern extraction module.** This module receives the set of subgraph sequences, $S$ as input and outputs sequence rules. In this phase, we use SPMF [8] implementation



**News that contain Subgraph 393:**

Meteoroloji Genel Müdürlüğü'nden yapılan son değerlendirmelere göre, kuzey ve doğu kesimleri parçalı ve yer yer çok bulutlu, öğle saatlerinden sonra Trabzon, Rize, Artvin, Erzurum, Kars ve Ardahan çevreleri ile Ağrı'nın kuzey, Giresun'un iç kesimleri sağanak ve gök gürültülü sağanak yağışlı, diğer yerlerin ise az bulutlu ve açık geçmesi tahmin ediliyor. Hava sıcaklığının iç ve doğu kesimlerinde 1 ila 3 derece artması beklenirken, diğer yerlerde önemli bir değişikliğin olması öngörülmüyor. Rüzgarın ise genellikle kuzey ve kuzeydoğu, Akdeniz kıyıları ile güneydoğu kesimlerde batı ve güneybatı yönlerden hafif, ara sıra orta kuvvette, Marmara, Kıyı Ege ile Batı Karadeniz kıyılarında kuvvetli (30-50 Km/sa) esmesi bekleniyor. ...

*According to the latest estimations of the General Directorate of Meteorology, the northern and eastern regions are cloudy, afternoon, around Trabzon, Rize, Artvin, Erzurum, Kars and Ardahan, the northern zones of Ağrı, the inner zones of Giresun will be thunderstorm and heavy rain while other places will be less cloudy and clear. While the temperature is expected to increase by 1 to 3 degrees in the inner and eastern regions, a significant change is not expected in other places. The wind, on the other hand, is expected to blow lightly from west and southwest directions in the north and northeast, Mediterranean coasts and southeast regions, occasionally medium strength, and strong (30-50 km / h) on the Marmara, Coastal Aegean and Western Black Sea coasts.*

**News that contain Subgraph 513:**

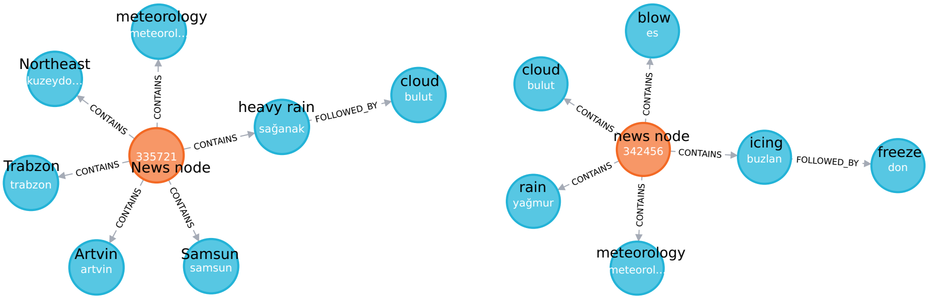
Yapılan son değerlendirmelere göre, ülkenin kuzey, iç ve batı kesimlerinin parçalı ve yer yer çok bulutlu, Marmara, İç Anadolu'nun kuzeyi, Batı ve Orta Karadeniz ile Kütahya, Uşak, Afyonkarahisar, Isparta, Konya, Gümüşhane, Bayburt ve Giresun ile gece saatlerinde Hatay, G.Antep ve Kilis çevrelerinin karla karışık yağmur ve kar yağışlı, diğer yerlerin az bulutlu ve açık geçeceği tahmin ediliyor. Doğu kesimlerde kuvvetli olmak üzere iç kesimlerde buzlanma ve don olayı ile birlikte yer yer sis ve pus hadisesi görüleceği tahmin ediliyor. ...

*According to the latest estimations, the northern, central and western parts of the country are partly cloudy and in some places very cloudy. Sleet and snowfall are expected in the Marmara, the north of Central Anatolia, the Western and Central Black Sea, and Kütahya, Uşak, Afyonkarahisar, Isparta, Konya, Gümüşhane, Bayburt and Giresun, and around Hatay, G.Antep and Kilis at night. Other places are expected to be slightly cloudy and clear. It is estimated that there will be icing and frost in the inner parts, especially in the eastern parts, and fog and haze from place to place.*

**Figure 16** News text of the matching subgraphs

of PrefixSpan [14] and RuleGen [44]. The PrefixSpan algorithm scans $S$ once to construct a tree representation of the sequence. Although processing on this tree structure has less cost than scanning the sequence collection, as the size of $S$ increases, the size of the tree and its processing cost increases [2]. Therefore, we need to limit the number of frequent subgraphs generated in the previous stage to obtain the rules in a reasonable time.

