The final authenticated version is available online at https://doi.org/10.1007/978-3-031-46846-9_19

Social Network Mining from Natural Language Text and Event Logs for Compliance Deviation Detection

Henryk Mustroph[1], Karolin Winter[2], and Stefanie Rinderle-Ma[1]

[1] Technical University of Munich, TUM School of Computation, Information and Technology, Garching, Germany
{henryk.mustroph,stefanie.rinderle-ma}@tum.de
[2] Eindhoven University of Technology, Department of Industrial Engineering and Innovation Sciences, Eindhoven, The Netherlands
k.m.winter@tue.nl

Abstract. Social network mining aims at discovering and visualizing information exchange of resources and relations of resources among each other. For this, most existing approaches consider event logs as input data and therefore only depict how work was performed (as-is) and neglect information on how work should be performed (to-be), i.e., whether or not the actual execution is in compliance with the execution specified by the company or law. To bridge this gap, the presented approach considers event logs and natural language texts as input outlining rules on how resources are supposed to work together and which information may be exchanged between them. For pre-processing the natural language texts the large language model GPT-4 is utilized and its output is fed into a customized organizational mining component which delivers the to-be organizational perspective. In addition, we integrate well-known process discovery techniques from event logs to gather the as-is perspective. A comparison in the form of a graphical representation of both, the to-be and as-is perspectives, enables users to detect deviating behavior. The approach is evaluated based on a set of well-established process descriptions as well as synthetic and real-world event logs.

Keywords: Social Network Mining · Organizational Mining · Natural Language Processing · Compliance Checking.

1 Introduction

"The discovery of organizational knowledge, such as organizational structures and social networks, enables managers to understand organizational structures and improve business processes" [24]. Hence, *organizational mining*, i.e., the discovery and analysis of the organizational perspective of a process, has been investigated by various approaches [3,4,9,11,13,14,17,19,23,25,26,27]. Especially, in cooperative environments, the understanding of the communication and interaction between resources is of utmost importance [22].

Organizational mining has mostly taken event logs as input for detecting resources and their relations, resulting in knowledge of how the actual process execution was carried out (as-is behavior). Textual information on organizational structures such as handbooks or regulatory documents has been neglected as a valuable source of information. Such documents typically prescribe the *to-be* behavior, for example, which resources are authorized to perform which tasks or which resources are supposed to work together. Taking into account both, regulatory documents and process event logs, it can be analyzed if the observed resource behavior complies with or deviates from the prescribed behavior. The results can be used for detecting compliance violations, improving the quality of organizational mining, and enhancing the process model. Quality of models and model enhancement can be realized as knowledge from two data sources is combined. The importance of checking compliance of process executions with organizational models has been confirmed in literature [26]. In previous work we presented approaches for compliance verification between textual sources and event logs [5,16]: temporal and resource compliance patterns such as *activity A must be performed by resource R* stated in natural language text are checked against event logs. In this work, we follow up on these ideas and elaborate on a novel approach to mine resource interaction and communication from textual information and event logs in parallel enabling a compliance deviation detection.

To this end, literature on organizational mining from event logs is gathered and categorized in order to identify directions for organizational mining from text. Afterwards, this is contrasted with approaches that extract organizational aspects from textual sources. Unlike existing approaches for knowledge graph extraction from natural language text, this work outputs a social network that captures the communication and information exchange among resources in an organization, focusing on the to-be scenario of resource interactions. Thereby, compared to existing approaches that are mostly rule-based, this approach incorporates a Large Language Model (LLM) for pre-processing. The output of the LLM is transformed into several knowledge graphs representing the social network which serves as ground truth data for compliance deviation detection. Afterwards, for each trace in the event log a graph is generated that is checked for deviations against the graphs of the text. For this, existing social network mining approaches are taken as the starting point for developing a customized solution that allows for comparing the resulting graph (as-is model) with the graph generated based on the textual input (to-be model). In the last step, compliance deviation detection is enabled by comparing the to-be and as-is graphs. This provides an insightful output on which trace is acceptable and which one contains deviations. The approach is outlined in Sect. 3 and evaluated in Sect. 4. The paper concludes in Sect. 5.

2 Literature Review and Scope of this Work

A literature review is conducted to determine if the objectives of existing techniques for organizational mining from event logs remain valid and feasible when

considering natural language texts as input (cf. Sect. 2.1). We identify social network mining as the most promising research direction as it requires the least additional knowledge and assumptions. Secondly, we explore related work explicitly focusing on extracting social networks and resource interactions from natural language texts (cf. Sect. 2.2). For the literature search various libraries including dblp, ACM digital library, Scopus, IEEExplore, and SpringerLink, targeting documents with titles containing keywords such as "(Organizational Mining)", "(Social Network) AND (Business Process)", "(Organizational Structure) AND (NLP OR Natural Language Processing)", and "(Knowledge Graph) AND (NLP OR Natural Language Processing)" were used as well as forward and backward searches to ensure comprehensive coverage of the relevant literature[3]. The presented search strings were deliberately chosen to find a list as complete as possible of all existing papers in the social network and organizational structure mining domains related to business process management, while still remaining within a manageable range of total results.

2.1 Organizational Mining from Event Logs

Based on [27] and most recent work [26], the following categories of organizational mining are identified and investigated regarding the feasibility of discovery when utilizing text data as input, instead of relying on event log data.

Social Network Mining. Social network mining (SNM) determines the information exchange and relationships among various resources. In our context, a resource can be both human or non-human and the resource structure contains users having roles that are part of organizational units, e.g., departments being defined in an organization. In [1] first the control-flow of a process is identified, then the organizational structure is expressed and lastly, a model based on the transfer of tasks through different resources being a social network of resource relationships is generated. The model can be used to further analyze the roles of resources and the identification of resource groups [1]. [9] also uses SNM for business processes to discover who is working with whom. The authors in [13,17] employ SNM by deriving an activity matrix that indicates which resources perform specific activities. Based on the handover of work, an organizational network is constructed. SNM can also be utilized for conformance and compliance checking when trying to detect anomalies in processes as demonstrated in [11]. Additionally, [21] employ SNM to generate knowledge and identify problematic communication behaviors in agile development processes. The feasibility of conducting SNM using textual data appears promising, primarily due to the availability of control-flow relations and information about the performers of specific activities, commonly found in process descriptions. By utilizing this information, it is possible to construct a directed graph in which the nodes represent resources and the edges represent the activities and information exchange between resources, thereby capturing the interaction flow. This ground truth data can then be used to evaluate the conformity of communication structures in event logs.

