LEKG: A System for Constructing Knowledge Graphs from Log Extraction

Fangrong Wang[1], Alan Bundy[1], Xue Li[1], Ruiqi Zhu[1], Kwabena Nuamah[1], Lei Xu[2], Stefano Mauceri[2], Jeff Z. Pan[1*]
[1]School of Informatics, the University of Edinburgh, UK; [2]Huawei Ireland Research Centre, Ireland

ABSTRACT

Logs record system events and status, which help developers and system administrators diagnose run time errors, monitor running status and mine operation patterns [13, 23]. However, logs are complex and weakly linked, making it difficult to diagnose the causes of failures. While recent studies on log knowledge extraction focus on lifting entities from log messages for enriching a background knowledge graph (BKG), they do not involve knowledge reasoning for inferring implicit relations nor guarantee that the knowledge learned from log streams is consistent with the background knowledge. In this preliminary research paper, we present a log extraction approach to log knowledge graph (KG) construction. It includes a novel strategy that utilizes inference rules from a background knowledge graph to learn new triples and validate triples. Also, it implements a local to global strategy to perform reasoning on temporary log instance graphs (LIGs) then on the extended BKG, which significantly reduces query space. Finally we demonstrate the applicability of this approach by a use case in the context of root cause analysis.

KEYWORDS

knowledge graph; relation linking; log analysis; root cause analysis

ACM Reference Format:
Fangrong Wang[1], Alan Bundy[1], Xue Li[1], Ruiqi Zhu[1], Kwabena Nuamah[1], Lei Xu[2], Stefano Mauceri[2], Jeff Z. Pan[1]. 2021. LEKG: A System for Constructing Knowledge Graphs from Log Extraction. In *The 10th International Joint Conference on Knowledge Graphs (IJCKG'21), December 6–8, 2021, Virtual Event, Thailand*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3502223.3502250

1 INTRODUCTION

Logs are a vital source of information for monitoring software systems' running status and diagnosing faults. Traditional methods of fault diagnosis through logs requires huge manual work from experts, which is not feasible for modern large-scale logs. Log analysis is a technique of deriving knowledge from log files [41]. It has been applied to a variety of applications such as anomaly detection [6, 22, 40], intrusion detection [5, 8, 15, 33], and root cause analysis [2, 19, 31, 42, 44].

Although logs contain valuable information, deriving knowledge from logs is different from classic information extraction against text: logs are weakly structured, vary in different formats and do not follow natural language grammar. Typical log analysis tasks start from template-based detection to parse useful information from logs [13]; then use data-mining algorithms to analyze or summarize patterns from events. However, these approaches do not represent events and their connections in formal knowledge representation nor do they consider other perspectives on data. Some recent studies have facilitated the integration of log information by linking knowledge extracted from logs [8] and aligning them with a background knowledge graph (BKG) [25, 26]. These approaches open up opportunities for downstream research and practice based on formal represented knowledge. For instance, it is possible to query the log KG and contextualize local event information. These approaches, however, only use the BKG as a target graph to align and merge new entities from logs, and do not leverage existing knowledge to infer implicit knowledge nor guarantee the quality of extracted knowledge.

Background knowledge, such as internal expert knowledge, architectural information and external knowledge [7, 8, 15], exists in real enterprise scenarios. For example, BKGs can be obtained by information extraction or exported from other data sources. Unlike existing works [7, 8, 15, 43] that only use BKGs to align and map new entities, our view of the background knowledge contains not only entities but also logical rules that describe the logical connections and constraints between modules. In our work, new triples are learned from the KG by rule inference and validated by constraints from background knowledge. Only valid triples will be included in the enriched KG.

To this end, we developed a system (LEKG) that extracts and learns knowledge from logs. To keep the reasoning in manageable query space, we designed a local to global strategy to infer triples in Section 3. We first construct temporary LIGs based on groups of logs having specific features, then perform reasoning against each LIG for inferring relations between log instance entities. After merging triples from LIGs into the BKG, we perform reasoning on this global BKG to infer implicit relations. Then we apply constraint-based triple validation on new triples. Finally, we describe a use case that uses the LEKG for root cause analysis in Section 4. The contributions of this work are summarized as below: (i) A novel strategy to extract knowledge from large volumes of logs; (ii) A system, called LEKG, to integrate extraction, rule-based relation linking, constraint-based validation; (iii) A use case that applies LEKG to root cause analysis.

