Inf Syst Front (2015) 17:1209–1226
DOI 10.1007/s10796-015-9563-4

Semantics-based event log aggregation for process mining and analytics

Amit V. Deokar[1] · **Jie Tao**[2]

Published online: 9 June 2015

Abstract In highly complex and flexible environments, event logs tend to exhibit high levels of heterogeneity, and clustering-based methods are candidate techniques for simplifying the mined process models from the process observations. To compensate for the information loss occurring during clustering, semantic information from event logs may be extracted and organized in the form of knowledge structures such as process ontologies using methods of ontology learning. In this article, we propose an overall computational framework for event log pre-processing, and then focus on a specific component of the framework, namely event log aggregation. We develop a detailed system architecture for this component, along with an implemented and evaluated research prototype *SemAgg*. We use phrase-based semantic similarity between normalized event names to aggregate event logs in a hierarchical form. We discuss the practical implications of this work for learning lower level process ontology classes as well as performing further process mining and analytics.

✉ Amit V. Deokar
amit.deokar@psu.edu

Jie Tao
jtao@fairfield.edu

[1] Sam and Irene Black School of Business, Pennsylvania State University, 5101 Jordan Road, Burke Center 268, Erie, PA 16563, USA

[2] Information Systems and Operations Management (IS&OM) Department, Dolan School of Business, Fairfield University, 1073N. Benson Road, Fairfield, CT 06824, USA

Keywords Process mining · Process analytics · Event logs · Natural language processing · Process ontologies · Agglomerative hierarchical clustering

1 Introduction

With business process management (BPM) playing an increasingly vital role in organizational strategy in general, and IT strategy in particular, deriving insights about extant processes has become an imperative need for organizations. Derived from data mining, process mining is a technique that facilitates the extraction of *tacit* knowledge from process logs (also known as *event logs*) that are generated by various information systems (Van der Aalst and Weijters 2004). Process mining can in turn provide *process analytics*, i.e., insights into the process structure, semantics, and operations through mining information from observations such as event logs. Simplifying mined process models has become one of the major concerns of the *process mining and analytics* (PMA) communities (Van der Aalst 2008).

Given the voluminous amount of event logs that are routinely generated by various organizational information systems, the complexity of the mined process models can become quite high, especially in environments that are characterized by high levels of complexity, flexibility and heterogeneity. Directly applying standard process mining techniques typically leads to "spaghetti-like" process models, which can be difficult to understand, even for domain experts (Song et al. 2009). As a case in point, we considered a real-world event log generated from the cancer diagnosis process at a Dutch academic hospital. Each process instance corresponds to a patient. The raw event log contains over 150,000 tasks in 1143 instances (Van Dongen et al. 2011). These tasks belong to 624 unique events. Directly applying process mining

approaches on the raw event logs leads to incomprehensible and overwhelming process models. We have applied one of the most popular process mining algorithms, the α-algorithm, on the raw logs (Van Dongen and van der Aalst 2004; Weber et al. 2013). Despite the high computational power, the mined process model containing over a few hundred nodes and arcs was cognitively unmanageable.

In such a case, pre-processing the event logs, perhaps through clustering, can be useful. However, this raises two questions. 1) How to compensate for the information loss in the clustering procedure? 2) Which is the appropriate clustering technique and associated metric(s) for preprocessing? As we show, these issues can be addressed in an interrelated manner. Rich semantics embedded in the textual context in the event logs may be used to offset the information loss through the clustering procedure (Greco et al. 2006; Song et al. 2009). We construct an ontology as the underlying knowledge structure for identifying and extracting such semantic information, as well as for utilizing the implicit semantic information later with reasoning and analytic techniques. The more fine-grained the ontology, the more semantic information can be captured that is relating the events in the log, which in turn can help reduce information loss in the clustering process. Further, leveraging the structure of the ontology as the basis for clustering event logs can provide a useful and effective mechanism for end users such as business analysts to perform business process analytics.

As such, in this study, we propose a framework for pre-processing event logs, and focus on the event log aggregation phase of this framework. The resultant aggregated event logs with a hierarchical structure that are generated through this phase can be used for learning lower level process ontology classes. We also propose an event log normalization method that improves concept-disambiguation and normalizes event logs, particularly focusing on event names. The framework also incorporates an event log aggregation approach based on phrase-based semantic similarity between normalized event names as the clustering metrics.

The article is organized as follows. Section 2 identifies research gaps through a review of prior related studies. Based on these research gaps, Section 3 articulates the design requirements as well as proposes an event log pre-processing framework based on these requirements. Section 4 focuses on the semantic event log aggregation aspect of the overall framework and presents a systems architecture along with specific technical details of each of its aspects. Section 5 presents the instantiation of the aforementioned design artifacts within a research prototype, *SemAgg*, together with the demonstration of its functionalities through a representative use case. Section 6 presents the evaluation of the proposed log aggregation approach through a computational experiment on a set of real-world event logs. Section 7 presents discussion on the practical implications of the work. Finally, Section 8 concludes the article.

2 Related studies

Various approaches for simplifying mined process models have been proposed in the literature. They include computational methods such as advanced mining algorithms (Alves de Medeiros et al. 2006), pattern-based methods (Bose and van der Aalst 2009; Edgington et al. 2010), and so forth. Among these approaches, applying clustering-based methods as a pre-processing stage of process mining has been proven to be the most promising method (Song et al. 2009). These pre-processed clusters of event logs can be used for further PMA activities related to process discovery, conformance checking, and model enhancement (Van der Aalst et al. 2007).

Even though clustering-based approaches can increase the simplicity and generalizability of process models, they also lead to a loss of information during the clustering process. Integrating semantic information extracted from the event logs with the mined process models can possibly alleviate this information loss issue. Ideally, a formalized knowledge structure such as process ontologies that systematically reflects concepts and the relations between them could be used as a basis for event log clustering. However, existing process ontologies (e.g., MIT process handbook (MIT-PH) (Malone et al. 2003)) are usually generic and lack domain-specificity. Accordingly, lower level process ontologies are required to have finer granularity and capture domain-specific details through event classes and hierarchical relationships between these classes. These lower level process ontologies can then be reconciled with existing upper level process ontologies (e.g., MIT-PH) using ontology learning and ontology integration techniques leading to subsequent PMA activities. In that regard, creating lower level process ontologies from event logs in a bottom-up manner is a critical step in event log pre-processing.

2.1 Process mining and analytics (PMA)

Process mining has received increased attention from BPM researchers over the past decade (Tiwari et al. 2008). Process mining is a technique that builds abstractions of business processes (in the form of *process models*) based on the analysis of the process executions (in the form of *process/event logs*) (Van der Aalst et al. 2007). Two main areas have been highlighted in the context of process mining, namely *computational intelligence/data mining* and *process modeling/analysis* (IEEE Task Force on Process Mining 2011).

Various *data mining* techniques have been employed in the process mining area, along with natively developed approaches. Table 1 summarizes representative computational intelligence techniques within the different categories. Among these different categories, clustering-based process mining approaches have been shown to be the most promising ones (Greco et al. 2006; Tiwari et al. 2008). It should be noted

Table 1 Summary of process mining techniques

Category	Key study	Contribution
Customized Algorithms – native in the process mining field	Alves de Medeiros et al. (2004)	The α-algorithm: trace-based process mining algorithm
	Weijters et al. (2006)	The heuristic-miner algorithm: causal-link-based process mining algorithm
	Van Dongen and van der Aalst (2004)	Extended α-algorithm for mining loops in process models
Soft Computing Approaches – adopted from the Artificial Intelligence field	Alves de Medeiros et al. (2006)	Genetic algorithm for process mining
	Günther and van der Aalst (2007)	Fuzzy-logic based process mining
Temporal Analysis Approaches – adopted from the Artificial Intelligence field	Veiga and Ferreira (2010)	Markovian-based process discovery
	Van der Aalst et al. (2005)	Linear Temporal Logic based process mining approach
Clustering/Pattern Recognition based Approaches – adopted from the data mining field	Song et al. (2009)	Trace clustering-based process mining approaches
	Bae et al. (2006) Ferreira and Thom (2012)	Pattern (recognition) based process mining approaches
	Song and Van der Aalst (2008)	Organization oriented clustering on event logs

that process mining approaches in other categories have also employed clustering mechanisms to varying degrees, including customized algorithms (e.g., trace-based α-algorithm). Section 2.2.2 discusses this area in further detail.

Within the process mining subarea of process modeling/analysis, *process model simplification* and *process model enhancement* are two key issues. In cases where the mined process models are highly complex, *process model simplification* can lead to better model comprehension and analysis. *Event log pre-processing* (referred as *pre-processing* below) prior to the actual process mining activities has been shown to be more effective in simplifying the outcome of process models (Folino et al. 2011; Mans et al. 2009; Veiga and Ferreira 2010). From a data flow perspective, Ly et al. (2012) have proposed a technique that relies on weeding out event logs based on domain specific constraints that complement embedded process knowledge in the event logs. From a control flow perspective, Bose and van der Aalst (2009) have proposed an abstraction-based pre-processing method to deal with loops in mined process models. Song et al. (2009) have applied trace clustering for pre-processing event logs. Notably, however, these approaches mainly rely on syntactic information (either control flow based or data flow based) in the event logs for the pre-processing, leaving a distinct research gap with respect to approaches utilizing semantic information in the pre-processing phase. *Process model enhancement* refers to incorporating additional information in the mined process models to reflect not only the data and control flows, but also additional knowledge (e.g., organizational rules). Annotating process models with complementary semantic information is considered an effective approach for model enhancement (Lin 2008). Semantic information derived from event pre-processing phases can be used for process model enhancement, but only limited research currently exists on this topic.

2.2 Clustering in process mining

As noted earlier, a majority of the process mining approaches make use of clustering techniques. A review of these approaches reveals different underlying algorithms and metrics, as summarized in Table 2. First, considering the purpose of these approaches and how they fit in the overall process mining process, all approaches, with the exception of trace

Table 2 Comparison of clustering-based process mining approaches

Approach	Algorithm basis	Clustering metric
Trace Clustering (Song et al. 2009)	Generic, so a variety of algorithms may be used	Generic
Disjunctive Workflow Schema (DWS) mining (Alves de Medeiros et al. 2008a)	*k*-means clustering	The frequency of an event appears in a trace
Sequence Clustering (Veiga and Ferreira 2010)	*k*-means clustering	First-order Markov chains
User behavior Profiling (Iglesias et al. 2012)	Agglomerative Hierarchical Clustering (AHC)	The frequency of a certain behavior from a particular user
Organizational Mining (Song and Van der Aalst 2008)	Agglomerative Hierarchical Clustering (AHC)	Hand-over-Work Relations

clustering (Song et al. 2009), are targeted at the actual process mining itself. In contrast, our research focuses on pre-processing of event logs to prepare for better process mining. Second, considering the algorithmic fit for the purpose of ontology learning and construction, *Agglomerative Hierarchical Clustering* (AHC) based algorithms are a better fit because they generate a hierarchical structure of events that is easily amenable for ontology development. Other clustering methods such as *k*-means based algorithms require defining the number of clusters prior to the actual clustering, which render them infeasible for pre-processing purposes. Third, organizational mining and user profiling approaches that rely on AHC-based algorithms do not utilize semantics-based metrics, which would be required if semantics from event logs are to be extracted and relied upon. Overall, existing clustering-based methods used in process mining have varied focus and do not leverage semantics during the pre-processing phase.

2.3 Process ontologies

The extant literature features some research in the area of incorporating semantics during BPM in general, and process mining in particular. Alves de Medeiros et al. (2008b) have proposed the semantic process mining framework, in which *ontologies*, *model references from ontologies*, and *ontology-based reasoners* are identified as the core elements. The major benefit of incorporating (machine-understandable) semantic information in the BPM context is to increase the automation level and to provide more meaningful and accurate process models. According to Wetzstein and Ma (2007), semantic-based approaches can provide support to all the phases in the BPM lifecycle, including (a) *process modeling* (e.g., domain-specific analysis using semantic information associated with process elements), (b) *process implementation* (e.g., semantics as a documentation mechanism for requirements, business policies and other constraints associated with process models), (c) *process execution* (e.g., semantic constraints that can be translated into execution rules that guide process model execution), and (d) *process analysis* (e.g., process mining and analytics can be used for discovering new process models through process mining, verifying existing models against process execution, and enhancing process models with new knowledge). In this context, semantic information extracted from event logs presents opportunities for developing novel process mining and analytics techniques (Alves de Medeiros et al. 2007).