[3] The detailed results are available at https://www.cs.cit.tum.de/bpm/data/.

Organizational Structure Discovery. This category is also called *Organizational Model Mining*. The aim is to create, e.g., an organizational hierarchical model similar to an "organigram" or to determine groups of resources by, e.g., identifying different organizational clusters of resources as described in detail in [24]. In [23,25], event log data is analyzed to construct a graphical tree format-like representation of the organizational structure. The graphical representation created from organizational structure discovery differs from the graph structure employed in SNM. In this case, the edges in the graph do not possess weights or directions, and the focus is not on modelling a network. Instead, a graph in tree format is constructed, comprising multiple tree structures associated with different activities, indicating the resources capable of performing each task and the authorities or resources' ability to perform various tasks. In the best case this results in an "organigram". Furthermore, resources can be grouped into different clusters based on similarities like the execution of the same activities or the contribution of the same amount of work to the current process from a temporal perspective. This can be combined with graphical representations and modelling approaches as described in [26]. Performing organizational structure discovery solely from text data is difficult and requires multiple additional data sources or assumptions. On the one hand, the communication structure and the flow of activities are crucial. The former could be captured through SNM. On the other hand, obtaining either the company's organizational structure would include expert knowledge of the general organizational structure within a particular domain or characteristics of graphs from the social network must be elaborated to establish an "organigram" from a social network. These requirements must be thoroughly examined to successfully build the desired model.

Resource Allocation and Mining. This category combines multiple categories found in the literature. *Resource Allocation* explores the resource activity pairs in an event log and seeks to enhance process execution efficiency by reallocating these pairs. *Role Mining* identifies which roles are allocated to which activities and aims to distribute activities among different roles. *Rule Mining* is described as allocating rules for efficient resource-task distribution and statistics for building expert teams which is similar to *Resource Allocation* and *Role Mining*. Lastly, *Behavioral Profile Mining* extracts individual resource behavior from event log data which provides statistics of resource activities and is therefore also similar to *Resource Allocation* and *Role Mining*. Those categories were merged into one category because of their similar types of output and purpose. In particular, all of them have the purpose of extracting statistical information based on execution data stored in the event logs in order to provide insights into resource-activity allocation to improve resource-task distributions and process efficiency. In [19], the presented approach utilizes event log data to perform several statistical evaluations of resource activity executions, role dependencies, and resource allocations. The data can be used to gain insights and improve process efficiency in future executions by appropriately distributing resource activities. In [14] decision trees are built based on the resource information of event logs helping in allocating the best suitable resource to several tasks. This type

of organizational mining is very difficult to apply to textual data. All resource allocation and rule mining tasks require a lot of execution data and create either statistics of resource executions such as the number of distinct activities done by a resource or train machine learning models such as decision trees with attributes of resources, i.e., the category or activity that has been executed or the roles, to classify the most suitable resource for an activity in future process executions. Since textual data only provides knowledge on the optimal resource allocation and the *to-be* model but not the *as-is* model, we lack information for creating statistics or machine learning models for optimal resource allocation.

Conclusion. From the lines of research discussed above, SNM requires the least assumptions and additional knowledge as input, i.e., a process description. Hence, we will employ SNM in the following. In addition to generating a social network graph (SNG) from textual input, we also mine event logs enabling the comparison between the *to-be* and *as-is* models.

2.2 Social Network Mining from Natural Language Text

In [3], email text data is utilized as input to generate an interaction graph as well as a hierarchy structure graph. In the former, each node represents a sender or receiver of an email and the nodes are connected with each other through directed, unweighted edges. The latter, the hierarchy structure graph, consists of a root node that is connected to intermediate nodes, and these intermediate nodes are further connected to leaf nodes. The edges in this graph are directed and point towards the bottom of the nodes, indicating the hierarchical relationship between senders and receivers of emails. These knowledge graphs fall within the scope of organizational mining, specifically in the subcategories of SNM and organizational structure mining. They provide insights into the communication patterns and hierarchical relationships within email traffic in an organization, thereby enhancing transparency in organizational processes. [3] focuses on communication through emails capturing the *as-is* scenario. In this paper, the purpose is to create communication networks representing the *to-be* scenario to check the compliance of information exchange in a process captured in the event logs.

In general, knowledge graphs create a network of entities, which are not limited to human resources as typically employed in SNM within the field of organizational mining. Previous papers have explored the utilization of NLP techniques to extract entities from unstructured text documents, e.g., [10,20]. An approach focusing on organizational aspects is [6]. NLP techniques were employed to extract organizational structure entities, i.e., the rank or organization of a resource from text data obtained from the Security Force Monitor, an institution that collects information on security forces worldwide. The authors utilized entity recognition whereas tokens within the text were classified into organizational structure categories. After that, the entities in a text were grouped into a graph based on similarity measures and relation extraction techniques embedding the resource into the full organizational structure and defining for each resource the rank, role, department, i.e., *person x* with *rank y* in *organization z*. This approach enabled meaningful insights into the composition and

hierarchy of security forces globally. However, no social network displaying the relationships among the resources was constructed.

Approaches discovering process models from natural language text, e.g., [12] target the organizational structure implicitly, e.g., through visualizing actors through lanes in BPMN. In [7], a Large Language Model is used to extract activities and actors from natural language text. In contrast, our work aims at explicitly visualizing the social network and comparing it to the observed social network from an event log to reveal deviations from the *to-be* behavior.