*Correspondence author: Jeff Z. Pan, https://knowledge-representation.org/j.z.pan/

IJCKG'21, December 6–8, 2021, Virtual Event, Thailand

ACM ISBN 978-1-4503-9565-6/21/12…$15.00
https://doi.org/10.1145/3502223.3502250

2 RELATED WORK

Typical log analysis tasks are designed to solve specific problems and do not learn general knowledge that has an insight into how software systems operate. Among these approaches, log representation in graphs has attracted recent research interest. Various graph-based approaches have been proposed in the literature, covering applications such as query log analysis [9], cybersecurity [3], anomaly detection [5, 32], root cause analysis [2], business process analysis [4]. These approaches focus on resolving problems by graph-theoretical methods, and do not aim to extract general knowledge from log data. Recent studies [7, 8] aimed at semantic lifting of general log data. These works involve not only a huge number of logs but also existing background knowledge. They rely on log parsing tools [43] to obtain event templates and then merge the lifted entities to a BKG by mapping and aligning based on similar vocabulary and common identifiers. Although these studies have similar purpose to us, they do not use the BKG to infer implicit relations nor validate the extracted knowledge. Kiesling et al. [15] builds a system to integrate newly available structured data from public sources into a cybersecurity KG, which involves acquisition, extraction, lifting, linking, and validation steps. Its validation is to make sure the necessary properties lifted from logs are included for each generated individual. However, the approach only checks if a reference entity exist, rather than checking logical or semantic consistencies of the extracted knowledge.

3 METHODOLOGY

3.1 Problem Definition

The combination of background knowledge and a huge number of logs is a typical scenario in information technology companies. Hence we formulate our research problem as follows.

Definition 3.1 (Aim of Research). Given a BKG $G_{\mathbb{B}}$, a set of logs in their original format $\mathbb{L}$ and a log theory $\tau = R_{positive} \cup R_{negative}$, where $R_{positive}$ are positive rules that infer new triples and $R_{negative}$ are negative rules that are constraints for validation, the task is to update BKG $\mathbb{G}$ by extracting knowledge from $\mathbb{L}$ to enrich $G_{\mathbb{B}}$.

3.2 Solution

The overall structure of our system is illustrated in Figure 1.

We construct a BKG by applying classic information extraction techniques, such as semantic parsing of technical manuals to produce a set of high-level conceptual triples. The automated log parsing converts free text to structured information. However, the parsed items usually only have vague categories; entities and events may hide in the body of parsed items. Hence, we combine Named Entity Recognition (NER)[1] and templates to identify entities from log messages. By training a domain-specific NER model, we identify and label the entities based on their semantic context. The entity labels are assigned to entities as RDF types and contribute to relation linking in downstream steps.

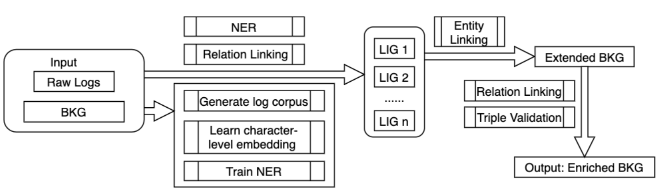

Figure 1: Process of Log Extraction

Instead of simply merging entities to BKG, we use Horn rules to infer relations. A Horn rule is a disjunction of atoms with at most one unnegated atom. In the implication form, they have the following format: $A_1 \wedge A_2 \wedge ... \wedge A_n \rightarrow B$, where $A_1 \wedge A_2 \wedge ... \wedge A_n$ is the body of the rule and B is the head. A positive rule has an atom as its head, while a negative rule has the head $\perp$. Thus, a negative rule is in the form of $A_1 \wedge A_2 \wedge ... \wedge A_n \rightarrow \perp$. Positive Horn rules can help to generate new triples, and negative rules can help to identify contradicting triples [1, 24].

In the real world, the knowledge grows as logs come on stream; as a result, reasoning [27, 28] cover a whole KG is more and more resource consuming over time. To this end, we apply a local to global strategy for inferring relations. Firstly, we aggregate logs by their source components. We believe the logs from the same source have more clustered interactions between them. Secondly, we construct LIGs using entities identified in aggregated log groups, after which rule inference over R_{local} is applied to each LIG. Then, the entities and triples are learned from each LIG locally.