Note that event pattern generation through these modules is not an online process. Although each module scans the collection it receives at least once, at each step the collection is modified from text to graph, then to sequence and rule, and meanwhile filtered with respect to the given threshold values.



(a) Subgraph 393 (b) Subgraph 513

**Figure 17** Predicted Subgraph and Matched Test Subgraph

At the event prediction phase, which can be applied as an online module, we use an algorithm for the rule-subgraph matching. The prediction module's performance mainly depends on the number of rule antecedents as explored at the previous section.

## 5 Related work

In this section, firstly an overview of event prediction studies in the literature are given. Then, graph-based approaches applied to text modeling, graph association pattern mining and graph embedding techniques are summarized.

### 5.1 Event prediction

Event prediction is an extensively studied field in data mining. There exists various applications of event prediction across different domains. The works on this field can be categorized into temporal, spatial, semantic and joint prediction [47].

Temporal approaches focus on predicting the time of the event that will emerge. Studies under this category can further be classified into three subcategories. The traditional and one of the most studied sub problem is the prediction of whether an event will occur or not [15, 23]. In addition to the event occurrence prediction, the approximate date of the event, discrete-time event prediction approaches are also studied [39]. Moreover, to discover continuous-valued events, regression and point process methods are leveraged [30].

Spatial event prediction concentrates on the prediction of the location for upcoming events. Location can be represented as a raster or a point. For raster-based areas, spatial clustering, spatial convolution and spatial embedding techniques are utilized [16]. On the other hand, point-based spatial event prediction methods focus on whether a future event will occur at that specific region or not [48].

Semantic event prediction studies aim to explore semantic properties of future events such as topics and descriptions. In addition to the numeric attributes, studies in this category take into account natural languages and symbols. Association-based [50] and causality-based [21] techniques are applied to infer future events.

Finally, joint prediction studies try to discover jointly the time, location and semantics of possible events [43, 49], and [32]. Our work can be considered a time and semantics event detection system because we both utilize the co-occurrence relationship of news and consider the temporal association of news in graph sequences.

### 5.2 Graph representation

In [10], Genc et al. propose a graph-based model to explore events from text-based social media data. The graph consists of a chain of tokens where there is a link between consecutive tokens which is enriched by introducing timestamp in various granularities.

Erdemir et al. [6] discuss the query performance of hyper-graph based and graph-based models on graph database. According to their findings, the hyper-graph model outperforms the latter in the execution time of the queries involving multiple types of nodes. On the other hand, the storage cost for the hyper-graph model is increased significantly due to the need for the hyper nodes.

Rousseau et al. [35] introduce a new paradigm for the text categorization problem by considering it as a graph classification problem. In their graph model, nodes consist of unique terms that exist in the document, and the edges exist between terms that appear within a fixed-size sliding window.

Schenker et al. [37] propose a novel way to cluster web documents. Each web document is represented as a graph where each term is assigned to a unique node, and the graph includes three types of edges, title, link, and text, depending on where the term appears in the document.

As summarized above, there is a variety of ways to represent text in the form of a graph. Our graph model is similar to Genc et al.'s [10] work in the way of creating news graphs. In contrast to that study, we eliminate both stopwords and the words with low TF-IDF scores. Furthermore, the way news graphs are used are different in both studies.

### 5.3 Graph-pattern association rules

Since the graph patterns are beneficial to detect the associations between the parts of the graphs, association rules within graphs has recently attracted attention. One of the main differences among such studies on GPAR (Graph-pattern Association Rules) is the type of graph patterns to be extracted.

GPARs are introduced by Fan et al. [7] to explore associations within social graphs. They explore rules in the form of $Q(x, y) \rightarrow q(x, y)$ where $Q(x, y)$ is a graph pattern and $q(x, y)$ is an edge. In other words, the proposed model discovers patterns whose consequents contain only one edge. Namaki et al. [26] introduce Graph Temporal Association Rules (GTAR) to discover streaming graph patterns. In the proposed model, events are illustrated as graph patterns, and the association between the temporal events within a pre-defined time window is discovered. Wang et al. [41] propose a novel technique to extract generic GPARs. After finding the frequent subgraphs, they generate rules in the form of $Q_r \rightarrow Q_l$, where both $Q_r$ and $Q_l$ can be any graph patterns and share at least one common node but no edge in common.