This paper differs from our most recent work on compliance verification between process descriptions and event logs [16], as the focus is not on compliance requirements such as *activity A must be performed by resource R*, but rather on how resources interact with each other and whether this interaction violates any compliance rules or constraints and conforms with the *to-be* behavior.

Conclusion. Unlike existing organizational aspects extraction approaches from natural language text, this paper aims to capture the communication and information exchange among resources in an organization, focusing on the *to-be* scenario of resource interactions. The resulting social network will serve as ground truth data for compliance checking with the *as-is* model from event logs.

3 Social Network Graph Construction and Compliance Deviation Detection

The approach, cf. Fig. 1, requires explicit control-flow relations descriptions in the natural language text since otherwise interactions among resources are not traceable. Four types of control-flow relations are distinguished: sequential, conditional, parallel, and loop. Sequential flow is considered during social network graph (SNG) construction from text and event logs. The conditional case requires handling only for texts because if an activity, executed within a condition, terminates the process, multiple different resource interaction networks exist. The parallel case only requires particular handling for event logs. Loops are not considered explicitly for SNG construction as it is expected that they only multiply existing resource interactions without adding new resource interactions. Section 3.1 details on SNG construction from text and we denote the set of resulting SNGs as *SNG-T*. Analogously, the set of SNGs generated from event logs is denoted as *SNG-EL* and its construction is outlined in Sect. 3.2. Section 3.3 describes how to detect compliance deviations and how to visualize the results.

3.1 Social Network Graph Construction from Text

This step of the approach is divided into a pre-processing part using a LLM to transform the unstructured text data into a structured format, and the *SNG-T* set construction through Alg. 1. Depending on whether we observe a decision or not in the text, the set *SNG-T* either contains one or multiple graphs which are considered as ground truth for detecting compliance deviations in Sect. 3.3.

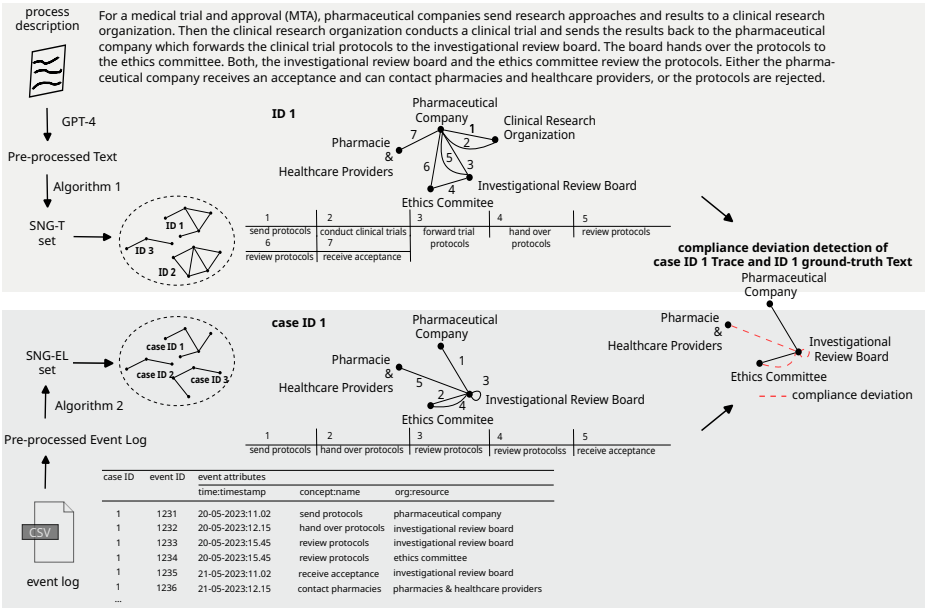

Fig. 1. Overview of SNG Construction and Compliance Deviation Detection Approach

Text Pre-processing. For pre-processing, OpenAI's GPT-4 model [18] is used. The output generated by GPT-4 is stored in a persistent file, allowing users to review the results generated by GTP-4 and to make modifications if necessary. A prompt, available at the corresponding repository at https://www.cs.cit.tum.de/bpm/software/, was designed to extract the source and sink activity, the resource performer and the resource consumer. While designing this prompt we recognized that GPT-4 delivers better results when using the term actor instead of resource. The source activity always serves as the starting point of the interaction and connects the resource performer with the resource consumer. The sink activity indicates the next activity that establishes an interaction between resources. In Fig. 1 an example of a source activity is *"send research approaches and results"*. It connects the resource performer *"pharmaceutical company"* with the resource consumer *"clinical research organization"*. The corresponding sink activity is *"conduct clinical trial"* which also acts as a connection to the following resource pairs. Moreover, information indicating whether the sink activity terminates the process, and the type of control-flow, i.e., whether the source and sink activities are part of a condition or not is extracted. It is important to detect control-flow relations as this will determine whether the *SNG-T* set consists of one or multiple graphs. The control-flow, represented as a sequence of activities, is tracked using an ascending ID. If a description solely contains sequential, loop or parallel control-flow relations and no conditional case, *SNG-T* consists of only one graph. Sequential cases are just an order of follow-up activities which is easy