Definition 3.2 (Local Reasoning). Given a set of logs $\mathbb{L}$ divided into n groups $g_1, \dots g_n$ of logs based on their sources, where $g_i (1 \leq i \leq n)$ is a group of logs from the same source, a subset $R_{local} \subseteq R_{positive}$ of local positive rules , a function f_g extracting a LIG from $g_i \in \mathbb{L}$, and the entailment closure operation Cn. The set of local logical consequences T_{g_i} of each g_i can be computed as follows.

$$T_{g_i} = Cn(f_g(g_i) \cup R_{local}) \tag{1}$$

Entities from different LIGs may have potential relationships, but are not linked within each local reasoning process. Hence, we apply reasoning with R_{global} against the whole extended BKG to infer implicit relations for all entities.

Definition 3.3 (Global Reasoning). Given an original BKG $G_{\mathbb{B}}$, local consequences T_{g_i}s, a subset $R_{global} \subseteq R_{positive}$ of global reasoning rules,and the entailment closure operation Cn. The extended BKG $G'_{\mathbb{B}}$ is computed as followed.

$$G'_{\mathbb{B}} = Cn(G_{\mathbb{B}} \cup \{T_{g_i} | 1 \leq i \leq n\} \cup R_{global}) \tag{2}$$

Although we perform relation linking by rules, there is no guarantee that these Horn rules are sound. Even if rules are sound for the current system, the conditions may change while the system evolves. Thus we validate new triples by a set of constraints and the triples that do not pass the validation are called negative triples. The triples from the input graphs $G_{\mathbb{B}}$ are not negative because $G_{\mathbb{B}}$ is from the last version and have passed its validation.

Definition 3.4 (Validation). The set of negative triples $\mathbb{N}_{triples}$ are ones involved in any proof of false $\perp$ in the extended graphs $G'_{\mathbb{B}}$ but which do not occur in the input $G_{\mathbb{B}}$. Here $\mathbb{T} = G'_{\mathbb{B}} \cup R_{global} \cup R_{negative}$ and $\mathbb{T} \vdash_{\pi} \perp$ means that π is a proof of $\perp$ in $\mathbb{T}$.

$$\mathbb{N}_{triples} = \{r(s,o) | \exists \pi.\ r(s,o) \dot{\in} (\pi, G_{\mathbb{B}}) \wedge \mathbb{T} \vdash_{\pi} \perp\} \tag{3}$$

[1] https://en.wikipedia.org/wiki/Named-entity_recognition

where function $\dot{\in}(\pi, G_{\mathbb{B}})$ returns the last incoming triple $r(s,o)$ that completes the proof π and $r(s,o) \notin G_{\mathbb{B}}$.

The new version of graph $\mathbb{G}$ is output after filtering out $\mathbb{N}_{triples}$.

Definition 3.5 (Filter). The final KG $\mathbb{G}$ is computed by removing negative triples $\mathbb{N}_{triples}$ from the extended graphs $G'_{\mathbb{B}}$.

$$\mathbb{G} = G'_{\mathbb{B}} \setminus \mathbb{N}_{triples} \tag{4}$$

3.3 Implementation

3.3.1 Entity Extraction. Instead of purely relying on templates, we combine the template-based approach and the NER approach to identify and label entities from log messages. Because raw log messages are usually unstructured and contain a lot of sub-words, acronyms and terminologies, we cannot leverage general pre-trained language models to tokenize and transform the raw log messages into vectors for NER. Hence we first generate a log text corpus by log parsing and learn character-level representation for log messages via FastText[2]. The word vector generated through FastText via N-Gram technique holds extra information about sub-words. We annotate the log corpus using entities from BKG and use the spaCy[3] toolkit to train its NER model. We combine Regex-based NER with the trainable NER model, because some entities like IP addresses are more suitable to be identified by Regex. The NER is performed on each log instance.

3.3.2 LIG Construction. We generate two types of graphs, one is the BKG, and the other is the LIGs. Initially the BKG holds concept level triples that are learned from technical manuals and converted from expert knowledge. The LIGs are temporary graphs consisting of entities identified from groups of logs. Triples learned from LIGs will be fused to the BKG, then the LIGs will be deleted.