Process ontologies are an essential component for enabling semantic BPM. However, in terms of ontology construction, current research primarily relies on manual specification approaches. For example, Jareevongpiboon and Janecek (2013) have proposed an ontology-based approach to enhance process models, in which the ontologies are manually specified. In another work, Ferreira and Thom (2012) have proposed ontological methods to map task traces during process execution with pre-defined patterns, and have adopted ontologies from the TOVE project in their study. A key limitation of manual specification approaches for process ontologies is that the design of the resultant ontologies is governed by the analyst's know-how of the context, rather than being grounded in process related data.

Last but not least, process reference models (APQC 2012) primarily concentrate on the control flow perspective rather than the semantic relations. As a result, their capabilities for semantic applications are limited. For example, user-centric querying and ontology-based reasoning is not possible without third-party tools (e.g., (Wang and Wu 2011)). Also, as noted by Wang and Wu (2011), integration among process reference models is a major challenge. Manual specification approaches further exacerbate the issue of ontology integration, whether dealing with ontologies at different levels of abstraction or different types of ontologies (e.g., process ontology, organizational ontology). A semi-automated ontology construction or extension approach is thus required for building lower level process ontologies as knowledge representation schemes that are based on process observations. Ontology learning, which is such an approach, refers to the process of extracting formalized conceptualizations from knowledge sources and assigning them to a hierarchical structure (Shamsfard and Abdollahzadeh Barforoush 2003). Ontology integration techniques for reconciling various process related ontologies are also desirable, given that current semantic annotation methods provide limited support in a multi-ontology environment (Grau et al. 2004; Maedche et al. 2002).

In summary, event log pre-processing techniques are needed that leverage semantic information for better alignment with the purpose of semi-automatically building, extending, and applying process ontologies.

3 A framework for event log pre-processing

Drawing on our research motivation presented in the introduction and the research gaps identified in Section 2, we stipulate the following six design requirements for event log pre-processing.

1) Process mining techniques relying solely on syntactic information result in complex and incomprehensible models, whereas using semantic information from pre-processing can aid in subsequent process mining and analytics. As such, event log pre-processing techniques that rely on semantic information embedded in process event logs are needed.
2) Lower level process ontologies that allow a fine-grained specification of events through class-subclass hierarchies

are recommended for event log pre-processing. Extant process ontologies are generic and at a higher level, although they support manual extensions to allow fine-grained specification.

3) A semi-automatic approach for building lower level process ontologies is preferred. Manual specification approaches only rely on the designer's know-how about the context and can lead to incompleteness. Data-driven ontology learning approaches leveraging voluminous event log data are promising candidates.
4) An event log aggregation approach is needed that is amenable for process ontology construction. Approaches that can result in a hierarchical structure of events should be preferred from the process ontology construction perspective.
5) Semantics extracted from event logs need to be integrated in designing a clustering approach for events. While several clustering-based approaches exist, none utilizes semantics-based metrics that can lead to better construction of lower level ontologies.
6) Ontology integration techniques are needed to reconcile different levels of ontologies (e.g., higher level and lower level process ontologies) as well as different types of ontologies (e.g., process ontology, organizational ontology).

The design requirements mentioned above have guided the design and development of our proposed framework for event log pre-processing. Figure 1 depicts the key components of the framework. The knowledge resources used and generated by these components are shown on the right side of the figure. The framework includes two key components, namely: *Event Log Aggregation* and *Ontology Learning*. The first component, *Event Log Aggregation*, concerns tasks such as disambiguating event logs (event names), normalizing event logs (event names), and aggregating events based on the semantic similarity between their names. The second component, *Ontology Learning*, utilizes the aggregated event logs as the input to generate lower-level classes in the process ontologies. Another task in the *Ontology Learning* component is the integration of process ontologies, which refers to the alignment between lower-level ontological classes learnt in the former task and the upper-level process ontologies by creating a bridge ontology between them.

Enhanced process ontologies with lower level classes thus generated can then be utilized by external process mining and

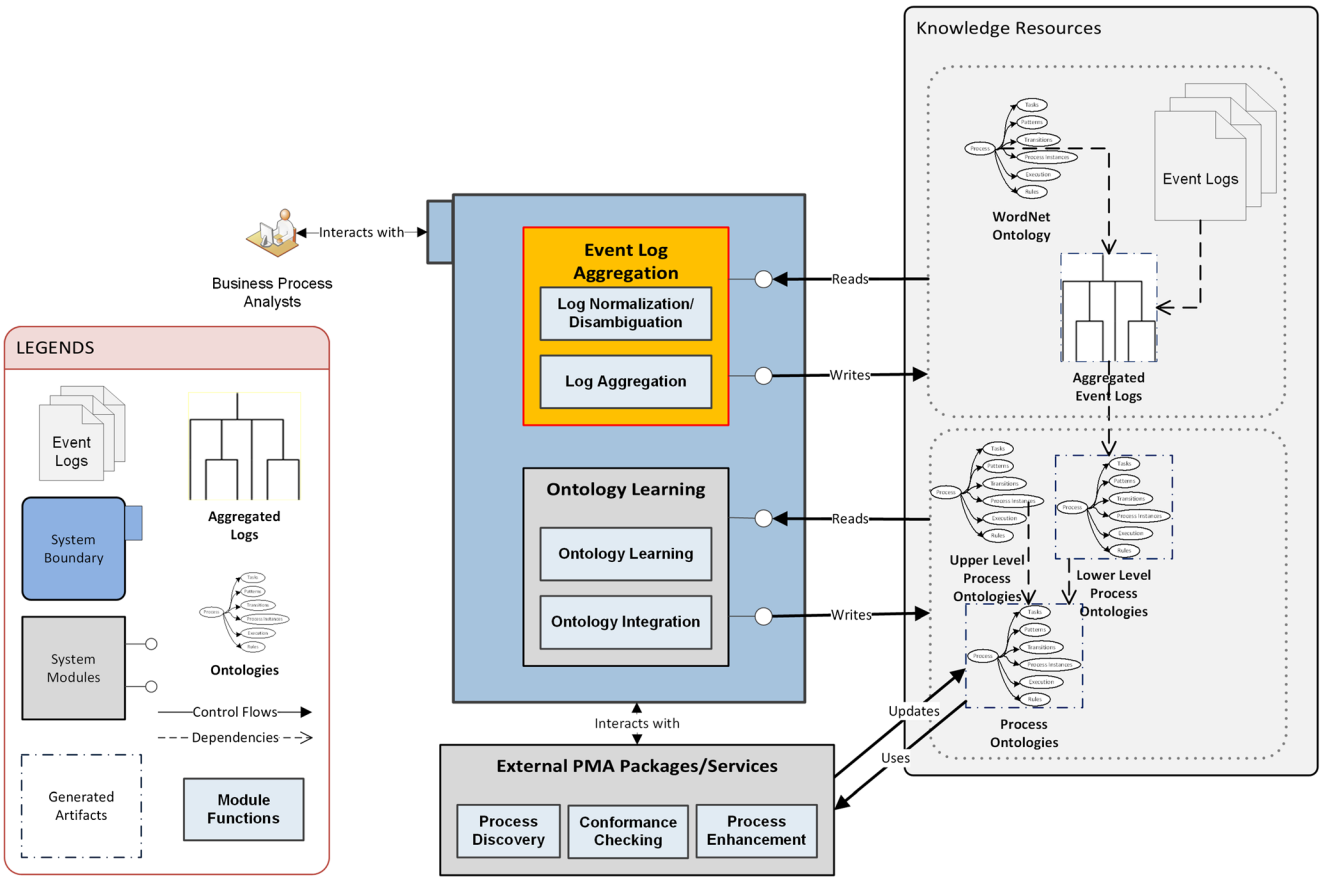


Fig. 1 Event log pre-processing framework

analytics services involving process discovery, conformance checking, and process enhancement. For example, the process ontologies generated can be used for annotating the event logs, which can then allow clustering of semantically-annotated event logs leveraging the hierarchical structure of these ontologies (Tao and Deokar 2012). In this study, we focus on the design and implementation of the *Event Log Aggregation* component of the event log pre-processing framework, highlighted in Fig. 1, which is discussed next.

4 A semantic event log aggregation approach

In this study, we propose a novel method of semantic event log aggregation through (a) normalization of event logs, and (b) their aggregation based on semantic similarity between events. The overall purpose of semantic event log aggregation is to generate a knowledge structure, representing concepts as lower-level classes, that is amenable for ontology learning. Several key steps in terms of log preparation need to be taken. This process is analogous to the critical data preparation stages prior to conducting data analysis to achieve meaningful results. Requisite aspects of event log preparation may be articulated in the form of design requirements as follows:

Event log conceptualization Given that an ontology is a formalized structure representing knowledge from a certain domain, it is imperative that the ontology classes representing key domain concepts be clearly specified. In other words, the classes in the ontology should be abstract (generalizable and formal), but not vague (accurate and concrete). Further, if ontology classes are to be constructed through automated learning from event logs, it is essential that the event names be standardized and normalized for machine understandability.

Event log aggregation Ontologies are knowledge structures capturing subsumption relationships through class hierarchy and relationships. If events mentioned in event logs are to be organized in the form of ontologies, then a mechanism to aggregate events in a hierarchical structure is required. Such a hierarchical structure can then be aligned with the process ontological classes.

Guided by the aforementioned design requirements, Fig. 2 shows the Semantic Event Log Aggregation system architecture, which includes three main modules: a) *a user interface* handling the input/output interaction with users of this system, which are primarily *business process analysts*; b) an event *log normalization* module handling parsing and normalization of event names in the original event logs; and c) an event *log aggregation* module responsible for computing semantic similarity between the event names and clustering the event logs based on the prior computational results. Design and functionality of these modules is discussed in the subsequent subsections.

Together, the two key modules, *log normalization* module and *log aggregation* module, support four distinct phases of event log aggregation process. In Phase I, the original event logs are parsed using linguistic annotations (*tokenizing*, *sentence splitting*, and *part-of speech tagging*) as well as

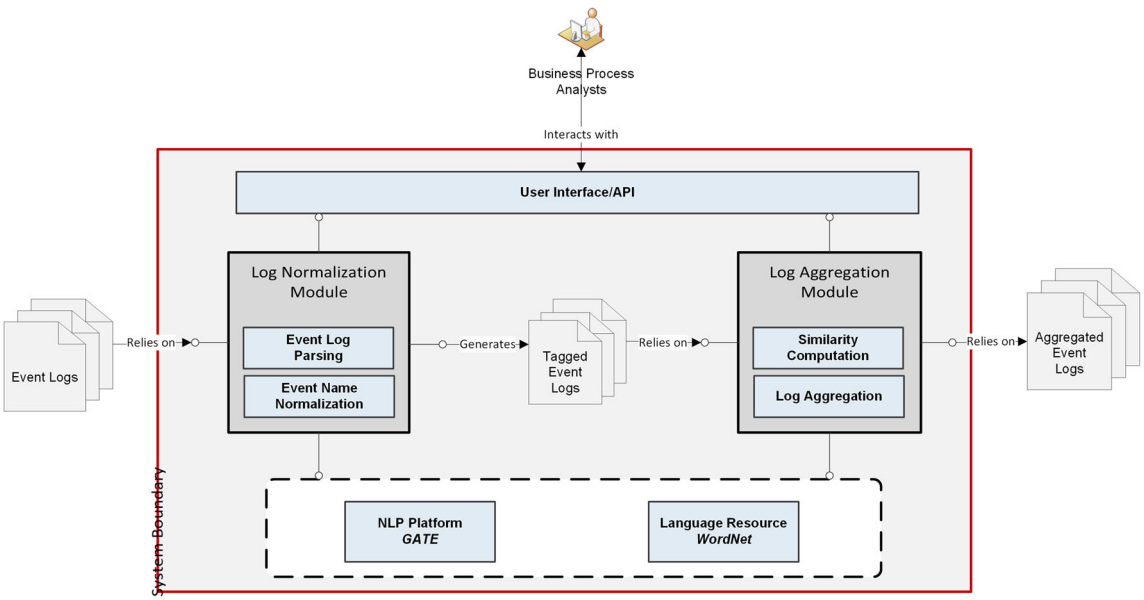


Fig. 2 Semantic event log aggregation system architecture

event log elements related annotations (event names, originators, and timestamps). Phase II is concerned with normalizing and/or disambiguating the event names extracted in the prior phase using WordNet (Princeton-University 2012) as the linguistic tool. In Phase III, a composite similarity metric is used to compute semantic similarity between normalized event names. In Phase IV, log aggregation is performed using the computed similarities as the clustering threshold parameter. Here, the original event logs are updated by adding cluster-related information to them, while maintaining compatibility with the MXML schema so that these clustered logs can be subsequently used for further mining and analytical purposes.