to track. Loop relations do not create new interactions between resources, because the same interactions are executed again and again. In the event log, this would result in counting activities multiple times. The parallel case, in the form of an "AND"-gateway does not require particular treatment because a flow of parallel activities does not result in several graphs compared to the conditional case where multiple branches can occur if one activity in the branch leads to the termination of the whole process, i.e., to termination of resource interactions. In addition, the parallel case will be tracked by GPT-4 as a normal sequential case for each parallel path. That means GPT-4 realizes automatically that parallel cases in the description exist and captures all required information of each parallel path after each other. In Fig. 1 an example of a parallel case is: *"investigational review board"* and *"ethics committee"* which are both resource consumers executing activity *"review protocols"*; after both have finished, they send their results to the *"pharmaceutical company"* which acts as a resource consumer. This case is captured by GPT-4 sequentially, i.e., added to the pre-processed output file after each other. Since both have the same consumer, the interaction of resources can still be built correctly and no special check is necessary for parallel executions of activities by different resources. The conditional case requires additional measures to be resolved, since activities in a conditional branch can terminate the process, i.e., a completely new and different resource interaction model will exist. This situation is illustrated in the running example in Fig. 1. The last two activities *"receive an acceptance and contact"*, and *"reject protocols"* are connected through an "XOR"-gateway implying that the resource consumers *"pharmacies and healthcare providers"* are only part of the SNG if *"receive an acceptance and contact"* is executed. In the case of *"reject protocols"*, a different, but also valid communication graph exists. An example pre-process output from text looks as follows: *{"actors": [{"actor_id": 1, "actor_name": "Pharmaceutical Company"}, ...], "activities": [{"activity_id": 1, "activity_name": "Send Research Approaches and Results"}, ...], "control_flow": [{"control_flow_id": 1, "activity_from": 1, "activity_to": 2, "actor_performer": 1, "actor_receiver": 2, "terminating_activity": false, "flow_type": "sequential"}, ...]}*

SNG-T Construction. Algorithm 1 outlines the *SNG-T* construction for which only the sequential control-flow and the conditional control-flow are relevant. The output from the pre-processing component is considered as input and the result consists of one or multiple *SNG-T* graphs. The edges of a graph correspond to activities and the nodes to resources. The algorithm starts by creating an empty list for *SNG-T* graphs and a counter that is necessary to check if the process terminated after an activity in a conditional statement after the last activity of the process (**Step 1**). Subsequently, the list of resources, activities and the control-flow list is derived from the pre-processed text file (**Step 2**). Furthermore, an empty list of edges is created storing information such as the resource performer, resource consumer, and source activity, which serves as the label for the edge connecting the performer and consumer (**Step 3**). A list of nodes is stored with all unique resources (**Step 4**). The main part of the algorithm starts to determine the edges and the different *SNG-T* graphs. The control-flow list

is iterated through a for-loop, till all values are checked. Information, such as the source activity, resource performer, and the resource consumer is cached in a local variable respectively (**Step 5**) and added to the list of edges, only if the resource performer and the resource consumer are no null values (**Step 6**). Suppose a sink activity is identified as a terminating activity and the counter value minus one is equal to the length of the control-flow list. In that case, the edge list is closed, and the node list is checked for any uninvolved resources, which are subsequently removed from the resource list (**Step 7**). Uninvolved in this case means a resource added to the node list in the beginning does not appear in the edge list since the terminating activity stops adding edges and therefore resource interaction. The resulting lists of nodes and edges are added to the list of *SNG-T* elements (**Step 8**) since a new unique element was found that can be used to check the compliance deviations of the event log captured resource interactions. The edge representing the terminating activity is removed from the list of edges, to further process with the not terminating activities and the list of nodes is again filled with all resources in the subsequent iteration(**Step 9 & 10**). Suppose the sink activity is not a terminating activity and the control counter value minus one is equal to the length of the control-flow list. In that case, all activities have been gone through and the node list is also checked for resources that are not in the edge list and removed if found (**Step 11**). The lists of nodes and edges are added to the graph list as a new graph object (**Step 12**).

Algorithm 1 *SNG-T* Construction from LLM-based Pre-Processed Text

Require: Pre-processed text file (based on specified prompt)
Ensure: Create a *list of SNG-T elements*
  Step 1: Create an empty *list of SNG-T elements* and *counter* ← 0
  Step 2: Get the list of *resources*, *activities* and *control_flow* from the pre-processed text file
  Step 3: Create a new empty list *edges*
  Step 4: Fill the empty list of *nodes* with all distinct resources
  for entry **in** *control_flow* **do**
    Step 5: Get from entry
    activity_from, *resource_performer*, *resource_consumer*,
    terminating_activity
    if *resource_performer* and *resource_consumer* != null **then**
      Step 6: Add to *edges*:
      (resource_performer, resource_consumer, *from_activity*)
    end if
    if *terminating_activity* == True **and**
      counter != *length(control_flow_list)* - 1 **then**
      Step 7: Remove all resources in *nodes* that are not on *edges*
      Step 8: Add tuple: *(nodes, edges)* to *list of SNG-T elements*
      Step 9: Remove from *edges* terminating activity edge
      Step 10: Fill the list of nodes with all distinct *resources*
    end if
    if *terminating_activity* == True **and**
      counter == *length(control_flow_list)* - 1 **then**
      Step 11: Remove all resources in *nodes* that are not in *edges*
      Step 12: Add tuple: *(nodes, edges)* to *list of SNG-T elements*
    end if
    counter = *counter* + 1
  end for

3.2 Social Network Graph Construction from Event Log

Analogously to the previous step, this one is again divided into pre-processing the event log and *SNG-EL* construction based on the pre-processed event log. The resulting *SNG-EL* serves, together with *SNG-T*, as input for compliance deviation detection in Sect. 3.3. First, we thoroughly assessed implementations provided by the PM4Py toolkit[4] for suitability in our setting. It provides social network mining methods such as *handover of work* and *working together*. Those calculate statistics over all the traces or activities which appear in the log and return a mean resource interaction. Since we want to detect compliance deviations, all distinct traces need to be searched for deviations instead of using average or accumulated statistics of all traces. In addition, restrictive assumptions are imposed, like either the follow-up activity performer is always the previous consumer or event attributes are required indicating the mapping between input and output activities. Therefore, we take up the challenge of generating *SNG-EL* graphs without those strict assumptions on the event log.