To construct the LIGs, we firstly group the logs by multiple fixed columns like containers and components, which are the source of logs. In each individual log group, the entities are identified and divided into two sets, one is of self-contained entities identified in the grouping keys, another is of entities identified in the variable log messages and we are not sure if they belong to the source components. Each entity is assigned RDF type properties by its NER label and an additional class *SELFCONTAINED* indicating whether it is identified from the source components. For example, given this log from a service-oriented system:

```
[ERROR][2021-10-07 20:48:15.347 +08:00][197.28.1.23]
[Session-database-container-0se45][UserExecSvc-098]:
"Failed to request UserDomainSvc-034, unreachable"]
```

Container instance entity *Session-database-container-0se45* and service instance entity *UserExecSvc-098* are the source container and source service that generate this log, and service instance entity *UserDomainSvc-034* is likely an external entity that interacted with *UserExecSvc-098*.

We link entities within each LIG group. The positive rules for relation linking in LIGs are designed based on the entity classes. An example positive rule to infer relation *host_service* is:

$$\begin{aligned}&Container(?s) \land Service(?o) \land \\ &SELFCONTAINED(?s) \land SELFCONTAINED(?o) \\ &\quad \rightarrow host\_service(?s, ?o)\end{aligned}$$

3.3.3 Link Entities to BKG. Linking and aligning entities from LIGs to the BKG is a key step that facilitates the enrichment of the BKG. We cannot simply merge the triples from LIG to BKG, because they have different IRIs and the former have class such as SELFCONTAINED, that we would not like to include in the BKG. Different from typical entity linking task that aims to link text to entities in a KG, our entity linking module have two functions: 1) link entities from the LIGs to the BKG based on text and NER labels; 2) link instance entities from LIGs to corresponding concepts in the BKG. For example, entity *Session-database-container-0* is an instance of class *Session-Database-Container*. We would like to link instance entities with their concept entities, so that we can learn concept level triples from instance triples or link instance entities to each other based on their conceptual dependency. The entity linking module is a combination of SPARQL query [4], ElasticSearch [5] and character-level text similarity comparison. We implement the entity linking module to support case-insensitive, sub-word matching and context-aware, so that it has the ability to apply both exact matches and fuzzy matches. If no existing entity is found in BKG, we create a new entity and assign *rdf:type* as its label.

3.3.4 Rule-based Relation Linking. We apply positive rules to infer new triples in both LIG and BKG. The rules for LIG and BKG are different. The former links entities in an LIG with explicit relations; while the latter infers implicit relations for both concept entities and instance entities.

We manually scripted such positive rules to involve domain knowledge in the BKG. This positive rule set could possibly be extended by rule mining tools such as AMIE [10, 11, 16] and Rudik [1, 24]. We use the Pellet reasoner [38] to infer triples. BKG reasoning happens after the set of LIG triples are merged into BKG.

3.3.5 Triple Validation. Given a new triple $r\ (s,o)$ and an example negative rule such as:

$$r\ (?s, ?o) \land r'\ (?s, ?o) \rightarrow \bot$$

If $r'\ (s,o)$ exists in BKG, then $r\ (s,o)$ is invalid. Practically we parse the negative rule to an SPARQL query and check whether there exist any contradictions in the BKG against the new triple.

The negative rules are converted from BKG's ontology or scripted manually to involve expert knowledge. There are different types of negative rules, such as class disjointness $C_1(?x) \land C_2(?x) \rightarrow \bot$, relation restriction $r(?a, ?b) \land r(?b, ?a) \rightarrow \bot$ and rules that restricts the system's behaviours: $r1(?a, ?b) \land r2(?a, ?c) \rightarrow \bot$.