4.1 Log normalization module

4.1.1 Phase I – event log parsing

The event logs to be processed, shown as input in Fig. 2, are represented in a standardized format, typically MXML or an adaptation of it (Van Dongen and van der Aalst 2005). MXML is a standardized event log format widely accepted in both academia and industry (Van der Aalst et al. 2007). The MXML format has several advantages for use in process mining, including: a) maintaining a top-down structure for building process hierarchies; b) capturing key fields such as process instance IDs, event names, originators, and timestamps; and c) delineating control flow objects from data objects.

Event log parsing phase involves common natural language processing steps such as tokenizing, sentence splitting, and part of speech (POS) tagging. In addition, event log elements (e.g., *event names, originators, timestamps*) are also annotated incrementally. Finally, all annotation sets including tokens as well as event log elements are merged and made available for the following processing phases and displaying to the user. Figure 3 shows the key annotation elements of the event logs in the form of a class diagram. In this schema, MXML elements, namely, *WorkflowModelElement* (representing event name), *Originator*, and *Timestamp* are considered as *EventAttributes* alongside token information considered as *TokenAttributes*.

4.1.2 Phase II – event name normalization

The second phase in the framework normalizes the event names by disambiguating them based on the senses provided in WordNet. Toward this end, we build on advances in computational linguistics, particularly Word Sense Disambiguation techniques as follows.

Word Sense Disambiguation (WSD) refers to the process in which a certain sense of word is selected over others, given the POS and contextual information of the word (Hwang et al.

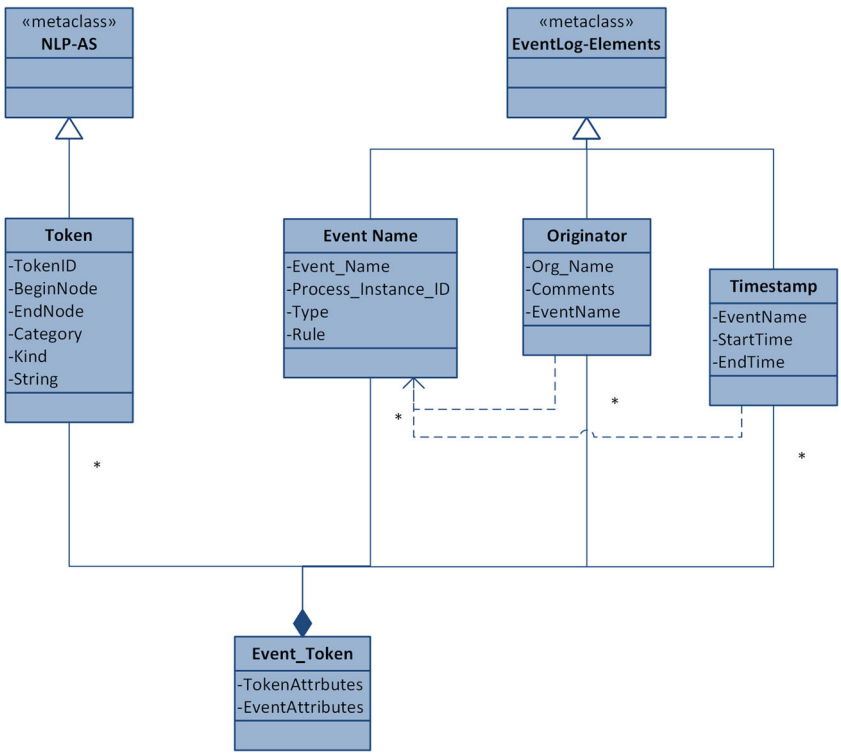


Fig. 3 Event log annotation set schema

2011; Navigli 2009). Approaches for WSD based on supervised learning require manually sense-tagged corpora for training, which make them impractical in cases such as event log aggregation. Semi-supervised learning approaches such as Yarowsky's (2000) bootstapping algorithm provide an alternative by leveraging the idea of creating clusters of words based on neighboring words. These approaches however lack in terms of accuracy when compared with supervised approaches. With the advent of lexical databases like WordNet, knowledge-based approaches that use external knowledge for disambiguation have been proposed (e.g., Patwardhan et al. (2003)). We build on this approach for disambiguating event names and extend it further for normalizing them.

Essentially, word sense is an accepted meaning of a word based on common usage. WordNet uses sets of synonyms called synsets to capture various senses of words or phrases. For instance, *check#v#1* is the first sense of the word *check* as a verb, and the WordNet gloss that describes this sense along with few examples is: (*examine so as to determine the accuracy, quality, or condition*) "*check the brakes*"; "*Check out the engine*". On the other hand, *check#v#5* is the fifth sense of *check* as a verb, with a different WordNet gloss describing it as: *stop for a moment, as if out of uncertainty or caution*. Obviously, the sense of word *check* in the phrase *check for application status* should be *check#v#1*. WSD algorithms aim to identify the appropriate word sense computationally. In event log aggregation, we consider verbs and nouns from the event names to be semantically most informative and as such focus the analysis on these parts of speech. The event normalization module derives the word sense for verbs (or verb phrases) and nouns (or noun phrases) in each unique event name. Given the appropriate word senses from WSD, the next step is to create normalized event names that consist of verb and noun pair for each event name. The underlying rationale is that verb and noun combinations can effectively capture the context of the event name. The disambiguation relation defined below is used to derive these verb-noun pairs.

Definition 1 (Disambiguation Relations in Event Names) Let E be the set of unique event names in the event log. Let w_n and w_v be nouns and verbs in an event name e, where $e \in E$ and $w_n \in \{n\}$ and $w_v \in \{v\}$. Here, $\{v\}$ and $\{n\}$ denote the finite set of verbs and nouns, respectively. The disambiguation relation $dis_n(w_n) = w_v$ denotes that w_n disambiguates w_v. Similarly, the disambiguation relation $dis_v(w_v) = w_n$ denotes that w_v disambiguates w_n.

It is worth noting that the following holds true for the disambiguation relations: $w_n(i) = dis_v(w_v(j))$ and $w_v(j) = dis_n(w_n(i))$, where $i, j \in N^+$. This implies that if a verb can be used for disambiguating a noun, then in turn that noun can be used for disambiguating the verb. The event name normalization algorithm is detailed in Fig. 10 in the Appendix and describes the logic of the implemented disambiguation relation. Essentially, the algorithm uses a two-fold search strategy. First, it searches for a match between the synset of the noun and the WordNet gloss of troponyms, synset, and hypernyms of the verb. Next, it searches for a match between the synset of the verb and the WordNet gloss of hyponyms, synset, and hypernyms of the noun. Any time a match is found, the disambiguation is established between the verb and noun in the event name. The search order is based on the rationale of starting the search from the most specific, then moving to the most general term.

Consider the following illustrative example with event name *check for application status*. The WSD computation identifies the following relevant senses: *check#v#1*, *application#n#2*, and *status#n#2*. Next, the event normalization component looks to find a verb-noun pair that represents the event name. According to the event name normalization algorithm, the gloss of synset of *check* with the sense of *check#v#1* (*gloss*(*syn*(*check#v#1*)) is *examine so as to determine accuracy, quality, or condition* and it contains the synonym of *status* with the sense of *status#n#2* (*syn*(*status#n#2*) = *condition*). In other words, a branch of search strategy A (from line 12–13) in the algorithm (Fig. 10 in the Appendix) is satisfied for the given words. Thus, the normalized form of event name *check for application status* is (*check*, *status*). We denote the normalization result as *Norm*(*check for application status*) = *check*, *status*).

4.2 Log aggregation module

4.2.1 Phase III – similarity computation

In this phase of the framework, the objective is to find similar events that can be later aggregated. In the discussion below, we first provide the rationale for the choice of semantic similarity metric and then describe its adaptation at the phrase-level for finding event similarity.

Semantic similarity is termed as a particular type of semantic relatedness that reflects the "*is-a*" relationship between two words based on the hypernymy/hyponymy links and "*a-kind-of*" relationship based on the generalization of two words based on information theory (Resnik 1995). A variety of approaches for computing semantic similarity have been proposed in the Natural Language Processing (NLP) and Information Retrieval (IR) domains. These approaches can be categorized broadly as *corpora-based* and *knowledge-structure-based/ontology-based* (Sánchez et al. 2009). *Corpora-based* metrics estimate the semantic similarity between two words/concepts based on their distributions in textual corpus. Usually, co-occurrences are observed as the basis for such estimation. *Knowledge-structure based* metrics, on the other hand, calculate the semantic similarity between two words by mapping them to the concepts in a formalized

knowledge structure (e.g., WordNet, domain ontologies) and then measuring the hierarchical relationships based on the knowledge structure. The later approach is selected in this framework given its reliance on semantic networks as well as to maintain consistency with the prior phase.

With regard to semantic similarity calculation methods, different metrics have been proposed in prior studies. The most popular metrics include Resnik's information content based measures (Resnik 1995), Lin's extended information content based measures (Lin 1998), L&C taxonomy depth based measures (Leacock and Chodorow 1998), and *WuP* (normalized related depth of concepts and their *Least Common Superconcept* (LCS)) (Wu and Palmer 1994).

The design choice of adapting the *WuP* measure in our work can be attributed to the following two reasons: First, the *WuP* measure uses relative depth that can leverage the deviation due to differing granularities between upper-level and lower-level concepts. This is particularly useful when dealing with a complex hierarchy such as WordNet. Second, the *WuP* measure is normalized to [0,1], so it is easier to compare different measures from different knowledge structures (e.g., one based on WordNet and another based on UMLS). The *WuP* measure can be computed as follows:

$$sim(w_1, w_2) = \frac{2 \times depth(LCS, root)}{depth(w_1, LCS) + depth(w_2, LCS) + 2 \times depth(LCS, root)} \quad (1)$$

In Eq. 1, for any given word pair (w_1, w_2), *LCS* refers to the *furthest* shared parent of (w_1, w_2) according to the knowledge structure, *root* refers to the root node in the knowledge structure (which is *Thing* in WordNet). The parameter *depth* is the path length between any two nodes, which is the number of intermediate nodes lying in a direct path between the two nodes. Figure 4 illustrates a computational example of the *WuP* measure of semantic similarity. In this example, $depth(w_1, LCS)=3, depth(w_2, LCS)=5$, $depth(LCS, root)=2$. Based on Eq. 1, $sim(w_1, w_2) = \frac{2\times2}{3+5+2\times2} = 0.33$

The extant metrics allow for calculating semantic similarity on *single-word* level. However, as noted earlier, the normalized event names in the event logs are in the form of verb-noun phrases. As such, no single word can be selected for accurately reflecting the meaning of the whole phrase. For instance, for the normalized event name '*check status*', neither '*check*' nor '*status*' fully represents the meaning of the event name. An approach for calculating semantic similarity at the *phrase-level* is thus required. We define a vector-based, correlated semantic similarity for this purpose.