Event Log Pre-processing. The event log data is parsed, and transformed into an event log object and all unique traces within the event log are identified. A distinct trace refers to a unique sequence of resource-activity pairs present in the log. If a trace occurs multiple times, only the corresponding case ID is recorded in a list associated with a similar unique trace. Like before, the event log's pre-processed file is persistently stored and contains information on resource performers, activities, and resource consumers. However, the pre-processed file for the log has a slightly different structure. It stores a flow of activities for each distinct trace, along with additional trace structures such as a unique ID, and all case IDs for each trace that share the same order of resource-activity pairs and are thus classified as similar traces to the stored one. Furthermore, unlike the text's pre-processed file, the event log's pre-processed file does not store multiple different lists for resources, activities, and control flow. Although this information is generally redundant, it has been observed that GPT-4 produces better results when there is a distinction between these components, while the control-flow list combines the IDs of activities and resources. The file generated after event log pre-processing offers several advantages, e.g., it can be easily modified, enhanced, and expanded, allowing for corrections to be made before conducting compliance deviation detection. An example pre-process output from an event log looks as follows: *{"pairs": [{"id": 1, "case_ids": ["1", ...], "network_trace": [{"Resource Performer": "Pharmaceutical Company", "Resource Consumer": "Investigational Review Board", "Activity": "send protocols"}, ...], ...}, ...]}*

SNG-EL Construction. The pre-processed file contains the necessary information for each distinct trace and is utilized to construct *SNG-EL* graphs that have the same structure as those in the *SNG-T* set. This ensures comparability during compliance deviation detection. Algorithm 2 details how to come up with several elements that detect sequential and parallel cases in each trace.

[4] https://pm4py.fit.fraunhofer.de, last access: 2023-07-06

Algorithm 2 *SNG-EL* Construction from Pre-Processed Event Log

Require: Pre-processed Event Log (own defined output structure)
Ensure: Create a *list of SNG-EL elements*
Step 1: Create an empty *list of SNG-EL elements* and *edges*
Step 2: Get the *trace_info_list* containing data of each distinct trace in the pre-processed event log file
Step 3: Create a Petri net of the event log, using a process mining algorithm
Step 4: Create a list *and_gateway_activities* that stores from the Petri net the *start_activity* before the parallel gateway, the *first_activities_after_gateway*, all *activities_in_each_path*, and the *end_activity_after_gateway*
for entry **in** *trace_info_list* **do**
  Step 5: Fill the empty list of *nodes* with all distinct resources of entry
  for (*resource performer*, *resource consumer*, *activity*) **in** entry **do**
    if *activity* is a *start_activity* in *and_gateway_activities* **then**
      Step 6: Create a list of *resource consumers*, from resources (performers) of the *first_activities_after_gateway*
      for *consumer* **in** *resource consumers* **do**
        for (*resource performer*, *resource consumer*, *activity*) **in** entry **do**
          if *consumer* == *resource consumer* **then**
            Step 7: Add (*resource performer*, *resource consumer*, *activity*) to *edges*
          end if
        end for
      end for
    else
      Step 8: Add (*resource performer*, *resource consumer*, *activity*) to *edges*
    end if
  end for
  Step 9: Add *trace id*, *corresponding case ids* and the *(nodes, edges)* to the *list of SNG-EL elements*
end for

First, an empty *SNG-EL* graph and edge list are created (**Step 1**). After that, the pre-processed file is parsed to cache for each trace, the resource and activity data (**Step 2**). A Petri net is constructed using a process discovery algorithm (**Step 3**). In this case, we use the alpha miner [2], but also other process discovery techniques can be employed. The Petri net is used to detect information about activities before, in, and after an "AND"-gateway. The activity before the gateway, the first activities of each branch after the gateway, the activities of each "AND"-path and the activity after the closing tag are cached (**Step 4**). Each distinct trace creates a *SNG-EL* graph. Thereby, a list of unique valid nodes is created (**Step 5**). A node is unique if no duplicate exists and valid if it is not a null, undefined or empty value. To create an edge it is first checked if the activity is the start activity before an "AND"-gateway starts. If this is the case, a list of resource consumers is created (**Step 6**) which are basically the resource performers of all follow-up activities that will be performed in the parallel branches. Then, a list of edges is created with a resource performer, resource consumer and activity for each consumer in the gathered list (**Step 7**). Otherwise, the activity is an activity in a sequential path and the resource performer, resource consumer and the activity are added to the list of edges (**Step 8**). Generally, the information of the resource consumer is stored for each activity in each trace in the pre-processed log and is, analogously to most existing work, the performer of the next activity. Lastly, after each activity in the entry is

checked, the trace ID, all corresponding case IDs and the graph object consisting of the list of nodes and edges are added to the list of *SNG-EL* graphs (**Step 9**).

3.3 Compliance Deviation Detection and Visualization

Compliance Deviation Detection. Both, *SNG-T* and *SNG-EL* can contain multiple graphs with those from *SNG-T* being considered as ground-truth. For compliance deviation detection we need to identify which graph from *SNG-EL* should be compared to which ground-truth graph from *SNG-T*. To determine this pair, all graphs from *SNG-EL* are contrasted with all graphs from *SNG-T*. Thereby, their amount of edges and nodes are compared. The graph pair with the relatively highest score, i.e., the most similar number of edges and nodes, is considered a match. Compliance deviation detection then involves analyzing all edges in the graph from *SNG-EL* and identifying any absent edges in the matched ground-truth graph from *SNG-T*. For example, in Fig. 1, in the *SNG-T* graph there is no edge between *"Investigational Review Board"* and *"Pharmacie & Healthcare Providers"* but it is present in the *SNG-EL* graph. Such disparities indicate compliance deviations. The output of this analysis can be used to improve the communication structure or highlight deviations in the process. In an optimal case the graph from *SNG-EL* and the ground-truth graph from *SNG-T* consist of the same nodes. If the graph from *SNG-EL* contains fewer nodes than the one from *SNG-T*, all nodes and edges from and to the nodes are removed from the ground-truth graph to only check the resource interaction of given granularity and data. Conversely, if the *SNG-EL* graph contains more edges or nodes than the ground-truth graph from *SNG-T*, it may be due to a high-level description lacking detail or the process being described across multiple textual descriptions. Such cases do not necessarily indicate a compliance violation.