4 USE CASE

We tested our proposed approach on an enterprise service-oriented network system. The service-oriented architecture breaks the system logic into different, small services where each one has a single task or responsibility. The services are hosted and executed inside containers such as Docker [6]. The different services communicate and cooperate with each other to provide the system functionality

[2] https://fasttext.cc/
[3] https://spacy.io/
[4] https://www.w3.org/2001/sw/wiki/SPARQL
[5] https://www.elastic.co/elasticsearch/
[6] https://www.docker.com/

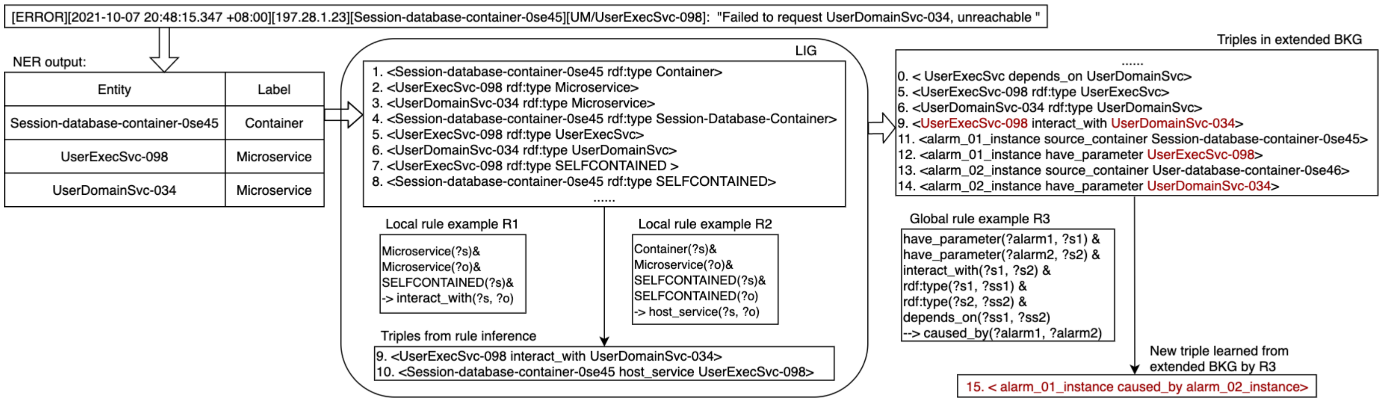

Figure 2: A use case of extracted knowledge for root cause analysis.

as a whole. Our test data were logs exported from an enterprise operation and management platform. In practice, the local to global strategy reduces the reasoning query space. Since reasoning is segmented within a set of LIGs, only a subset of inference rules are applied to the whole BKG to infer implicit relations between entities from different LIGs. Also, the reasoning in LIGs can be performed in parallel to save execution time. Our use case demonstrated (1) how to learn knowledge from logs and (2) how the knowledge learned from logs facilitates root cause analysis.

4.1 Extracting Knowledge from Logs

A full set of log groups is too large to demonstrate in this paper, therefore we use a single log instance as an example and pretend it is a log group having only one log instance. The extraction process is illustrated in Figure 2. The process is described as below:

(1) The NER identifies a list of entities from the log.
(2) The entities and labels are converted into RDF triples as $\{1, 2, 3\}$. Three entities are linked to concepts in BKG: $\{4, 5, 6\}$. Two entities are assigned with rdf:type SELFCONTAINED as they are in fixed columns: $\{7, 8\}$. These triples compose an initial LIG.
(3) Apply the set of positive rules on this LIG, and learn a set of triples as $\{9, 10\}$. The local rule $R1$ infers triple 9 and $R2$ infers triple 10.
(4) Align and merge triples from LIG to the background graph. Extend the background graph with triples $\{1, 2, 3, 4, 5, 6, 9, 10\}$.
(5) Infer new triples based on all triples in the background graph. 0 is the triple from the initial background graph, $\{5, 6, 9\}$ are from the above LIG, $\{11, 12, 13, 14\}$ are from other LIGs.
(6) New triple 15 is learned from the extended BKG, based on triples $\{0, 5, 6, 9, 12, 14\}$ and rule $R3$.

After extraction and inference, we expand the BKG with a set of instance level triples $\{1, 2, 3, 4, 5, 6, 9, 10\}$ from the log and a triple 15 that connects two alarm instance entities.

4.2 Use the Log Knowledge Graph for Root Cause Analysis

The background knowledge now contains initial concept level triples and triples learned from a set of LIGs. In Figure 2, triples $\{12, 13, 14, 15\}$ describes two alarm events, both related to the triples from our example LIG. The alarm events are reported to the operation and management platform, but administrators don't know the causal relationship given the huge list of alarms reporting similar alarms on different containers and services. By applying $R3$ on the whole KG, we learned triple 16, showing the causal relation between two alarm instances.

5 CONCLUSION AND FUTURE WORK

This paper introduces a system LEKG for automated knowledge model construction from arbitrary log data. The proposed system extracts and learns triples from unstructured logs to construct a log KG based on a BKG. The key idea of this approach is to utilize Horn rules from background knowledge to infer additional triples and validate new triples. From the angle of practice, we proposed a local to global strategy for triple inference, which reduced reasoning query space. Finally, we demonstrate the knowledge extraction process and how it facilitates root cause analysis.

A limitation evident from our experiment was the uncertain coverage of the rule set and the effort of preparing the rules. This limitation could be eased by rule-mining technologies to some extent [1, 20, 36]. The automatic rule-mining tools [1, 10, 11, 16] rely on a large set of examples. This improvement is not applicable from a cold start but can be arranged when the KG is expanded to a large size. Usually the rules learned by automatic tools are not as concise as those scripted by experts. This problem can be tackled by the ABC [18] system, which has ability of repairing Horn rules based on an ABox with the minimal changes w.r.t. entrenchment scores [17]. In our future work, the triples that failed the validation will play an important role in revising rules and repairing KG with minimal changes. Furthermore, there might also a need to use some uncertainty [37] mechanisms, such as fuzzy extensions [29, 30, 39], possibilistic extensions [34, 35] or probabilistic extensions [12, 21], of knowledge graphs in the future. We also plan to look into Shape constraints [14].

ACKNOWLEDGMENTS

The authors would like to thank Huawei for supporting the research on which this paper was based under grant CIENG4721/LSC.

REFERENCES

[1] Naser Ahmadi, Viet Phi Huynh, Vamsi Meduri, Stefano Ortona, and Paolo Papotti. 2020. Mining Expressive Rules in Knowledge Graphs. *Journal of Data and Information Quality* 12, 2 (2020). https://doi.org/10.1145/3371315

[2] Álvaro Brandón, Marc Solé, Alberto Huélamo, David Solans, María S. Pérez, and Victor Muntés-Mulero. 2020. Graph-based root cause analysis for service-oriented and microservice architectures. *Journal of Systems and Software* 159 (2020). https://doi.org/10.1016/j.jss.2019.110432

[3] Harith A. Dawood. 2014. Graph theory and cyber security. *Proceedings - 3rd International Conference on Advanced Computer Science Applications and Technologies, ACSAT 2014* (2014), 90–96. https://doi.org/10.1109/ACSAT.2014.23

[4] Youcef Djenouri, Asma Belhadi, and Philippe Fournier-Viger. 2018. Extracting useful knowledge from event logs: A frequent itemset mining approach. *Knowledge-Based Systems* 139 (2018), 132–148. https://doi.org/10.1016/j.knosys.2017.10.016

[5] Hristo Djidjev, Gary Sandine, Curtis B. Storlie, and Scott Vander Wiel. 2011. Graph Based Statistical Analysis of Network Traffic. *Mlg '11* (2011), 8. https://www.cs.purdue.edu/mlg2011/papers/paper{_}10.pdf

[6] Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. 2017. DeepLog: Anomaly detection and diagnosis from system logs through deep learning. *Proceedings of the ACM Conference on Computer and Communications Security* (2017), 1285–1298. https://doi.org/10.1145/3133956.3134015

[7] Andreas Ekelhart, Fajar J. Ekaputra, and Elmar Kiesling. 2021. The SLOGERT Framework for Automated Log Knowledge Graph Construction. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 12731 LNCS (2021), 631–646. https://doi.org/10.1007/978-3-030-77385-4_38

[8] Andreas Ekelhart, Elmar Kiesling, and Kabul Kurniawan. 2018. Taming the logs – Vocabularies for semantic security analysis. *Procedia Computer Science* 137 (2018), 109–119. https://doi.org/10.1016/j.procs.2018.09.011

[9] Alexandre P. Francisco, Ricardo Baeza-Yates, and Arlindo L. Oliveira. 2012. Mining query log graphs towards a query folksonomy. *Concurrency and Computation: Practice and Experience* 24, 17 (2012), 2179–2192. https://doi.org/10.1002/cpe.1773