Let $Norm(e_i) = (w_v(i), w_n(i))$ $(i=1, 2)$ denote two normalized event names, where $w_v(i)$ is the verb in $Norm(e_i)$ and $w_n(i)$ is the noun in $Norm(e_i)$. Thus, $Norm(e_1) = \{w_v(1), w_n(1)\}$, and $Norm(e_2) = \{w_v(2), w_n(2)\}$. The similarity vector between $Norm(e_1)$ and $Norm(e_2)$ can be denoted as:

$$\overrightarrow{sim}(Norm(e_1), Norm(e_2)) = (sim(w_v(1), w_v(2)), sim(w_n(1), w_n(2)))$$

The normalized phrase-based semantic similarity can be defined as the norm of the similarity vector, as follows.

Definition 2 (Normalized Phrase-based Semantic Similarity) Let $\overrightarrow{sim}(Norm(e_1), Norm(e_2))$ be the similarity vector between two normalized event names $Norm(e_1)$ and $Norm(e_2)$. Then the *Normalized Phrase Similarity* between the two normalized event names is the norm of $\overrightarrow{sim}(Norm(e_1), Norm(e_2))$, and denoted as $PSim(Norm(e_1), Norm(e_2)) \in [0, 1]$:

$$PSim(Norm(e_1), Norm(e_2)) = \sqrt{\frac{(sim(w_v(1), w_v(2))^2 + (sim(w_n(1), w_n(2))^2}{2}} \quad (2)$$

Equation 2 assumes equal importance to both the verb and the noun. In certain domains, there may be exceptions where the verb might be more important than the noun. This may be the case in event logs from the healthcare industry. For example, in an event named "*perform CT*", the verb "*perform*" is more important than the noun "*CT*". In such cases, this assumption can be relaxed and Eq. 2 can be modified accordingly.

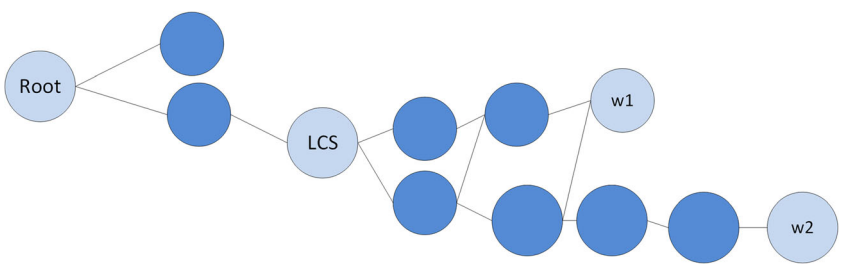


Fig. 4 *WuP* measure computation example

4.2.2 Phase IV – log aggregation

In Phase IV, a stepwise clustering approach is used on the normalized event logs in order to discover the log aggregations for subsequent ontology learning purposes. *Agglomerative Hierarchical Clustering* algorithms are better suited for concept-based learning because more information is contained in the clusters than non-hierarchical methods. This type of clustering is also widely adopted in the PMA field, given that other clustering methods require the number of clusters to be pre-specified, which is typically infeasible in the context of PMA.

In this module, we have used the group-average AHC algorithm proposed by Shepitsen et al. (2008). The normalized phrase similarity computed in phase III is used as the input along with unique normalized event names from the event logs. Essentially, the group-average AHC algorithm implementation has the following logical steps. Each event is assigned to a singleton cluster to begin with. Next, clusters of events are aggregated based on the semantic similarity between the event names. Aggregation is done in a descending manner, aggregating highly similar events first and then less similar events. At each level of clustering, the clusters are updated with additional information from the events clustered together. For example, the start time of the cluster is noted as the earliest start time of the set of events in that cluster and the end time of the cluster is noted as the latest end time of the set of events in that cluster. Similarly, other event details including the originator list and data items are aggregated at the cluster level. The updating step also ensures that the updated event log follows the standardized MXML schema that can be used for subsequent PMA services.

5 Demonstration of event log aggregation

The proposed *Log Aggregation* architecture has been instantiated in a research prototype, called *SemAgg*, and is discussed below with a use case and an implementation summary.

5.1 Representative use case scenario

The motivational scenario discussed in the introduction is a representative example use case where using the *SemAgg* system can be helpful. The sequence diagram in Fig. 5 depicts the interaction between the user interface and various system components for this example use case. It illustrates the feasibility and utility of the proposed approach, which is the key design artifact in this study. The typical interaction shown in Fig. 5 is explained below:

i) The user (typically a business analyst) uses the user interface to send an analytical request to the *Log Normalization* module by selecting the desired event log *E*.
ii) Event log *E* is parsed using the *Log Normalization* module in two substeps: first, *E* is parsed using the MXML schema; and second, it is linguistically parsed using *tokenizer*, *sentence splitter*, and *POS tagger*.
iii) Event Name Normalization is performed on the parsed event log *E*, the normalized log is then recorded and made available to the users through the user interface.
iv) The user may subsequently send a clustering request through the user interface.
v) The *Similarity Computation* phase in the *Log Aggregation* module computes the phrase-based semantic similarity between pairs of event names in the normalized event log.
vi) The results (similarities) and the normalized event log are fetched by the *Log Aggregation* module, which generates the aggregated log *E'*.
vii) The clustered logs *E'* are made available to the user for subsequent PMA.

5.2 *SemAgg* implementation

SemAgg event log aggregation system is implemented using the General Architecture for Text Engineering (GATE) platform. GATE is a comprehensive architecture supporting Natural Language Processing (NLP) tasks for developing *Information Extraction* (IE) engines (Cunningham et al. 2002). In our prototype development, we have leveraged GATE's standard language Processing Resources (PR) as well as pattern-matching grammar rule language called JAPE (*Java Annotations Patterns Engine*) for developing customized NLP tasks required for event log pre-processing. *SemAgg* has been implemented as a GATE corpus pipeline that encapsulates different phases of event log aggregation by sequencing different PRs, including custom-developed JAPE grammar rules.

We now describe the implementation details related to the *Log Normalization* module in *SemAgg*. Figure 6 shows the implementation scheme. The selected event log and WordNet serve as Language Resources (LRs) in GATE. A *WordNet* GATE plugin is used in the *SemAgg* implementation (Princeton-University 2012). The processing resources along with their corresponding sample outputs are marked as PR blocks *A* and *B* respectively.

In regards to block A, four key annotation sets (*Token, Event Name, Originator*, and *Timestamp*), as described earlier in Section 4.1.1, are categorized in two groups (*NLP-AS* and *EventLog-Elements*). The *Token* set and its features are

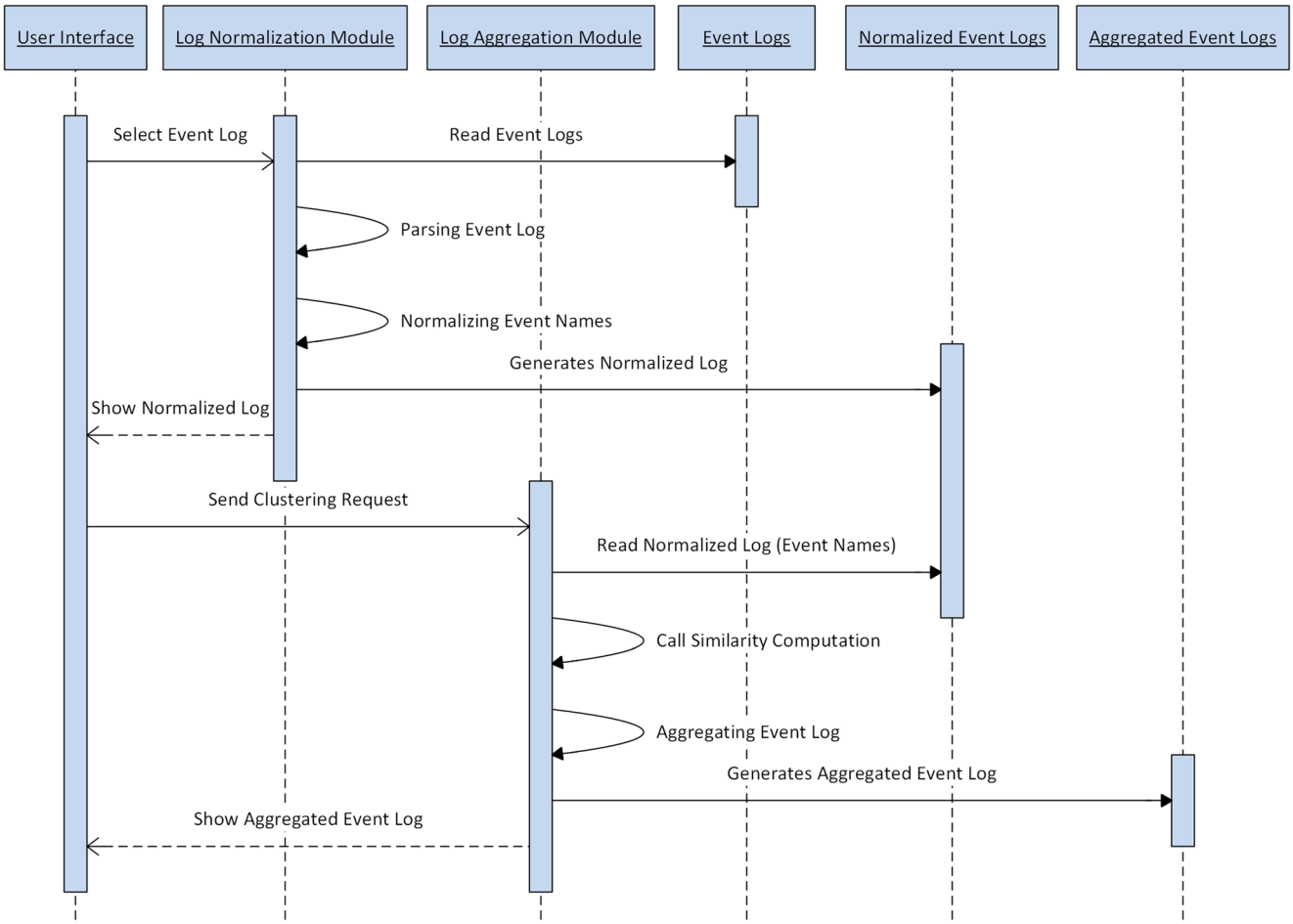


Fig. 5 Sequence diagram for an illustrative use case

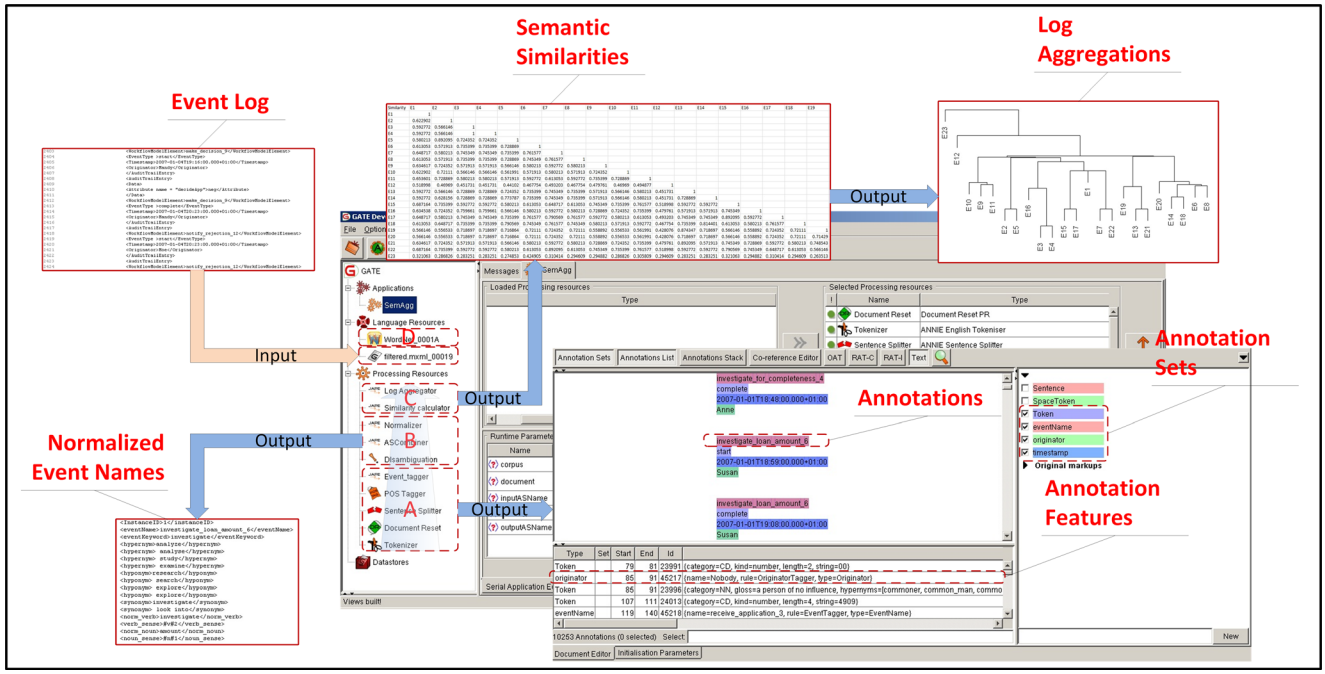


Fig. 6 *SemAgg* implementation

generated by GATE PRs, including: *ANNIE* (A Nearly-New Information Extraction system) *English Tokenizer, Sentence Splitter*, and *POS Tagger*. The other annotation sets, belonging to the *EventLog-Elements* group, are generated by the JAPE-based PR, namely *Event_Tagger*. Eventually, these annotation sets are merged together. The annotation sets, annotations, and their detailed features are shown in the lower right side in Fig. 6. The JAPE rule *Normalizer* shown in PR block *B* is used to perform event name normalization, as discussed in Section 4.1.2. The normalized event names are shown in the lower left corner of Fig. 6.

For implementing the *Log Aggregation* module in *SemAgg*, we leveraged the *WordNet::Similarity* software package for computing semantic similarity between words (Pedersen et al. 2004). In particular, we used the *ws4j* (WordNet Similarity for Java) build of *WordNet::Similarity* for developing *SemAgg*, which is a Java programming interface that encodes several semantic similarity algorithms including the *WuP* measure (Shima 2013). *ws4j* has been adapted and integrated in a JAPE-based PR, *Similarity Calculator*, for computing phrase-based semantic similarities between event names. The results of this JAPE rule are shown on the upper middle portion of Fig. 6. Another JAPE-based PR *Aggregator* then uses these calculation results to perform log aggregation on the event log. Both these JAPE-based PRs are shown in block *C* of Fig. 6. A sample aggregated event log from *SemAgg*, in the form of a dendrogram, is shown on the upper right corner in Fig. 6.

6 Experimental evaluation

We designed a computational experiment to perform an empirical evaluation for validating the efficiency and accuracy of the proposed event log aggregation approach. The experimental setup, data, and results are described next.

The experimental setup involved the following software tools: GATE developer version 7.1, 64-bit), WordNet (version 3.1, stand-alone package), the process mining tool ProM (Version 6.2), the statistical toolkits R and R Studio, and Java SE Runtime Environment (build 1.7.0_17-b02). The experimental data is based on the event logs (termed as *original log* in this discussion) from the Business Process Intelligence Challenge 2012 (Van Dongen et al. 2012). The event log has been obtained from a Dutch financial institute, and relates to its application process for personal loan/overdraft control.

The *original log* contains 262,000 events (in 36 distinct event classes) in 13,087 process instances. The following aspects were considered while data preparation for analysis. Each of the events has one of three statuses: *Start*, *Scheduled*, or *Completed*. Arguably, if all different types of statuses are considered, a larger number of events could be considered. However, the goal in this study is to develop event log aggregations depicting distinct event classes that can be used for subsequent process mining steps. In doing so, the *semantic distance* between the event classes is important as opposed to the particular status of the events. Accordingly, only events with *Completed* status have been considered for our analysis. Also, the event log contains three interleaved sub-processes related to *offer* (event name starts with "O_"), *application* (event name starts with "A_"), and *work items* (event name starts with "W_"), respectively. For the experimental purpose, we standardized the event names to follow the verb-noun phrase format. For instance, the original event name *O_ACCEPTED* is turned into *offer accepted*. Finally, we selected events above the event occurrence threshold of 0.8 to eliminate rare event occurrences. As a result of these data preparation steps, the standardized, and filtered data set (termed as *experiment log* in this discussion) used in this experiment has 164,506 events in 13,087 process instances with 23 distinct event classes. Table 3 summarizes the parameters of the experiment log and the original log, and the percentages in the brackets after the parameters indicate the coverage of experiment log. With the coverage reported, the experiment log can be said to be representative of the original log.

After parsing/tagging the *experiment log* in GATE, the JAPE rule *Normalizer* transformed the original event names into the normalized verb-noun phrases. The part of speech and the word senses are retrieved from the *experiment log* from the WordNet ontology. The original and normalized event names in *experiment log* are shown in Table 4. The normalized event names are used for computing phrase-based semantic similarity between event names.

Using the normalized event names, the JAPE-based PR - *Similarity Calculator* is then used to compute the *WuP* similarities between each pair of the event names, as discussed in Section 4.2.1. The results, in the form of a similarity matrix, are given in Fig. 11 in the Appendix. The aggregated event logs are then created using the group-average AHC Algorithm. Figure 7 illustrates the log aggregations in the form of the dendrogram, which we plotted using the *R-Studio* statistics toolkit.

For the purpose of evaluating the accuracy of the proposed approach, two independent domain experts manually defined the semantic distances between the normalized verbs and nouns in the event names, respectively. The raters used the

Table 3 Experiment event log parameters

	Original log	Experiment log
Total number of process instances	13,087	13,087 (100 %)
Total number of events	262,000	164,506 (62.8 %)
Total number of event classes	36	23 (63.9 %)
Total number of originators	69	69 (100 %)

Table 4 Original and normalized event names in experiment event log

ID	Original event name	Normalized event name	
		Verb sense	Noun sense
E1	Filling in information for application	Complete#v#5	Information#n#1
E2	Calling after sent offers	Send#v#2	Offer#n#2
E3	Application Partially submitted	Submit#v#4	Application#n#2
E4	Application submitted	Submit#v#4	Application#n#2
E5	Calling to add missing information to the application	Add#v#2	Information#n#1
E6	Assessing the application	Assess#v#1	Application#n#2
E7	Application declined	Decline#v#2	Application#n#2
E8	Application preaccepted	Accept#v#2	Application#n#2
E9	Offer sent	Send#v#2	Offer#n#2
E10	Offer created	Create#v#1	Offer#n#2
E11	Offer selected	Select#v#1	Offer#n#2
E12	Fixing incoming clue	Handle#v#1	Clue#n#2
E13	Application accepted	Accept#v#3	Application#n#2
E14	Application finalized	Finalize#v#1	Application#n#2
E15	Offer cancelled	Cancel#v#1	Offer#n#2
E16	Offer returned	Return#v#2	Offer#n#2
E17	Application cancelled	Cancel#v#1	Application#n#2
E18	Application activated	Activate#v#2	Application#n#2
E19	Application approved	Approve#v#1	Application#n#2
E20	Application registered	Archive#v#1	Application#n#2
E21	Offer accepted	Accept#v#3	Offer#n#2
E22	Offer declined	Decline#v#2	Offer#n#2
E23	Assess fraud	Assess#v#1	Fraud#n#1

following scoring schema: *0 = non-related*, *0.2 = weakly related*, *0.4 = somewhat related*, *0.6 = quite related*, *0.8 = very highly related*, and *1.0 = fully related*. The inter-rater reliability between the two researchers was 91.3 %. After reconciliation, the phrase-based similarity between the event names was calculated based on Eq. 2, and the results are shown in Fig. 12 in the Appendix. Based on the similarities, a set of log aggregations was created and used as the ground truth for further evaluation. The log aggregations based on manually rated similarities are shown in Fig. 8.

We now assess the goodness-of-fit of the event log aggregations (i.e., the dendogram in Fig. 7) in relation to the phrase-based event similarity from *SemAgg*. For event log aggregations, the cophenetic distance is the height at which two events are clustered together, as measured from a baseline indicated with distance 0. A cophenetic matrix for a dendogram is computed to record the pairwise cophenetic distances between events. Comparing the cophenetic matrix from Fig. 7 and the distance matrix (corresponding to the phrase-based similarity matrix in Fig. 11), we compute the Cophenetic Correlation Coefficient (CCC) (Fowlkes and Mallows 1983) as 0.866 that indicates a good fit. Note that the values of CCC range between −1 and 1 with values close to 1 indicating a good fit while −1 indicating low fit. This high correlation is also visually depicted through a Shepard Plot in Fig. 9 that plots the cophenetic distances and the semantic distances between the normalized event names.

CCC may also be computed to compare the resemblance between two cophenetic matrices, i.e., two dendograms (Sokal and Rohlf 1962). We computed the CCC between the *SemAgg* generated dendogram (Fig. 7) and dendogram based on manual similarity ratings (Fig. 8). This CCC value is 0.885, which indicates a relatively high resemblance between the two event log aggregations. Essentially, this validates the accuracy of the semi-automated approach embedded in *SemAgg* in relation to a domain expert-based aggregation approach.

Finally, we acknowledge some limitations of the current experiment setting. We observe from the data set that the variation among the verbs is much higher than among the nouns, i.e., there are18 unique verbs versus 4 unique nouns in 23 event names. However, in the current parameter settings, we have assigned equal weights to verb and noun phrases in the

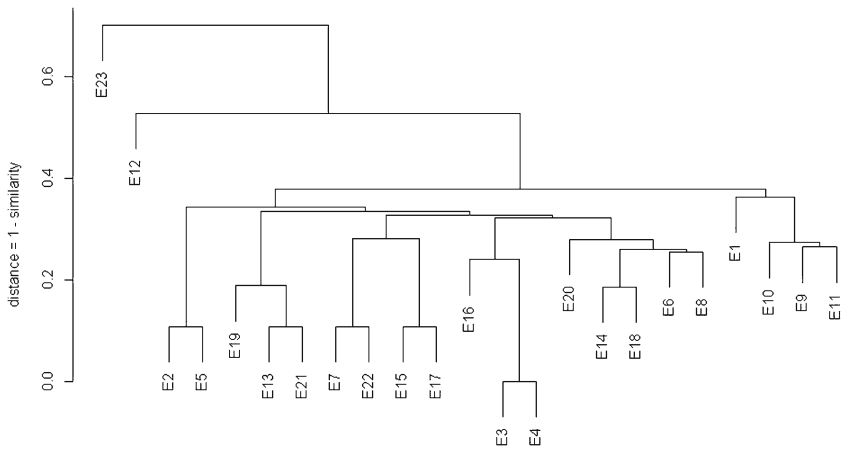


Fig. 7 Experiment event log aggregations from *SemAgg*

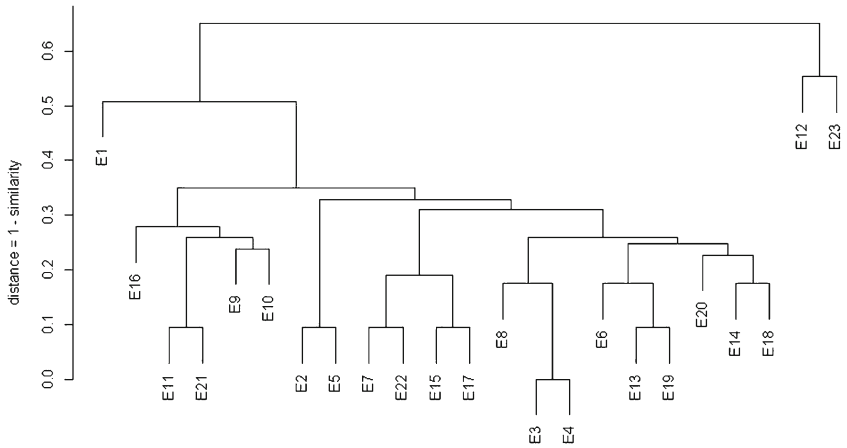


Fig. 8 Experiment event log aggregations based on manual similarity ratings

event names. We acknowledge that using a different set of weights on verbs/nouns based on domain knowledge might lead to different aggregation results that are more representative of the domain.

7 Discussion

The design requirements stated in Section 4 can be revisited in the light of the design, implementation, and evaluation of the design artifacts presented. The *SemAgg* prototype system provides an instantiation of the proposed event log aggregation architecture. First, *SemAgg* is capable of normalizing the event names in the event logs based on sense of word usage. Given a particular log aggregation, the event names and their WordNet synsets and hypernyms are potential candidates for building ontological class structure. Second, *SemAgg* supports creating a hierarchical structure based on semantic distance between the event names, which can be aligned with lower-level structure of process ontologies. Based on the experimental evaluation in Section 6, it is observed that while some of the clusters might need further manual inspection and tweaking, the log aggregation serves as a fairly accurate template for ontology learning purposes. In sum, both of the design requirements for event log aggregation are addressed by the proposed architecture and the instantiated *SemAgg* system. From a practical standpoint, event log aggregations can be used to augment process ontologies and in turn advance each of the three key PMA activities, namely process discovery, conformance checking, and process enhancement, as described below.

In process discovery activities, temporal-based approaches suffer from issues such as ad hoc events that are considered

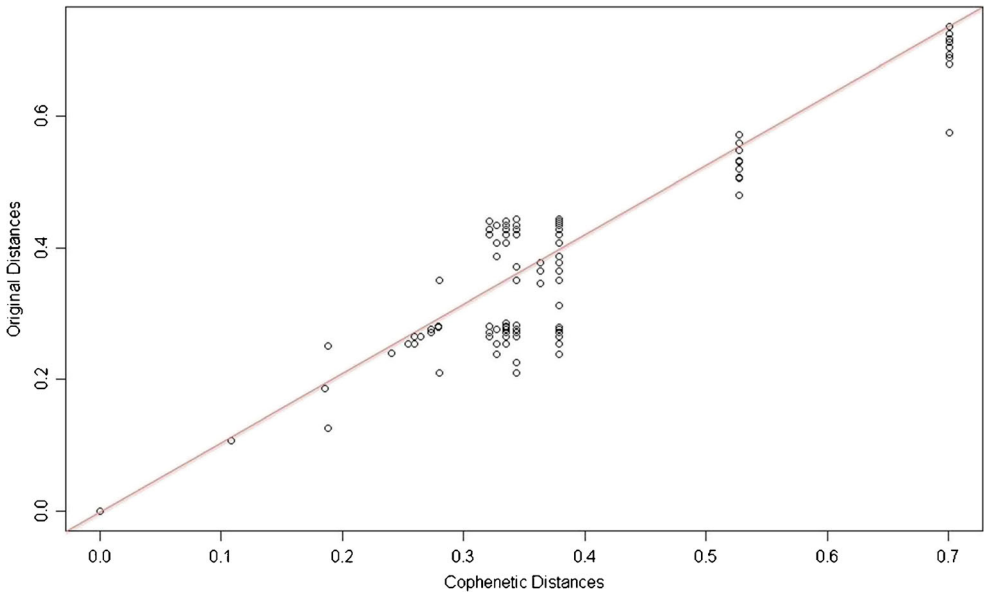


Fig. 9 Shepard Plot showing Correlations between the event log aggregations and the semantic similarities between event names

noise or disturbances. Concurrent activities further exacerbate the problem of clear separation of semantically different process structures. Process mining algorithms typically apply various measures such as clustering to counteract or alleviate these noises/disturbances, which can lead to information loss or *over-generalization* in mined process models (Alves de Medeiros et al. 2008a; Smirnov et al. 2011). The proposed meronymy-based approach can help address this issue of over-generalization when used in conjunction with existing approaches. The hierarchical abstractions captured in process ontologies generated from even log aggregations can provide knowledge structure that can enable a balanced approached between clustering solely based on temporal dependencies or solely based on contextual meanings of activities. Moreover, the different levels of abstractions in process ontologies can serve as a tuning parameter in addressing the over-generalization issue.

The design artifacts in this paper are complementary to the existing event clustering methods, particularly trace clustering approaches (Song et al. 2009) and organizational mining approaches (Song and van der Aalst 2008). It can be noted that the focus of Song et al.'s (2009) study is on providing a generalized view of trace clustering based on the temporal relations or other logical links between events, while the focus of Song and van der Aalst (2008) approach is on utilizing Hand-over-Work (*HoW*) links between the originators of events. As noted above, the semantic event log aggregation approach can be used in a complementary manner through the use of lower level, domain-specific process ontologies during clustering while conducting process mining. For example, the aggregation generated in our experiment, {{E14, E18}, {E20, {E6, E8}}}, may form a hierarchical structure with event class *application-oriented activities* that contains two sub-classes, namely *review application* and *finish application*. The event normalization approach can also facilitate clarification of vague event descriptions or false signifiers in event logs that may lead to interpretation problems for process analysts (Thomas and Fellmann 2006).

From a conformance checking perspective, it is important to validate the mined process model obtained through process discovery in relation to the intended process in the organization. Process descriptions are a common occurrence in organizations for capturing process know-how although formal process management systems may be absent. These process descriptions can be mined with text analytic techniques. Toward this end, event logs from information systems can be aggregated using the proposed approach and used in conjunction with text analytic techniques for conformance checking purposes. Another key use case for comparing process models using a semantic approach is in the context of inter-organizational workflows where different vocabularies can be expected to exist for interconnected processes (Ehrig et al. 2007).

Process enhancement is another area where semantic-based event log aggregation can be applied. In *organizational mining*, the goal is to derive information about the organizational roles and structure from the event logs to infer cross-functional dependencies. In such scenarios, process ontologies enhanced with properties related to activity owners and functional areas can help uncover the organizational aspects during process mining (Song and van der Aalst 2008). Similarly, process ontologies based on event log aggregations can be used in *decision mining* where the goal is to discover common decision choices within process execution logs. In this case, target class labels for classifying events can be defined at variety of abstraction levels based on the hierarchical structure of the ontology. Also, in case of *performance analysis*, where the objective to improve processes with respect to operational performance indicators such as time, cost, and so forth, process ontologies built upon event log aggregation can play a key role by clarifying assumptions and aligning mined process models with process specifications such as business rules. Finally, a formalized knowledge structure in the form of process ontologies can be used in conjunction with other knowledge structures related to monitoring metrics to develop symptom detection and diagnosis techniques that facilitate *process auditing* (Alves de Medeiros et al. 2007).

8 Conclusion

Abundant availability of event logs through process-oriented systems has resulted in increased attention to the process mining and analytics research within the larger business process management area. While PMA activities are focused on process discovery, process conformance checking, and process enhancement, pre-processing process observations or event logs are critical steps in achieving PMA objectives. In this research study, we have proposed a framework for event log pre-processing that departs from conventional, syntax-based approaches by specifically using a semantics-based method for event log aggregation. We also propose an event normalization technique for concept disambiguation and normalization of event log text information. This is further used in an event log aggregation approach based on phrase-based semantic similarity between normalized event names to generate a hierarchical structure of events that can be later used for lower level process ontology construction. The approach has been instantiated in a research prototype system *SemAgg* and the experiment conducted shows promising results with practical implications for advancing process mining and analytics area.

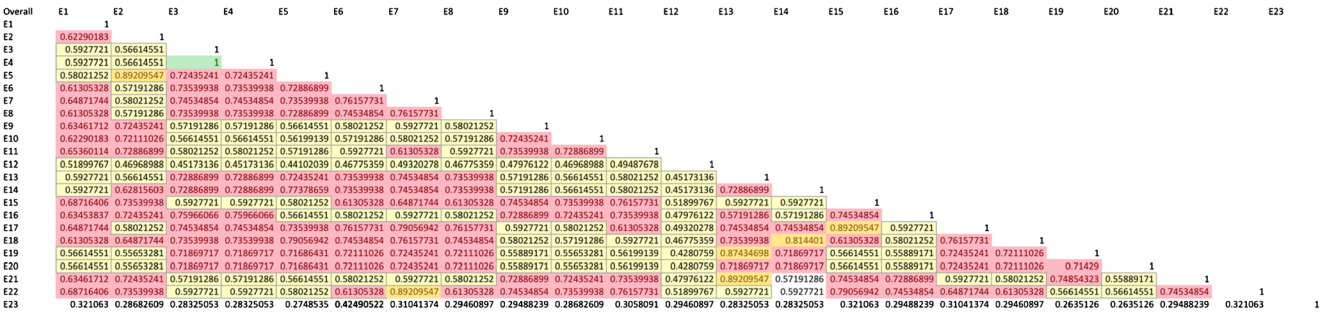
Appendix

Algorithm1: EVENT NAME NORMALIZATION

INPUT: W_V, W_N (W_V, W_N denote sets of verbs and nouns in an event name, respectively)
INPUT: $s = w_v\#v\#i$, $s' = w_n\#n\#j$, $w_v \in W_V, w_n \in W_N$
OUTPUT: (w_v, w_n) $(w_v = dis(w_n), w_n = dis(w_v), w_v \in W_V, w_n \in W_N)$

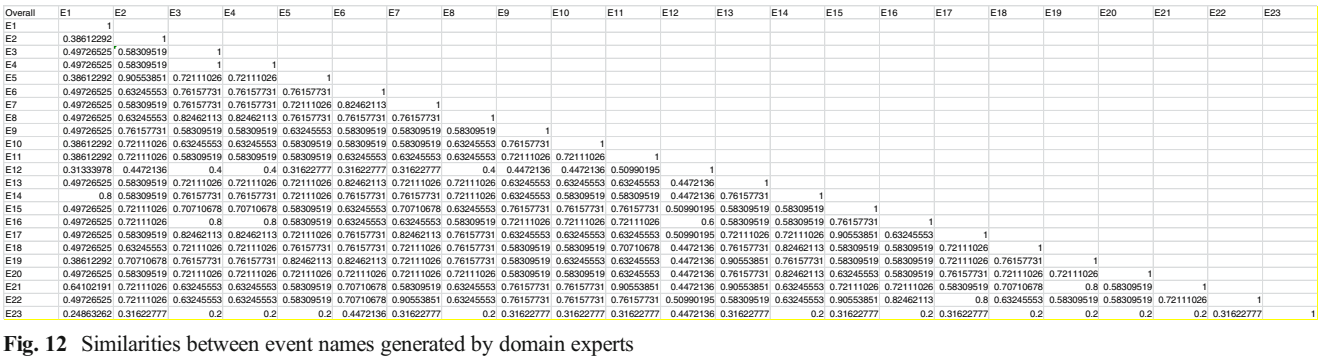
```
INITIALIZE
s = w_v#v#i, s' = w_n#n#j          // WordNet senses of w_v and w_n respectively
tropo(w_v), syn(w_v), hyper(w_v) // direct troponyms, synonyms and hypernym of w_v respectively
hypo(w_n), syn(w_n), hyper(w_n) // direct hyponyms, synonyms and hypernym of w_n respectively
PROCEDURE NORM(w_v, w_n) { // Define event name normalization procedure
   READ s, s' from WSD
   READ tropo(w_v, s), syn(w_v, s), hyper(w_v, s) from WordNet
   READ hypo(w_n, s'), syn(w_n, s'), hyper(w_n, s') from WordNet
     FOREACH (troponym in tropo(w_v,s)) { // start search strategy A
              IF (gloss(troponym) contains syn(w_n,s)) { // iterate through gloss of troponym
                   RETURN (w_v, w_n) }}
     IF (gloss(syn(w_v,s)) contains syn(w_n,s)) { // moving up to gloss of synset
                   RETURN (w_v, w_n) } // no need for iteration since synset shares same gloss
     IF (gloss(hyper(w_v,s)) contains syn(w_n,s)) { // moving up to gloss of hypernym
                   RETURN (w_v, w_n) } // end search strategy A
     FOREACH (hyponym in hypo(w_n, s')) { // start search strategy B
              IF (gloss(hyponym) contains syn(w_v,s)) { // iterate through gloss hyponym
                   RETURN (w_v, w_n) }}
     IF (gloss(syn(w_n,s')) contains syn(w_v,s)) { // moving up to gloss of synset
                   RETURN (w_v, w_n) } // no need for iteration since synset shares same gloss
     IF (gloss(hyper(w_n,s') contains syn(w_v,s)) { // moving up to gloss of hypernym
                   RETURN (w_v, w_n) } // end search strategy B
BEGIN
TRY { // try-catch block to catch if Norm succeed
     FOREACH (w_v in W_V , w_n in W_N) {
          NORM(w_v, w_n) }} // calling event name normalization procedure
CATCH (no match in NORM) {
     THROW ("event name cannot be normalized")}
//Show the results
RETURN (w_v, w_n)
END
```