Visualization. Two visualizations were developed. First, a directed graph representing information exchange between resources. This visualization component can be utilized for graphs in *SNG-T* as well as *SNG-EL*. Prior to visualization creation, the nodes undergo weighting and sizing based on the results obtained from the page-rank algorithm [8], which determines the importance of a node within a process and the volume of traffic passing through that resource. Second, compliance deviations are represented in a graph through red edges, indicating discrepancies previously detected in the communication patterns of resources.

4 Evaluation

The approach has been implemented as a prototype in Python 3 and can be accessed at https://www.cs.cit.tum.de/bpm/software/. All input, and intermediate files, like JSON files from GPT-4 prompt execution as well as the prompt itself, pre-processed event logs, and output files are also available. The evaluation features synthetic and real-world datasets. As synthetic data, the *Bicycle Manufacturing* (BM) and *Schedule Meetings* (SM) datasets as introduced in [16] were considered. Those exist of process descriptions from the PET dataset [7]

and corresponding events logs which were generated using the Cloud Process Execution Engine (CPEE) [15]. Moreover, the running example (RE) as introduced in Sect. 3 is included for which event logs were generated as well. In addition to those three synthetic datasets, the Business Process Intelligence Challenge 2020 (BPIC2020) dataset is used consisting of event logs with resources in verbal form, e.g., *budget owner* and a detailed textual process description. All steps of the approach as outlined in Sect. 3 are evaluated. In particular, Sect. 4.1 presents the evaluation results for text pre-processing and *SNG-T* construction while Sect. 4.2 evaluates *SNG-EL* construction compared to existing social network mining approaches provided by the PM4Py toolkit. The compliance deviation detection and visualization are evaluated in Sect. 4.3.

4.1 Results Social Network Graph Construction from Text

To evaluate the different processes for each text file, a corresponding gold standard is manually created for the pre-processing and *SNG-T* construction step.

Text Pre-Processing. The evaluation of the GPT-4 output encompasses both quantitative and qualitative measures. Quantitatively, the comparison includes extracted network information from the pre-processed text, such as the control-flow and order of resources and activities. Qualitatively, the evaluation examines the assigned labels for the resources and activities. The latter is crucial due to the variation in interpreting resource communication based on textual activities, making it challenging to establish a definitive standard.

Table 1 presents the quantitative evaluation results with precision and recall being calculated to assess the accuracy of the text pre-processing. True positives are identified as correct instances of resource interaction, communication presence meaning the exact prediction of the resource performer and resource consumer, while true negatives represent correct predictions of resource interaction absence. False positives occur when an resource interaction is absent but should be present, and false negatives occur when a resource interaction is present but should be absent. Additionally, three assumptions are considered when evaluating the resource interaction and the matched resource performers and consumers. Resource interactions in the gold standard are not counted as present if the resource is the performer and consumer or if one resource is null, or if the resource interaction can be also be seen as an optional interaction. The results for the PET dataset, specifically for BM, SM, and RE, are promising, demonstrating an overall precision and quality of social network extraction of 1. For the BPIC20 dataset, the results are also satisfactory, considering the inherent difficulty in extracting resource interaction information. For instance, the description mentions situations where *"the budget owner and the supervisor are sometimes the*

Table 1. Precision and Recall for Text Pre-Processing

	BM	**SM**	**RE**	**BPIC20**
Precision	1	1	1	0.57
Recall	1	1	0.86	0.67

same" and the *"director is not always present"*. From a qualitative perspective, in terms of activity and resource labelling and summary by the model, the results are consistently good across all datasets. In the BM dataset, variations such as *"member of the sales department" and "sales department"* are effectively merged into a single resource. Activities in the BPIC20 as *"submit and send"* are merged and only create one resource interaction instead of multiple ones.

SNG-T Construction. It is important to note that the results for this part of the approach strongly rely on the preceding outputs of GPT-4. Precision and recall are calculated by averaging the values across all different graph objects created by Alg. 1. For instance, for the BM, and BPIC20 datasets, the *SNG-T* set contains two ground-truth graphs.

The overall results, presented in Table 2, are once again promising. Algorithm 1 successfully creates the expected number of graph objects for each dataset, such as two for BM, and BPIC20, and one for SM, and RE confirming the effectiveness of the introduced conditional split of graph objects. While the labelling of activities is less significant in this evaluation, the focus lies on the creation of edges between resources. False negatives are detected when the labelled activity in the graph object appears nonsensical. For example, in a graph object of the BPIC20 the resources "x" and *"y"* are matched based on the activities *"accept request"* and *"reject request"*, one of these two edges created in this scenario would be considered as a false negative edge presence.

Table 2. Precision and Recall for *SNG-T* Construction

	BM	**SM**	**RE**	**BPIC20**
Precision	1	1	0.93	0.5
Recall	1	1	0.86	0.585

4.2 SNG-EL Construction Compared to Existing Solutions

To assess the *SNG-EL* construction, the resulting social network graphs are compared with those generated using the established *handover of work* and *working together* implementations for analyzing collaborative patterns in process log available in the PM4Py toolkit. The *handover of work* approach involves creating an undirected graph that captures the frequency with which another follows one resource within the process log. This graph provides insights into potential handovers of work between resources during the execution of activities. The *working together* metric focuses on quantifying the instances in which different resources collaborate to complete activities within a trace. This results in a directed graph in which nodes are assigned weights based on their frequency of occurrence in various process instances. Table 3 presents the results, i.e., the amount of constructed graphs in *SNG-EL* for each dataset, how many graphs in *SNG-EL* contain the same nodes as in the handover of work graph (HoW) in relation to all constructed graphs which are in *SNG-EL*, how many graphs in *SNG-EL* visualize the same graph as the working together (WT) graph in relation to all existing graphs stored in *SNG-EL* and how many graphs visualizes a different graph with respect to nodes and edges, as the WT graph in relation to all existing graphs in *SNG-EL*.