[10] Luis Galárraga, Christina Teflioudi, Katja Hose, and Fabian M. Suchanek. 2015. Fast rule mining in ontological knowledge bases with AMIE+. *VLDB Journal* 24, 6 (2015), 707–730. https://doi.org/10.1007/s00778-015-0394-1

[11] Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. 2013. AMIE: Association Rule Mining under Incomplete Evidence in Ontological Knowledge Bases. (2013), 413–422. https://doi.org/10.1145/2488388.2488425

[12] Víctor Gutiérrez-Basulto, Jean Christoph Jung, Carsten Lutz, and Lutz Schröder. 2017. Probabilistic description logics for subjective uncertainty. *Journal of Artificial Intelligence Research* 58(1) (2017), 1–66.

[13] Pinjia He, Jieming Zhu, Shilin He, Jian Li, and Michael R. Lyu. 2016. An evaluation study on log parsing and its use in log mining. *Proceedings - 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2016* (2016), 654–661. https://doi.org/10.1109/DSN.2016.66

[14] Aidan Hogan. 2020. Shape Constraints and Expressions. In *The Web of Data*. Springer, 449–513.

[15] Elmar Kiesling, Andreas Ekelhart, Kabul Kurniawan, and Fajar Ekaputra. 2019. *The SEPSES Knowledge Graph: An Integrated Resource for Cybersecurity*. Vol. 11779 LNCS. Springer International Publishing. 198–214 pages. https://doi.org/10.1007/978-3-030-30796-7_13

[16] Jonathan Lajus, Luis Galárraga, and Fabian Suchanek. 2020. Fast and Exact Rule Mining with AMIE 3. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 12123 LNCS (2020), 36–52. https://doi.org/10.1007/978-3-030-49461-2_3

[17] Xue Li, Alan Bundy, and Eugene Philalithis. 2021. Signature Entrenchment and Conceptual Changes in Automated Theory Repair. In *The Ninth Annual Conference on Advances in Cognitive Systems*. Cognitive Systems Foundation.

[18] Xue Li, Alan Bundy, and Alan Smaill. 2018. ABC repair system for datalog-like theories. *IC3K 2018 - Proceedings of the 10th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management* 2 (2018), 335–342. https://doi.org/10.5220/0006959703350342

[19] Jian Guang Lou, Qiang Fu, Yi Wang, and Jiang Li. 2010. Mining dependency in distributed systems through unstructured logs analysis. *Operating Systems Review (ACM)* 44, 1 (2010), 91–96. https://doi.org/10.1145/1740390.1740411

[20] Ravi Lourdusamy and Stanislaus Abraham. 2020. A Survey on Methods of Ontology Learning from Text. May (2020), 113–123. https://doi.org/10.1007/978-3-030-38501-9_11

[21] Thomas Lukasiewicz. 2008. Expressive probabilistic description logics. *Artificial Intelligence* 172(6-7) (2008), 852–883.

[22] Weibin Meng, Ying Liu, Yichen Zhu, Shenglin Zhang, Dan Pei, Yuqing Liu, Yihao Chen, Ruizhi Zhang, Shimin Tao, Pei Sun, and Rong Zhou. 2019. Loganomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs. *IJCAI International Joint Conference on Artificial Intelligence* 2019-Augus (2019), 4739–4745. https://doi.org/10.24963/ijcai.2019/658

[23] Adam Oliner, Archana Ganapathi, and Wei Xu. 2012. Advances and challenges in log analysis. *Commun. ACM* 55, 2 (feb 2012), 55–61. https://doi.org/10.1145/2076450.2076466

[24] Stefano Ortona, Venkata Vamsikrishna Meduri, and Paolo Papotti. 2018. Robust discovery of positive and negative rules in knowledge bases. *Proceedings - IEEE 34th International Conference on Data Engineering, ICDE 2018* (2018), 1180–1191. https://doi.org/10.1109/ICDE.2018.00108

[25] J.Z. Pan, D. Calvanese, T. Eiter, I. Horrocks, M. Kifer, F. Lin, and Y. Zhao. 2017. *Reasoning Web: Logical Foundation of Knowledge Graph Construction and Querying Answering*. Springer.

[26] J.Z. Pan, G. Vetere, J.M. Gomez-Perez, and H. Wu. 2016. *Exploiting Linked Data and Knowledge Graphs for Large Organisations*. