Outline:

Abstract: describe four main points of this method.

Introduction: Describe the context: spaghetti environments. Describe key terminology to be used.

Previous Work: SUBDUE, GBAD, Genga.

Approach:

Experiment/Results:

Conclusions and Future Work

Abstract

This paper discusses an unsupervised, threshold-based method for anomaly detection and normative-pattern mining using the iterative SUBDUE graph compression method and the Inductive Miner. We provide an overview of process mining definitions and existing approaches, then test the method on synthetic data provided by a data generation algorithm (eg, from BEZERRA). This unsupervised approach requires a single parameter describing the anomaly threshold. In turn, the method generates a taxonomical dendrogram of compressing process features by which further automated or human analysis can be performed. This method provides a framework by which many organizational, technological, and natural processes can be monitored and modeled.

Introduction

As described in [VDA], a Process Aware Information System is defined as *a software system that manages and executes operational processes involving people, applications, and/or information sources on the basis of process models [VDA].* This formal definition simply distinguishes operational management systems as systems which are both aware of process data and likewise prescribe tasks and activities per well-defined process models. Thus, such systems incorporate process data and process models in a feedback loop whereby processes can be tracked, defined, and organized according to process models; likewise, process models may be derived and analyzed on the basis of process data.

In real-world scenarios, enterprises rely on many overlapping but non-interoperable systems, enterprise management systems, and tribal knowledge to monitor and control processes, such that the over-arching requirements of formal PAIS’s become infeasible or intrusive. This paper focuses on contexts in which a PAIS is instead an abstraction consuming noisy process data derived from multiple systems, from which process models can be mined and analyzed to improve prescribed processes. This interpretation is more amenable to scenarios in which processes execute within an embedded framework of changing people, tools, and institutional knowledge, when there often is no prescribed definition for the underlying process model. In this context, a PAIS is a collection of operational systems and disparate data sources by which one derives traces characterizing the underlying process-oriented view of a department, a workflow, and so forth.

In this context, the ability to mine and analyze normative process patterns is critical for extracting actionable information and detecting unusual activity. Detecting unusual activities requires developing some prior normative activity patterns by which anomalous behavior can be detected, hence these are complementary tasks. To this end, we present a method for mining process patterns from workflow logs that exhibits useful anomaly detection properties, using the Inductive Miner to construct a general process model, then applying the SUBDUE graph compression method to extract the normative activity patterns of this model.

This hybrid approach is useful since the Inductive Miner extracts generality from process log data, outputting a graphical model *M,* capable of generating all traces in a process log, regardless of how overly-inclusive the model may be or how noisy the input log. SUBDUE then reductively infers a hierarchy of structures of *M* most relevant to the log as a dendrogram, extracting from M only its most informative component sub-processes. In short, for a given trace log, the Inductive Miner is used to mine the most general process model, then SUBDUE is used to extract only the most specific components of the model. Using this unsupervised method, one can mine normative process patterns and detect anomalies to those patterns.

The preceding captures the general spirit and contribution of our method, but some definitions are in order. From the control-flow perspective, common process mining terms can be framed in a more familiar graph-theoretic manner:

-process model: A graph whose vertices represent activities, and whose edges represent transitions between activities. Naturally, processes can contain many constructs representing linear and non-linear constructs, and a variety of notations and languages have been defined over the space of process models. The notable example is the Petri-Net [CITE].

-process grammar: Constructs defining common process traversal patterns. An AND-SPLIT for instance is a set of edges branching from a single node and traversing activities in parallel before synchronizing at some later activity. Other constructs include OR-SPLIT, XOR, LOOP, JOIN, and so on.

-workflow trace: A single execution of a process represented as a partially-ordered sequence of activities, which is any valid path from some START node to an END node on the process model. This is best thought of as a string composed of letters representing the activities traversed.

-workflow log: A set of workflow traces. Like any dataset, this is expected to be noisy and incomplete, for which pre-processing or parametric measures are taken to mitigate the properties of the log.

-process miner: An algorithm for deducing a process model from a workflow log, according to criteria such as specificity and generality. Specificity restrictively favors models including only (or less than) the behavior described by the traces, whereas generality favors larger models describing all traces and likely also including behavior not included in the traces.

-partial-order property: The property by which activities within a workflow trace are assumed to be randomly ordered with respect to parallel activities. For instance, if ‘ABCD’ and ‘ACBD’ are workflow traces from some model, where ‘C’ and ‘B’ are parallel sub-processes, and may themselves be recursively embody further parallel sub-processes. Given the arbitrary complexity of processes, the task for process mining algorithms is disambiguating the partial orderings given by the traces of a log, by applying various rules and heuristics per pre-defined criteria of specificity and generality.

-spaghetti model: A workflow defined by highly diverse, informal, and possibly disorderly behavior, typically containing lots of scattered repetitive events. These represent immature business processes, in contrast to “lasagna” processes which have more structured, stratified behavior logic.

-inductive miner: A process mining algorithm which builds a process model that generalizes the log traces by applying a few logical rules to the constraints given by the partial orderings in the traces of a log [CITE]. It suffices for this work to view the Inductive Miner as the mining algorithm which, given a log, outputs a process model capturing the most general, all-inclusive view of the traces. It is used in this manner as a pre-processing step, generating a graphical process model capturing the *necessary* logical constraints given by a log. Other discriminative methods are applied in post-processing to generate a *sufficient* view of the traces per user-defined criteria.

-SUBDUE: Short for “Substructure Discovery”, this method searches across a collection of graphs, and by applying the mdl-principle, returns the top-k most compressing sub-graphs [CITE].

-GBAD: Short for “graph-based anomaly detection”, this method internally calls SUBDUE, and then implements various methods for detecting anomalies which occur in the context of the discovered patterns.

A more in-depth overview of process mining formal languages, goals, data sources, and mining algorithm properties can be found in [VDA book].

Previous Work

The SUBDUE graph-compression method naturally extends any context in which one is interested in mining graph data for normative patterns. It was previously used for knowledge representation systems [CITE EARLIEST], and more recently in security applications for intrusion detection. Previous work pertinent to process mining has been successfully performed by [GENGA et al]. Their results have shown the method’s utility for “spaghetti processes” describing more real-life processes in which loose collaborations and external conditions create informal processes that are disorderly, high degree, high concurrency, or iterative.

GBAD has also been deployed to detect anomalous activities, but only in the neighborhood of discovered normative patterns. This is befitting of contexts in which there exists some underlying, prescribed process model according to which normative patterns can be assumed to have some policy ground-truth, but not when the overall process is less formal.

Bezerra’s work on anomaly detection, the primary reference for this work, examined anomaly detection using several threshold-based approaches within the process-mining algorithm itself. Bezerra decomposes the predominant categories of process-based anomaly detection into two groups: threshold-based and iterative. Though closely related, our approach does not fit into these categories, as it is compression based: the process model is discovered, the log is then iteratively compressed away, and any traces below some hard threshold are reported as anomalies. Except for iterative compression, the method is effectively one-step, generating a model in the form of a dendrogram, and reporting all components of that dendrogram not satisfying some certain features. As such, our work replicates some of Bezerra’s data generation methods, but otherwise builds on this work by applying anomaly detection in post-processing of the discovered process model.

The Method

Under our relaxed definition, a PAIS is a composition of process monitoring systems from which log representations can be extracted for the purposes of process mining. We assume such extraction and transformation steps have already occurred, and the process information has been converted into a popular log format such as XES. Our goal decomposes to three tasks: mine the generalized process model describing the workflow log, extract its most descriptive patterns, and finally detect outliers and anomalies to these normative patterns.

For the first of these tasks, the Inductive Miner is clearly a suitable mining method for generating the most general process model described by some log. Such a model is overly inclusive, hence the second and third tasks discover those patterns and features most relevant to describe the log. For this task, we use the SUBDUE graph-compression method to discover the normative patterns, which in turn allow us to discover anomalies.

The workflow of these tasks is extensible to any context in which one doesn’t wish to discover process features per rigidly prescribed process definitions. Instead, it is extensible to any context in which one wishes to discover the overall characteristics of a process without respect to such prior constraints. Our approach applies to more realistic “spaghetti” model scenarios in which processes are organically-defined to the point of being somewhat chaotic, making it more extensible to any context which can be so represented: enterprises, social networks, fraudulent or criminal networks, and so on.

Using Graph Compression to Discover and Cluster Patterns

The SUBDUE [Holder] method discovers highly compressing patterns in graph data using the notion of minimum description length to discover and output the most compressing pattern describing sets of graphical data. SUBDUE works by searching across the set of all subgraphs within a set of input graphs for the most highly-compressing pattern(s) according to the m.d.l. principle.

[more on SUBDUE?]

This property of SUBDUE answers the requirement for an unsupervised method of discovering the most meaningful components of some process model, since a trace log can be viewed as a set of subgraphs generated by a process model. The Inductive Miner complements our approach, since it provides the super-graph (a process model) by which the log traces can be converted from partially-ordered activities into subgraphs; these subgraphs are then passed to SUBDUE to discover meaningful patterns.

Prior work on SUBDUE showed great potential when running the method iteratively on a set of graphs [NOBLE COOK]. At each iteration, the most compressing subgraph discovered by SUBDUE is used to replace all such instances with a single node, and then the method repeats until no further compression is possible. At the end, one obtains a recursive and loosely hierarchical description of a set of graphs, in which the graph has been compressed away by recursively-defined subgraphs, all of which have been aliased and replaced by single nodes.

Using the GBAD system (which calls SUBDUE internally) we tested a similar approach, by which workflow traces were iteratively recompressed using the most-compressing subgraph found at each iteration. The three versions of GBAD were then used to detect anomalies at each iteration. While partially successful in terms of discovering patterns, this method suffered a very high false positive rate for anomaly detection. Ultimately, the problem lies with iterative recompression: on successive iterations, the most highly compressing subgraph was often only a small alteration (substitution, deletion, or insertion) to a compressing subgraph found on a previous iteration. GBAD’s primary deficiency in this context is that its anomaly-detection methods only apply to the local vicinity of the compressing pattern discovered by SUBDUE. Hence, the search space was highly redundant, often analyzing the same regions of the graph, whilst failing to reach the further reaches where compressing structure begins to decay, and where anomalies often lie.

Since the requirement was to force SUBDUE to search in new regions of the graphs, the solution is quite simple: at each iteration, delete all instances of the most-compressing subgraph from the traces. The net effect of this method is that SUBDUE is encouraged to find compressing graphical features in new regions among a set of graphs. This ends up being quite amenable to anomaly detection, as the method makes much faster progress compressing away primary graphical features until only the least compressing features remain. Additionally, the method generates a natural, hierarchical derivation of process substructures in the form of a dendrogram, which can be leveraged analytically or may simply be inspected by a human observer for process patterns.

In sum, this gives the following pattern-mining and anomaly-detection algorithm:

[algo]

As shown, the Inductive Miner is run on a workflow log to give a process model by which the traces can be re-generated not as sequences but as subgraphs. The subgraphs are fed to SUBDUE to find the most compressing subgraph, which is then deleted from all traces in which it occurs. The method repeats, until no further progress can be made as traces are gradually deleted when they contain no other structure by some compressing substructure.

Visually, the success of this method lies in the dendrogram it produces. This is useful both analytically and for manual-inspection of process features, redundant behavior, outlier behavior, and so on. [GENGA et al] have successfully detailed the many uses for such dendrograms using highly similar methods, especially in the context of spaghetti processes cohering to no strict process definition.

This method accurately belongs to the family of dendrogram- or tree-induction methods which occur frequently in process mining literature [CITE SOME]. In this context, anomaly detection is just one instance among many in which one is querying the dendrogram according to some statistical or other criteria, for which many other purposes could be devised. For instance, whilst the low-frequency, outlier components of the dendrogram may characterize anomalies, the ancestral components can be interpreted as encoding the most relevant substructures of the workflow log, by which the process model output by the Inductive Miner could be reduced to a more concise process description. Thus, coupling SUBDUE with the generalization qualities of the Inductive Miner creates an extensible framework for more concise modelling activities in the context of disorderly spaghetti models. Similarly, an analyst may find components of the dendrogram which are highly similar, and thus may represent duplicate work or poor cohesion amongst business processes. In this manner, using this method extends to a range of pattern mining and other enterprise uses, beyond the scope of anomaly detection.

Anomaly detection lends a particularly illustrative example in this context because of the structural characteristics foregrounded by the dendrogram: given that anomalies are assumed to be infrequent events, subgraphs containing these will be among the last components to be compressed. The result is that the size of the dendrogram components decreases smoothly, but then drops suddenly, such that the only remaining traces/subgraphs are those representing anomalies, outliers, or simply noise in the log.

[plot size of dendrogram components, per iteration?]

Algorithm Evaluation

Although real process-oriented datasets are available, they do not offer the controlled conditions sufficient to compare the characteristics of different algorithms. We instead opted to use a synthetic data generation algorithm outlined in [BEZERRA, Appendix A], which was reproduced as faithfully as possible. This approach generates random process models from which synthetic traces are generated, and thus the performance of an anomaly detection method can be assessed with respect to a known process model. Likewise, to cohere to a stable performance baseline, we also used the same experimental parameters as described in [BEZERRA]: 60 randomly-generated process models,

Results

Conclusions and Future Work

The method presented here demonstrates the desirable qualities shared by any anomaly detection approach: strong normative pattern definitions, and strong separation between anomalies and normative patterns. This method has been shown to satisfy both requirements, strongly distinguishing anomalous traces, and generating normative patterns by which other process mining tasks can be performed. Notably, these properties satisfy a range of other process mining and monitoring tasks, as well as other process and graph mining tasks in fields such as biology, pharmacology, and chemical interaction.

The drawback to this method is its lack of noise-tolerance, a common problem faced by mining algorithms. While SUBDUE can find graphical patterns in an unsupervised manner, any pattern that it finds become the only pattern by which the log is compressed further; that is, even small deviations to the normative pattern are ignored, and may be flagged as anomalies later. From an anomaly-detection perspective, this strongly discriminatory behavior is desirable. On the other hand, from a process mining perspective, the goal is often to mitigate such strong discrimination, and to create noise-tolerant algorithms yielding more *general* process models. This yields the recurring discussion on specificity-generalization tradeoffs, to which all process mining approaches are subject. Fruitful future work lies in making the approach more noise tolerant, similar to how the GBAD system determines acceptable deviations in the local context of a normative pattern discovered by SUBDUE using graph distance metrics.

Related and Future Work

Previous work applying SUBDUE to process mining has been performed successfully by [GENGA et al]. [describe some of Laura’s papers]. This group has noted the method’s utility for what are often described as “spaghetti processes” describing more real-life processes in which loose collaboration and external conditions create informal processes that are typically disorderly, high degree, highly-concurrent, or iterative. The value of SUBDUE in this context is that it finds graphical patterns in an unsupervised way, without the rigid formalisms and drawbacks of process mining approaches for which highly concurrent and iterative processes present a significant algorithmic bottleneck.

Bezerra’s paper, the primary reference for this work, decomposes the predominant categories of process-based anomaly detection into two groups: threshold-based and iterative. Although closely-related, our approach does not fit neatly into either of these categories, as it is compression based: the log is iteratively compressed away, and any traces below some hard threshold are reported as anomalies. Except for iterative compression, the method is effectively one-step, generating a model in the form of a dendrogram, and reporting all components of that dendrogram that do not satisfy some certain features.

Ubiquitous computing populate enterprise databases with traces which, from the perspective of process mining, contain “process aware” data. Mining such data provides valuable insights into the structural characteristics of various process flows, and into the normative patterns by which anomalies can be detected in an unsupervised manner.

-describe traces, “process-aware” data, partial orderings, trace-logs

-describe use cases: structural analysis and anomaly detection. Benefit of our method is it is unsupervised.

-mention that discovered anomalies can be used to discover other anomalous instances in the log, results from doing so

-mention the overarching aim that the dendrogram characteristic is amenable to process-mining/anomaly detection in that well-defined process *ought to be* compressible; hence the well-defined, well-enforced process definitions yield better awareness of anomalies when they occur. Therefore the dendrogram “elbow” perspective is amenable to process mining, for which at least some underlying process structure can be estimated.

Definitions of Anomalies and Normative Patterns: focus on the “bump” in the dendrogram

Prior: Formulate goals of a process mining approach to anomaly detection, now describe how that maps onto the use of SUBDUE to accomplish these ends. Describe benefits of an algorithm used to derive partial-order mappings (process models) using the inductive miner.

Experiment

One of the primary difficulties in testing process mining strategies is a lack of stable experimental baselines for evaluating the properties of different methods, whether the methods are for process mining or anomaly detection (although the latter typically entails the former).

Definitions:

Anomaly: An anomaly is a trace that occurs in the context of a normative pattern.

Outlier: Outliers are defined as noise due to variance in some process’ execution. These are typically not strongly-associated with the underlying process model.

Refer to this work for graph anomaly definitions:

http://www3.cs.stonybrook.edu/~leman/pubs/14-dami-graphanomalysurvey.pdf

Related work:

Genga et al, multiple works using SUBDUE to generate process descriptions. Other trace analyzers and anomaly detection methods.

Conclusion