Table 3. *SNG-EL* comparison with *working together* and *handover of work*

	BM	**SM**	**RE**
Amount Graphs	5	4	3
Same as HoW	0.8	1	0.67
Same as WT	0.8	1	0
Different to WT	0.2	0	1

For the BM dataset, five distinct *SNG-EL* graphs were constructed. The sales department node was connected with the engineering department node and the storehouse node in the HoW graph. Similarly, the WT graph featured several directed edges representing collaborative interactions, including reflexive edges. Notably, four out of these five *SNG-EL* graphs resulted in identical graphs to the WT graph. One of the *SNG-EL* graphs depicted a solitary node representing the sales department. This node was connected to itself through a reflexive edge, signifying the exclusive involvement of this department throughout the entire process. This singular representation was a consequence of a unique process instance wherein the sales department received an order but promptly rejected it which is not reflected in a WT or HoW graph. All four resulting graphs in *SNG-EL* of the SM event log contain the same nodes as in HoW and all illustrate the same graph as the WT graph. Two-thirds of the graphs *SNG-EL* set of the RE event log contain the same nodes as in the HoW graph but no graph corresponds to the WT graph. This circumstance highlights the distinctiveness of our approach, as it not only captures typical collaborative dynamics but also accommodates exceptional cases that existing methods might overlook. Evaluating the BPIC20 dataset was challenging. It consists of multiple large event logs which are rarely connected with each other. Therefore, the working together graph and Petri net, which is mandatory to check parallel cases, are too complex and a quantitative comparison becomes infeasible.

4.3 Compliance Deviation Detection and Visualization

We evaluate whether the matching between graphs from *SNG-T* and *SNG-EL* is correct, and whether compliance deviations were correctly identified. Since user studies on the usefulness of the visualizations are beyond the scope of the paper those are only evaluated in the sense of whether results are depicted correctly. For BM, two different *SNG-T* graphs were created, and the event log contained five distinct traces. All *SNG-EL* graphs were correctly matched with a corresponding *SNG-T* graph, including matches between *SNG-EL* graph 1 and *SNG-T* graph 1, and *SNG-EL* graphs 1, 3, 4, and 5 with *SNG-T* graph 2. The detected violations in the BM event log were also all correctly identified. For the SM dataset, *SNG-T* contains one graph and *SNG-EL* four graphs. All *SNG-EL* graphs were correctly matched with their corresponding *SNG-T* graph, as there was only one possible match. The approach accurately detected compliance deviations in all cases. In RE, which consisted of three *SNG-EL* graphs and one *SNG-T* graph, all matches were correct, and all deviating edges were identified. The deviations included direct interactions between the *"pharmaceutical company"* and the *"ethics committee"*, as well as the *"return of review protocols"* from the *"ethics committee"* to the *"investigational review board"*. Figure 2 shows the compliance deviations of the running example's first trace, i.e., the

"ethics committee" sends the reviews to the "investigational review board" being a compliance deviation as the interaction was not captured in the *SNG-T* graph. This suggests either unwanted communication or the transmission of data that should not have been sent. The *SNG-EL* construction for BPIC20 delivered too complex results, so the compliance deviation detection could not be evaluated.

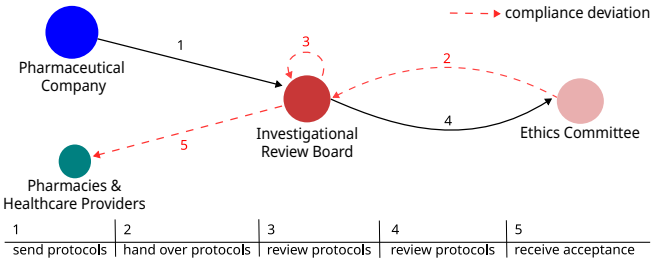

Fig. 2. Example Output Depicting Compliance Deviation for Running Example

5 Discussion and Conclusion

This work presents social network graph construction from natural language text and event logs for compliance deviation detection. First, resource interactions based on natural language texts like process descriptions using an LLM are identified and multiple *SNG-T* graphs are constructed serving as reference models for compliance deviation detection. Furthermore, the approach includes a novel algorithm that identifies more precisely the consumer of a task by taking parallel control-flow into account. The result is a set of *SNG-EL* graphs. Based on *SNG-T* and *SNG-EL*, resource interactions are identified and visualized.

The *SNG-T* construction from natural language text demonstrates promising outcomes in terms of identifying nodes and edges, facilitating the identification of interactions and communication between resources in a process. Nevertheless, some limitations have been identified. The use of large language models like GPT-4 poses challenges (cf. [16]) in terms of reliability, and transparency, particularly when labelling edges with activity names, which may differ from human-assigned labels. Additionally, the *SNG-T* construction algorithm for text requires improvement to appropriately handle the splitting of graph objects in the presence of nested conditions. The algorithm to construct *SNG-EL* elements currently neglects cases where multiple activities exist within each parallel activities path. Furthermore, the compliance deviation component assumes that matching *SNG-EL* and *SNG-T* elements entails the same resource and activity labels in the text and the event log, ideally considering word similarities.

Future work will involve implementing a pre-processing component independent of GPT-4 for extracting requirements from natural language text and comparing its results with the GPT-4-based solution. Moreover, leveraging social

network models, additional mathematical assumptions, statistical calculations, and expert knowledge can be employed to generate an "organigram" for more accurate resource-activity compliance deviation detection as discussed in [16].

Acknowledgements This work has been partly funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – project number 514769482.

References

1. van der Aalst, W.M.P., Song, M.: Mining social networks: Uncovering interaction patterns in business processes. In: Business Process Management. pp. 244–260 (2004). https://doi.org/10.1007/978-3-540-25970-1_16
2. van der Aalst, W.M.P., Weijters, T., Maruster, L.: Workflow mining: Discovering process models from event logs. IEEE Trans. Knowl. Data Eng. **16**(9), 1128–1142 (2004). https://doi.org/10.1109/TKDE.2004.47
3. Abdelkafi, M., Mbarek, N., Bouzguenda, L.: Mining organizational structures from email logs: an NLP based approach. In: Knowledge-Based and Intelligent Information & Engineering Systems. pp. 348–356 (2021). https://doi.org/10.1016/j.procs.2021.08.036
4. Appice, A.: Towards mining the organizational structure of a dynamic event scenario. J. Intell. Inf. Syst. **50**(1), 165–193 (2018). https://doi.org/10.1007/s10844-017-0451-x
5. Barrientos, M., Winter, K., Mangler, J., Rinderle-Ma, S.: Verification of quantitative temporal compliance requirements in process descriptions over event logs. In: Advanced Information Systems Engineering. pp. 417–433 (2023). https://doi.org/10.1007/978-3-031-34560-9_25
6. Bauer, D., Longley, T., Ma, Y., Wilson, T.: NLP in human rights research - extracting knowledge graphs about police and army units and their commanders. CoRR (2022), https://arxiv.org/abs/2201.05230
7. Bellan, P., Dragoni, M., Ghidini, C.: Extracting business process entities and relations from text using pre-trained language models and in-context learning. In: Enterprise Design, Operations, and Computing. pp. 182–199 (2022). https://doi.org/10.1007/978-3-031-17604-3_11
8. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. Computer Networks and ISDN Systems **30**(1), 107–117 (1998). https://doi.org/https://doi.org/10.1016/S0169-7552(98)00110-X
9. Busch, P., Fettke, P.: Business process management under the microscope: The potential of social network analysis. In: Hawaii International Conference on System Sciences (2011). https://doi.org/10.1109/HICSS.2011.93
10. Dessì, D., Osborne, F., Reforgiato Recupero, D., Buscaldi, D., Motta, E.: Generating knowledge graphs by employing natural language processing and machine learning techniques within the scholarly domain. Future Generation Computer Systems **116**, 253–264 (2021). https://doi.org/10.1016/j.future.2020.10.026
11. Ebrahim, M., Golpayegani, S.A.H.: Anomaly detection in business processes logs using social network analysis. J. Comput. Virol. Hacking Tech. **18**(2), 127–139 (2022). https://doi.org/10.1007/s11416-021-00398-8
12. Friedrich, F., Mendling, J., Puhlmann, F.: Process model generation from natural language text. In: Advanced Information Systems Engineering. pp. 482–496 (2011). https://doi.org/10.1007/978-3-642-21640-4_36

13. Gao, A., Yang, Y., Zeng, M., Zhang, J., Wang, Y.: Organizational structure mining based on workflow logs. In: Business Intelligence: Artificial Intelligence in Business, Industry and Engineering. pp. 455–459 (2009). https://doi.org/10.1109/BIFE.2009.109
14. Ly, L.T., Rinderle, S., Dadam, P., Reichert, M.: Mining staff assignment rules from event-based data. In: Business Process Management Workshops. vol. 3812, pp. 177–190 (2005). https://doi.org/10.1007/11678564_16
15. Mangler, J., Rinderle-Ma, S.: Cloud process execution engine: Architecture and interfaces (2022). https://doi.org/10.48550/ARXIV.2208.12214
16. Mustroph, H., Barrientos, M., Winter, K., Rinderle-Ma, S.: Verifying resource compliance requirements from natural language text over event logs. In: Business Process Management (2023). https://doi.org/10.1007/978-3-031-41620-0_15
17. Ni, Z., Wang, S., Li, H.: Mining organizational structure from workflow logs. In: Proceeding of the International Conference on e-Education, Entertainment and e-Management. pp. 222–225 (2011). https://doi.org/10.1109/ICeEEM.2011.6137791
18. OpenAI: GPT-4 technical report. CoRR **abs/2303.08774** (2023). https://doi.org/10.48550/arXiv.2303.08774
19. Pika, A., Leyer, M., Wynn, M.T., Fidge, C.J., ter Hofstede, A.H.M., van der Aalst, W.M.P.: Mining resource profiles from event logs. ACM Trans. Manag. Inf. Syst. **8**(1), 1:1–1:30 (2017). https://doi.org/10.1145/3041218
20. Qin, S., Xu, C., Zhang, F., Jiang, T., Ge, W., Li, J.: Research on application of chinese natural language processing in constructing knowledge graph of chronic diseases. In: 2021 International Conference on Communications, Information System and Computer Engineering (CISCE). pp. 271–274 (2021). https://doi.org/10.1109/CISCE52179.2021.9445976
21. Raitubu, N., Sungkono, K.R., Sarno, R., Wahyuni, C.S.: Detection of bottleneck and social network in business process of agile development. In: 2019 International Seminar on Application for Technology of Information and Communication (iSemantic). pp. 208–213 (2019). https://doi.org/10.1109/ISEMANTIC.2019.8884341
22. Reijers, H.A., Song, M., Jeong, B.: Analysis of a collaborative workflow process with distributed actors. Inf. Syst. Frontiers **11**(3), 307–322 (2009). https://doi.org/10.1007/s10796-008-9092-5
23. Sellami, R., Gaaloul, W., Moalla, S.: An ontology for workflow organizational model mining. In: Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises. pp. 199–204 (2012). https://doi.org/10.1109/WETICE.2012.29
24. Song, M., van der Aalst, W.M.P.: Towards comprehensive support for organizational mining. Decis. Support Syst. **46**(1), 300–317 (2008). https://doi.org/10.1016/j.dss.2008.07.002
25. Tao, J., Deokar, A.V.: An organizational mining approach based on behavioral process patterns. In: Americas Conference on Information Systems. Association for Information Systems (2014), http://aisel.aisnet.org/amcis2014/EndUserIS/GeneralPresentations/11
26. Yang, J., Ouyang, C., van der Aalst, W.M.P., ter Hofstede, A.H.M., Yu, Y.: *OrdinoR*: A framework for discovering, evaluating, and analyzing organizational models using event logs. Decis. Support Syst. **158**, 113771 (2022). https://doi.org/10.1016/j.dss.2022.113771
27. Zhao, W., Zhao, X.: Process mining from the organizational perspective. Advances in Intelligent Systems and Computing **277**, 701 – 708 (2014). https://doi.org/10.1007/978-3-642-54924-3_66