Our work differs from these studies in several aspects. First of all, our model discovers sequential patterns within textual contents. As a result, our primary data source is text which make the task more challenging. Additionally, our model explores sequential patterns instead of association rules. Sequential patterns consist of an ordered list of items, so time-based order is taken into account in our solution

### 5.4 Graph embeddings

Graphs do not have a grid structure like an image, so applying machine learning and deep learning algorithms on a graph incurs additional challenges. In recent years due to the advancements in this field, many machine learning-based techniques are proposed to represent graphs in a lower-dimensional space.

The conventional way for information extraction from graphs is extracting hand-crafted features [18]. Although these methods scale well, they may output some pairs that do not seem similar. Graph kernels [40] measure the similarity between graph pairs using kernel function. The main disadvantage of applying these methods is that they are time-consuming.

Random walk approaches such as DeepWalk [29] and node2vec [11] are used for node embedding techniques. These methods' main limitation is that they do not share any parameters, so it is not possible to encode graphs that have not observed during training. The random walk methods aim to maximize the probability of visiting a vertex $v_j$ from an initial vertex $v_i$, $p(v_j|v_i)$ if $v_j$ is visited by a walk of length T. Since optimization process of embeddings for each vertex at each step is very costly, these algorithms use negative sampling and update only small percentage of the weights of negative examples (vertices) at each iteration. struc2vec [33] introduces the structural identity of nodes while generating node embeddings. In other words, the structural similarity of nodes and their vectorial representations are correlated.

In addition to the node embeddings, we can also represent the whole graph or subgraph in a single vector. Sub2vec [1] uses biased random walks to learn the latent representation of an arbitrary subgraph. It provides two properties, namely neighborhood and structural, to learn subgraph embeddings. While the neighborhood property captures the local connectivity of any graph, the structural property captures structural information such as the degree of nodes. Subgraph2vec [27] learns the latent representation of rooted subgraphs. The framework uses the Weisfeiler-Lehman isomorphism test [38] to capture the structural equivalence information. Graph2vec [28] represents the whole graph in a single vector. It is based on doc2vec [20] where each graph is considered as a document and rooted subgraphs as words. Using the skip-gram model, graphs are represented in a latent vector.

More recently, graph neural networks such as GCN [17], GraphSAGE [12] are proposed to aggregate local neighborhoods and encode structural information. At each layer, neighborhood information is aggregated that can be any function such as mean, max-pool, or LSTM [13]. Deep graph models are inherently inductive. Hence, they share hidden parameters, so they capture the encoding of a graph that they have not seen during training. Graph neural networks suffer as the model's depth increases because they could not sufficiently represent training data. Furthermore, deep models tend to flatten the node representations. To solve these issues, Zhang et al. [46] inspired by BERT [5], a transformer-based deep language model, and adapted the idea of transfer learning into graphs. Instead of word tokens, Graph-Bert samples nodes with their context. They called this structure as linkless subgraphs. Graph-Bert is pre-trained on node attribute reconstruction and graph structure recovery tasks and fine-tuned on node classification and graph clustering tasks. Scalability of the node embeddings is a challenging issue. Lerer et al. [22] propose a distributed model, Pytorch-BigGraph, for huge graphs to find node embeddings by partitioning nodes and edges and executing threads simultaneously.

In this study, we apply node2vec [11], Sub2vec [1] and graph2vec [28] to encode the graph patterns into latent vectors. This latent representation is used for similarity calculation of graphs in our event prediction method.

## 6 Conclusion

In this work, we propose an event prediction method by extracting and using event patterns in the form of graphs and sequence rules. The core of the method lies in the concept of *event skeleton* in the form of subgraph, and in *event pattern* to represent the associations between the events along the time line. Hence our method aims to extract these associations and predict the event skeleton for the future time window. The proposed method includes the steps of graph construction from news text, frequent subgraph extraction, sequence rule generation and prediction.

Experimental results have shown that the proposed approach is more effective for event prediction than the baseline methods. Moreover, node2vec embeddings generate similar embeddings for the semantically similar subgraphs. On the other hand, it takes more time to match a sequential rule with a test sequence using node2vec embedding. Another outcome of this study is that sequence length affects the model's success. To be more specific, the proposed method provides the highest prediction performance with subgraph sequences of length 10, *Seq10*, among all settings when we utilized all sequential rules. According to the conducted experiments, as we increase the number of unique rule antecedents, the duration of the process also increases linearly.

As a future work, one possible research direction is to apply deep graph embedding models since they are inherently inductive. Hence they have the potential to represent graphs that were not seen while training the model. Additionally, it is possible to consider alternative structures, such as cliques or quasi-cliques for subgraph extraction.

**Acknowledgements** This work (No:117E566) is funded by the Scientific and Technological Research Council of Turkey (TUBITAK).

## Declarations

**Conflict of Interests** The authors declare that they have no conflict of interest.

## References

1. Adhikari, B., Zhang, Y., Ramakrishnan, N., Prakash, B.A.: Sub2vec: feature learning for subgraphs. In: Pacific-Asia conference on knowledge discovery and data mining. Springer, pp 170–182 (2018)
2. Agrawal, R., Imieliński, T., Swami, A.: Mining association rules between sets of items in large databases. In: Proceedings of the 1993 ACM SIGMOD international conference on management of data, pp 207–216 (1993)
3. Atefeh, F., Khreich, W.: A survey of techniques for event detection in twitter. Comput Intell **31**(1), 132–164 (2015)
4. Cheng, H., Yan, X., Han, J.: Mining graph patterns. In: Frequent pattern mining. Springer, pp 307–338 (2014)
5. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. arXiv: abs/1810.04805 (2018)
6. Erdemir, M., Goz, F., Mutlu, A., Karagoz, P.: Comparison of querying performance of neo4j on graph and hyper-graph data model. In: KDIR, pp 397–404 (2019)
7. Fan, W., Wang, X., Wu, Y., Xu, J.: Association rules with graph patterns. Proc VLDB Endowment **8**(12), 1502–1513 (2015)
8. Fournier-Viger, P., Lin, J.CW., Gomariz, A., Gueniche, T., Soltani, A., Deng, Z., Lam, H.T.: The spmf open-source data mining library version 2. In: Joint European conference on machine learning and knowledge discovery in databases. Springer, pp 36–40 (2016)
9. Fournier-Viger, P., Lin, J.CW., Kiran, R.U., Koh, Y.S., Thomas, R.: A survey of sequential pattern mining. Data Sci Pattern Recogn **1**(1), 54–77 (2017)

10. Genc, H., Yilmaz, B.: Text-based event detection: deciphering date information using graph embeddings. In: International conference on big data analytics and knowledge discovery. Springer, pp 266–278 (2019)
11. 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)
12. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: Advances in neural information processing systems, pp 1024–1034 (2017)
13. Hamilton, W.L., Ying, R., Leskovec, J.: Representation learning on graphs: Methods and applications. arXiv:1709.05584 (2017)
14. Han, J., Pei, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., Hsu, M.: Prefixspan: mining sequential patterns efficiently by prefix-projected pattern growth. In: proceedings of the 17th international conference on data engineering. Citeseer, pp 215–224 (2001)
15. Inceoglu, F., Jeppesen, J.H., Kongstad, P., Marcano, N.JH., Jacobsen, R.H., Karoff, C.: Using machine learning methods to forecast if solar flares will be associated with cmes and seps. Astrophys J **861**(2), 128 (2018)
16. Jiang, Z.: A survey on spatial prediction methods. IEEE Trans Knowl Data Eng **31**(9), 1645–1664 (2018)
17. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv:1609.02907 (2016)
18. Koutra, D., Parikh, A., Ramdas, A., Xiang, J.: Algorithms for graph similarity and subgraph matching. In: Proc. Ecol. inference conf, vol 17 (2011)
19. Kumaran, G., Allan, J.: Text classification and named entities for new event detection. In: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval, pp 297–304 (2004)
20. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: International conference on machine learning, pp 1188–1196 (2014)
21. Lei, L., Ren, X., Franciscus, N., Wang, J., Stantic, B.: Event prediction based on causality reasoning. In: Asian Conference on intelligent information and database systems. Springer, pp 165–176 (2019)
22. Lerer, A., Wu, L., Shen, J., Lacroix, T., Wehrstedt, L., Bose, A., Peysakhovich, A.: Pytorch-biggraph: a large-scale graph embedding system. arXiv:1903.12287 (2019)
23. Lin, Y.L., Yen, M.F., Yu, L.C.: Grid-based crime prediction using geographical features. ISPRS Int J Geo-Inform **7**(8), 298 (2018)
24. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems, pp 3111–3119 (2013)
25. Nagel, S.: Cc-news. http://web.archive.org/save/http://commoncrawl.org/2016/10/newsdatasetavailable (2016)
26. Namaki, M.H., Wu, Y., Song, Q., Lin, P., Ge, T.: Discovering graph temporal association rules. In: Proceedings of the 2017 ACM on conference on information and knowledge management, pp 1697–1706 (2017)
27. Narayanan, A., Chandramohan, M., Chen, L., Liu, Y., Saminathan, S.: subgraph2vec: learning distributed representations of rooted sub-graphs from large graphs. arXiv:1606.08928 (2016)
28. Narayanan, A., Chandramohan, M., Venkatesan, R., Chen, L., Liu, Y., Jaiswal, S.: graph2vec: learning distributed representations of graphs. arXiv:1707.05005 (2017)
29. 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)
30. Qiao, Z., Zhao, S., Xiao, C., Li, X., Qin, Y., Wang, F.: Pairwise-ranking based collaborative recurrent neural networks for clinical event prediction. In: Proceedings of the twenty-seventh international joint conference on artificial intelligence (2018)
31. Rajaraman, A., Ullman, J.D.: Mining of massive datasets. Cambridge University Press (2011)
32. Ramakrishnan, N., Butler, P., Muthiah, S., Self, N., Khandpur, R., Saraf, P., Wang, W., Cadena, J., Vullikanti, A., Korkmaz, G., et al: 'beating the news' with embers: forecasting civil unrest using open source indicators. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, pp 1799–1808 (2014)
33. Ribeiro, L.F., Saverese, P.H., Figueiredo, D.R.: struc2vec: learning node representations from structural identity. In: Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining, pp 385–394 (2017)
34. Ritter, A., Etzioni, O., Clark, S.: Open domain event extraction from twitter. In: Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, pp 1104–1112 (2012)

35. Rousseau, F., Kiagias, E., Vazirgiannis, M.: Text categorization as a graph classification problem. In: Proceedings of the 53rd Annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: long papers), pp 1702–1712 (2015)
36. Sanfeliu, A., Fu, K.S.: A distance measure between attributed relational graphs for pattern recognition. IEEE Trans Syst Man Cybern **SMC-13**(3), 353–362 (1983)
37. Schenker, A., Last, M., Bunke, H., Kandel, A.: Clustering of web documents using a graph model. In: Web document analysis: challenges and opportunities. World Scientific, pp 3–18 (2003)
38. Shervashidze, N., Schweitzer, P., Van Leeuwen, E.J., Mehlhorn, K., Borgwardt, K.M.: Weisfeiler-lehman graph kernels. J Mach Learn Res **12**(77), 2539–2561 (2011)
39. Tops, H., van den Bosch, A., Kunneman, F.: Predicting time-to-event from Twitter messages (2013)
40. Vishwanathan, S.VN., Schraudolph, N.N., Kondor, R., Borgwardt, K.M.: Graph kernels. J Mach Learn Res **11**, 1201–1242 (2010)
41. Wang, X., Xu, Y., Zhan, H.: Extending association rules with graph patterns. Expert Syst Appl **112897**, 141 (2020)
42. Yan, X., Han, J.: gspan: graph-based substructure pattern mining. In: 2002 IEEE International conference on data Mining, 2002. Proceedings. IEEE, pp 721–724 (2002)
43. Yang, Q., Wang, H., Zhang, W.: Web-log mining for quantitative temporal-event prediction. IEEE Comput Intell Bull **1**(1), 10–18 (2002)
44. Zaki, M.J.: Spade: an efficient algorithm for mining frequent sequences. Mach Learn **42**(1-2), 31–60 (2001)
45. Zeng, Z., Tung, A.K., Wang, J., Feng, J., Zhou, L.: Comparing stars: on approximating graph edit distance. Proc VLDB Endowment **2**(1), 25–36 (2009)
46. Zhang, J., Zhang, H., Sun, L., Xia, C.: Graph-bert: only attention is needed for learning graph representations. arXiv:2001.05140 (2020)
47. Zhao, L.: Event prediction in the big data era: a systematic survey. ACM Comput Surv (CSUR) **54**(5), 1–37 (2021)
48. Zhao, L., Wang, J., Chen, F., Lu, C.T., Ramakrishnan, N.: Spatial event forecasting in social media with geographically hierarchical regularization. Proc IEEE **105**(10), 1953–1970 (2017)
49. Zhao, X., Tang, J.: Modeling temporal-spatial correlations for crime prediction. In: Proceedings of the 2017 ACM on conference on information and knowledge management, pp 497–506 (2017)
50. Zhou, C., Cule, B., Goethals, B.: A pattern based predictor for event streams. Expert Syst Appl **42**(23), 9294–9306 (2015)

**Publisher's note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.