Fig. 10 Event name normalization algorithm



Overall	E1	E2	E3	E4	E5	E6	E7	E8	E9	E10	E11	E12	E13	E14	E15	E16	E17	E18	E19	E20	E21	E22	E23
E1	1																						
E2	0.62290183	1																					
E3	0.5927721	0.56614551	1																				
E4	0.5927721	0.56614551	1	1																			
E5	0.58021252	0.89209547	0.72435241	0.72435241	1																		
E6	0.61305328	0.57191286	0.73539938	0.73539938	0.72886899	1																	
E7	0.64871744	0.58021252	0.74534854	0.74534854	0.73539938	0.76157731	1																
E8	0.61305328	0.57191286	0.73539938	0.73539938	0.72886899	0.74534854	0.76157731	1															
E9	0.63461712	0.72435241	0.57191286	0.57191286	0.56614551	0.58021252	0.5927721	0.58021252	1														
E10	0.62290183	0.72111026	0.56614551	0.56614551	0.56199139	0.57191286	0.58021252	0.57191286	0.72435241	1													
E11	0.65360114	0.72886899	0.58021252	0.58021252	0.57191286	0.5927721	0.61305328	0.5927721	0.73539938	0.72886899	1												
E12	0.51899767	0.46968988	0.45173136	0.45173136	0.44102039	0.46775359	0.49320278	0.46775359	0.47976122	0.46968988	0.49487678	1											
E13	0.5927721	0.56614551	0.72886899	0.72886899	0.72435241	0.73539938	0.74534854	0.73539938	0.57191286	0.56614551	0.58021252	0.45173136	1										
E14	0.5927721	0.62815603	0.72886899	0.72886899	0.77378659	0.73539938	0.74534854	0.73539938	0.57191286	0.56614551	0.58021252	0.45173136	0.72886899	1									
E15	0.68716406	0.73539938	0.5927721	0.5927721	0.58021252	0.61305328	0.64871744	0.61305328	0.74534854	0.73539938	0.76157731	0.51899767	0.5927721	0.5927721	1								
E16	0.63453837	0.72435241	0.75966066	0.75966066	0.56614551	0.58021252	0.5927721	0.58021252	0.72886899	0.72435241	0.73539938	0.47976122	0.57191286	0.57191286	0.74534854	1							
E17	0.64871744	0.58021252	0.74534854	0.74534854	0.73539938	0.76157731	0.79056942	0.76157731	0.5927721	0.58021252	0.61305328	0.49320278	0.74534854	0.74534854	0.89209547	0.5927721	1						
E18	0.61305328	0.64871744	0.73539938	0.73539938	0.73539938	0.79056942	0.74534854	0.76157731	0.74534854	0.58021252	0.57191286	0.5927721	0.46775359	0.73539938	0.814401	0.61305328	0.58021252	0.76157731	1				
E19	0.56614551	0.55653281	0.71869717	0.71869717	0.71686431	0.72111026	0.72435241	0.72111026	0.55889171	0.55653281	0.56199139	0.4280759	0.87434698	0.71869717	0.56614551	0.55889171	0.72435241	0.72111026	1				
E20	0.56614551	0.55653281	0.71869717	0.71869717	0.71686431	0.72111026	0.72435241	0.72111026	0.55889171	0.55653281	0.56199139	0.4280759	0.71869717	0.71869717	0.56614551	0.55889171	0.72435241	0.72111026	0.71429	1			
E21	0.63461712	0.72435241	0.57191286	0.57191286	0.56614551	0.58021252	0.5927721	0.58021252	0.72886899	0.72435241	0.73539938	0.47976122	0.89209547	0.57191286	0.74534854	0.72886899	0.5927721	0.58021252	0.74854323	0.55889171	1		
E22	0.68716406	0.73539938	0.5927721	0.5927721	0.58021252	0.61305328	0.89209547	0.61305328	0.74534854	0.73539938	0.76157731	0.51899767	0.5927721	0.5927721	0.79056942	0.74534854	0.64871744	0.61305328	0.56614551	0.56614551	0.74534854	1	
E23	0.321063	0.28682609	0.28325053	0.28325053	0.2748535	0.42490522	0.31041374	0.29460897	0.29488239	0.28682609	0.3058091	0.29460897	0.28325053	0.28325053	0.321063	0.29488239	0.31041374	0.29460897	0.2635126	0.2635126	0.29488239	0.321063	1

Fig. 11 Similarities between event names from the experiment output



Overall	E1	E2	E3	E4	E5	E6	E7	E8	E9	E10	E11	E12	E13	E14	E15	E16	E17	E18	E19	E20	E21	E22	E23
E1	1																						
E2	0.38612292	1																					
E3	0.49726525	0.58309519	1																				
E4	0.49726525	0.58309519	1	1																			
E5	0.38612292	0.90553851	0.72111026	0.72111026	1																		
E6	0.49726525	0.63245553	0.76157731	0.76157731	0.76157731	1																	
E7	0.49726525	0.58309519	0.76157731	0.76157731	0.72111026	0.82462113	1																
E8	0.49726525	0.63245553	0.82462113	0.82462113	0.76157731	0.76157731	0.76157731	1															
E9	0.49726525	0.76157731	0.58309519	0.58309519	0.63245553	0.58309519	0.58309519	0.58309519	1														
E10	0.38612292	0.72111026	0.63245553	0.63245553	0.58309519	0.58309519	0.58309519	0.63245553	0.76157731	1													
E11	0.38612292	0.72111026	0.58309519	0.58309519	0.58309519	0.63245553	0.63245553	0.63245553	0.72111026	0.72111026	1												
E12	0.31333978	0.4472136	0.4	0.4	0.31622777	0.31622777	0.31622777	0.4	0.4472136	0.4472136	0.50990195	1											
E13	0.49726525	0.58309519	0.72111026	0.72111026	0.72111026	0.82462113	0.72111026	0.72111026	0.63245553	0.63245553	0.63245553	0.4472136	1										
E14	0.8	0.58309519	0.76157731	0.76157731	0.72111026	0.76157731	0.76157731	0.72111026	0.63245553	0.58309519	0.58309519	0.4472136	0.76157731	1									
E15	0.49726525	0.72111026	0.70710678	0.70710678	0.58309519	0.63245553	0.70710678	0.63245553	0.76157731	0.76157731	0.76157731	0.50990195	0.58309519	0.58309519	1								
E16	0.49726525	0.72111026	0.8	0.8	0.58309519	0.63245553	0.63245553	0.58309519	0.72111026	0.72111026	0.72111026	0.6	0.58309519	0.58309519	0.76157731	1							
E17	0.49726525	0.58309519	0.82462113	0.82462113	0.72111026	0.76157731	0.82462113	0.76157731	0.63245553	0.63245553	0.63245553	0.50990195	0.72111026	0.72111026	0.90553851	0.63245553	1						
E18	0.49726525	0.63245553	0.72111026	0.72111026	0.76157731	0.76157731	0.72111026	0.76157731	0.58309519	0.58309519	0.70710678	0.4472136	0.76157731	0.82462113	0.58309519	0.58309519	0.72111026	1					
E19	0.38612292	0.70710678	0.76157731	0.76157731	0.82462113	0.82462113	0.72111026	0.76157731	0.58309519	0.63245553	0.63245553	0.4472136	0.90553851	0.76157731	0.58309519	0.58309519	0.72111026	0.76157731	1				
E20	0.49726525	0.58309519	0.72111026	0.72111026	0.72111026	0.72111026	0.72111026	0.72111026	0.58309519	0.58309519	0.63245553	0.4472136	0.76157731	0.82462113	0.63245553	0.58309519	0.76157731	0.72111026	0.72111026	1			
E21	0.64102191	0.72111026	0.63245553	0.63245553	0.58309519	0.70710678	0.58309519	0.63245553	0.76157731	0.76157731	0.90553851	0.4472136	0.90553851	0.63245553	0.72111026	0.72111026	0.58309519	0.70710678	0.8	0.58309519	1		
E22	0.49726525	0.72111026	0.63245553	0.63245553	0.58309519	0.70710678	0.90553851	0.63245553	0.76157731	0.76157731	0.76157731	0.50990195	0.58309519	0.63245553	0.90553851	0.82462113	0.8	0.63245553	0.58309519	0.58309519	0.72111026	1	
E23	0.24863262	0.31622777	0.2	0.2	0.2	0.4472136	0.31622777	0.2	0.31622777	0.31622777	0.31622777	0.4472136	0.31622777	0.2	0.31622777	0.2	0.31622777	0.2	0.2	0.2	0.2	0.31622777	1

Fig. 12 Similarities between event names generated by domain experts

References

Alves de Medeiros, A. K., van Dongen, B. F., van der Aalst, W. M. P., & Weijters, A. J. M. M. (2004). Process mining for ubiquitous mobile systems: An overview and a concrete algorithm. In L. Baresi, S. Dustdar, H. Gall, & M. Matera (Eds.), *Ubiquitous Mobile Information and Collaboration Systems (UMICS 2004)* (Vol. 3272, pp. 151–165). Berlin: Springer.

Alves de Medeiros, A. K., Weijters, A. J. M. M., & van der Aalst, W. M. P. (2006). Genetic process mining: an experimental evaluation. *Data Mining and Knowledge Discovery, 14*(2), 245–304.

Alves de Medeiros, A. K., Pedrinaci, C., van der Aalst, W. M. P., Domingue, J., Song, M., Rozinat, A., Norton, B., Cabral, L. (2007). An outlook on semantic business process and monitoring. In *Proceedings of the 2007 OTM Confederated international conference on the move to meaningful internet systems - Volume Part II* (1244–1255). Berlin: Springer-Verlag.

Alves de Medeiros, A. K., Guzzo, A., Greco, G., & van der Aalst, W. M. P. (2008a). Process mining based on clustering: A quest for precision. In *Business process management workshops lecture notes in Computer Science*, (Vol. 4928, pp. 17–29. Berlin: Springer.

Alves de Medeiros, A. K., Karla, A., van der Aalst, W. M. P., Pedrinaci, C., & Alves de Medeiros, A. K. (2008b). Semantic process mining tools: Core building blocks. In W. Golden, T. Acton, K. Conboy, H. van der Heijden, & V. Tuunainen (Eds.), *Proceedings of the 16th European Conference on Information Systems (ECIS'08)* (pp. 1953–1964). Ireland: Galway.

APQC. (2012). *APQC's Process Classification Framework, Version 6.0.0-en-XI*.

Bae, J., Caverlee, J., Liu, L., & Yan, H. (2006). Process mining by measuring process block similarity. In *Business Process Management Workshops, Lecture Notes in Computer Science* (Vol. 4103, pp. 141–152). Berlin: Springer-Verlag.

Bose, R., & van der Aalst, W. M. P. (2009). Abstractions in process mining: A taxonomy of patterns. In *Business process management lecture notes in Computer Science* (Vol. 5701, pp. 159–175). Berlin: Springer.

Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V. (2002). GATE: An architecture for development of robust HLT applications. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)* (pp. 168–175).

Edgington, T. M., Raghu, T. S., & Vinze, A. S. (2010). Using process mining to identify coordination patterns in IT service management. *Decision Support Systems, 49*(2), 175–186. doi:10.1016/j.dss.2010.02.003.

Ehrig, M., Koschmider, A., & Oberweis, A. (2007). Measuring similarity between semantic business process models. In *Proceedings of the 4th Asia-Pacific conference on Comceptual modelling - Volume 67 (APCCM '07)* (pp. 71–80). Australia: Darlinghurst.

Ferreira, D. R., & Thom, L. H. (2012). A semantic approach to the discovery of workflow activity patterns in event logs. *International Journal of Business Process Integration and Management, 6*(1), 4–17.

Folino, F., Greco, G., Guzzo, A., & Pontieri, L. (2011). Mining usage scenarios in business processes: outlier-aware discovery and run-time prediction. *Data & Knowledge Engineering, 70*(12), 1005–1029. doi:10.1016/j.datak.2011.07.002.

Fowlkes, E. B., & Mallows, C. L. (1983). A method for comparing two hierarchical clusterings. *Journal of the American Statistical Association, 78*(383), 553–569.

Grau, B. C., Parsia, B., Sirin, E. (2004). Working with Multiple Ontologies on the Semantic Web. In *The Semantic Web – ISWC 2004, Lecture Notes in Computer* Science (Vol. 3298, pp. 620–634)

Greco, G., Guzzo, A., Pontieri, L., & Sacca, D. (2006). Discovering expressive process models by clustering log traces. *IEEE Transactions on Knowledge and Data Engineering, 18*(8), 1010–1027.

Günther, C. W. & van der Aalst, W. M. P. (2007). Fuzzy mining – adaptive process simplification based on multi-perspective metrics. In *Business process management lecture notes in Computer Science* (Vol. 4714, pp. 328–343). Berlin: Springer.

Hwang, M., Choi, C., & Kim, P. (2011). Automatic enrichment of semantic relation network and its application to word sense disambiguation. *IEEE Transactions on Knowledge and Data Engineering, 23*(6), 845–858.

IEEE Task Force on Process Mining. (2011). *Process mining manifesto*.

Iglesias, J. A., Angelov, P., Ledezma, A., & Sanchis, A. (2012). Creating evolving user behavior profiles automatically. *IEEE Transactions on Knowledge and Data Engineering, 24*(5), 854–867.

Jareevongpiboon, W., & Janecek, P. (2013). Ontological approach to enhance results of business process mining and analysis. *Business Process Management Journal, 19*(3), 459–476. doi:10.1108/14637151311319905.

Leacock, C., & Chodorow, M. (1998). Combining local context and WordNet similarity for word sense identification. In *WordNet: An electronic lexical database* (pp. 265–283). MIT press.

Lin, D. (1998). An information-theoretic definition of similarity. In *Proceeding ICML'98 Proceedings of the Fifteenth International Conference on Machine Learning* (pp. 296 – 304).

Lin, Y. (2008). *Semantic annotation for process models: facilitating process knowledge management via semantic interoperability. Department of computer and information science*. Trondheim: Norwegian University of Science and Technology.

Ly, L. T., Indiono, C., Mangler, J., Rinderle-Ma, S. (2012). Data transformation and semantic log purging for process mining. In *Advanced Information Systems Engineering, Lecture Notes in Computer Science* (Vol. 7328, pp. 238–253).

Maedche, A., Motik, B., Stojanovic, L., Studer, R., Volz, R. (2002). Managing multiple ontologies and ontology evolution in ontologging. In *Intelligent Information Processing: IFIP — The International Federation for Information Processing* (Vol. 93, pp. 51–63).

Malone, T. W., Crowston, K., & Herman, G. A. (2003). *Organizing business knowledge: The MIT process handbook*. Cambridge: The MIT Press.

Mans, R. S., Schonenberg, M. H., Song, M., & van der Aalst, W. M. P. (2009). Application of process mining in healthcare—a case study in a dutch hospital. *Biomedical Engineering Systems and Technologies Communications in Computer and Information Science, 25*, 425–438.

Navigli, R. (2009). Word sense disambiguation: a survey. *ACM Computing Surveys, 41*(2), 1–69. doi:10.1145/1459352.1459355.

Patwardhan, S., Banerjee, S., Pedersen, T. (2003). Using measures of semantic relatedness for word sense disambiguation. In *Proceedings of the Fourth International Conference on Intelligent Text Processing and Computational Linguistics* (pp. 241–257). Mexico City, Mexico.

Pedersen, T., Patwardhan, S., Michelizzi, J. (2004). WordNet: Similarity - measuring the relatedness of concepts. In *Proceeding HLT-NAACL–Demonstrations'04 Demonstration Papers at HLT-NAACL 2004* (pp. 38–41).

Princeton-University. (2012). About WordNet. Retrieved from http://wordnet.princeton.edu/.

Resnik, P. (1995). Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence* (Vol. 1).

Sánchez, D., Batet, M., Valls, A., & Gibert, K. (2009). Ontology-driven web-based semantic similarity. *Journal of Intelligent Information Systems, 35*(3), 383–413. doi:10.1007/s10844-009-0103-x.

Shamsfard, M., & Abdollahzadeh Barforoush, A. (2003). The state of the art in ontology learning: a framework for comparison. *The*

Knowledge Engineering Review, 18(4), 293–316. doi:10.1017/S0269888903000687.

Shepitsen, A., Gemmell, J., Mobasher, B., & Burke, R. (2008). Personalized recommendation in social tagging systems using hierarchical clustering. *Proceedings of the 2008 ACM conference on Recommender systems - RecSys'08*, 259. doi:10.1145/1454008.1454048.

Shima, H. (2013). WS4J. Retrieved from https://code.google.com/p/ws4j/.

Smirnov, S., Reijers, H. A., & Weske, M. (2011). From fine-grained to abstract process models : a semantic approach. *Information Systems, 37*(8), 784–797.

Sokal, R. R., & Rohlf, F. J. (1962). The comparison of dendrograms by objective methods. *Taxon, 11*(2), 33–40.

Song, M., & van der Aalst, W. M. P. (2008). Towards comprehensive support for organizational mining. *Decision Support Systems, 46*(1), 300–317. doi:10.1016/j.dss.2008.07.002.

Song, M., Günther, C., & van der Aalst, W. (2009). Trace clustering in process mining. *Business Process Management Workshops Lecture Notes in Business Information Processing, 17*(2), 109–120.

Tao, J., & Deokar, A. V. (2012). Creating semantic activity profiles using semantically-annotated event logs. In *Proceedings of the 2012 SIGBPS Workshop on Business Processes and Services (SIGBPS'12)* (pp. 136–140).

Thomas, O., & Fellmann, M. (2006). Semantic event-driven process chains. In *Proceedings of the Workshop on Semantics for Business Process Management (SBPM'06), held at the 3rd European Semantic Web Conference (ESWC 2006)*. Budva, Montenegro.

Tiwari, A., Turner, C. J., & Majeed, B. (2008). A review of business process mining: state-of-the-art and future trends. *Business Process Management Journal, 14*(1), 5–22. doi:10.1108/14637150810849373.

Van der Aalst, W. M. P. (2008). Decision support based on process mining. In F. Burstein & C. W. Holsapple (Eds.), *Handbook on decision support systems*. Berlin: Springer.

Van der Aalst, W. M. P., & Weijters, A. J. M. M. (2004). Process mining: a research agenda. *Computers in Industry, 53*(3), 231–244. doi:10.1016/j.compind.2003.10.001.

Van der Aalst, W. M. P., de Beer, H. T., van Dongen, B. F. (2005). Process mining and verification of properties: An approach based on temporal logic. In *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE, Pt 1, Proceedings* (Vol. 3760, pp. 130–147). Berlin: Springer-Verlag Berlin.

Van der Aalst, W. M. P., Reijers, H. A., Weijters, A. J. M., van Dongen, M., de Medeiros, B. F., Song, A. K. A. M., & Verbeek, H. M. W. (2007). Business process mining: an industrial application. *Information Systems, 32*(5), 713–732. doi:10.1016/j.is.2006.05.003.

Van Dongen, B. F., & van der Aalst, W. M. P. (2004). EMiT: A process mining tool. In *Applications and Theory of Petri Nets 2004, Proceedings* (Vol. 3099, pp. 454–463). Berlin: Springer-Verlag Berlin.

Van Dongen, B. F., & van der Aalst, W. M. P. (2005). A meta model for process mining data. In J. Casto & E. Teniente (Eds.), *Proceedings of the open interop workshop on enterprise modelling and ontologies for interoperability (EMOI-INTEROP "05), co-located with CAiSE"05 conference* (Vol. 5, pp. 309–320). Porto: FEUP.

Van Dongen, B., Ferreira, D. R., & Weber, B. (2011). *Business Processing Intelligence Challenge (BPIC)*. doi:10.4121/uuid:d9769f3d-0ab0-4fb8-803b-0d1120ffcf54.

Van Dongen, B., Ferreira, D. R., Weber, B. (2012). Business Processing Intelligence Challenge (BPIC) 2012. Retrieved from http://www.win.tue.nl/bpi2012/doku.php?id=challenge.

Veiga, G. M., & Ferreira, D. R. (2010). Understanding spaghetti models with sequence clustering for ProM. *Business Process Management Workshops Lecture Notes in Business Information Processing, 43*, 92–103.

Wang, H. J., & Wu, H. (2011). Supporting process design for e-business via an integrated process repository. *Information Technology and Management, 12*(2), 97–109. doi:10.1007/s10799-010-0076-z.

Weber, P., Bordbar, B., & Tiňo, P. (2013). A framework for the analysis of process mining algorithms. *Systems IEEE Transactions on Systems Man and Cybernetics, 43*(2), 303–317.

Weijters, A. J. M. M., van der Aalst, W. M. P., Alves de Medeiros, A. K. (2006). *Process Mining with the Heuristics Miner-algorithm*. Eindhoven.

Wetzstein, B., & Ma, Z. (2007). Semantic business process management: A lifecycle based requirements analysis. In *Workshop on Semantic Business Process Lifecycle Management* (pp. 1–11).

Wu, Z., & Palmer, M. (1994). Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics (ACL'94)* (pp. 133–138).

Yarowsky, D. (2000). Hierarchical decision lists for word sense disambiguation. *Computers and the Humanities, 34*(1/2), 179–186. doi:10.1023/A:1002674829964.

Amit V. Deokar is an Assistant Professor of Management Information Systems in the Sam and Irene Black School of Business at Pennsylvania State University, Erie. His recent research interests are in decision support and predictive analytics, business process management, and collaboration processes and technologies. He has published several conference publications, journal articles, and book chapters in these areas. He holds a BE in Mechanical Engineering from V.J. Technological Institute, Mumbai, a MS in Industrial Engineering from the University of Arizona, and a PhD in Management Information Systems from the University of Arizona. He is a member of AIS, ACM, and AAAI, and is currently serving as the Program Chair for the AIS SIG on Decision Support and Analytics (SIGDSA).

Jie Tao is an Assistant Professor of Information Systems and Operations Management in the Dolan School of Business at Fairfield University. He holds a doctoral degree in Information Systems from Dakota State University. Dr. Tao's recent research interests include business process management and business analytics. He has published several journal articles and has presented his work at premier academic conferences including AMCIS, HICSS, ICIS, and WITS. Dr. Tao is also an active member of INFORMS, Decision Support Institute (DSI), and Association for Information Systems (AIS). He is also a member of the AIS technology committee, and received the AIS ATLAS award in 2013 for his service contributions.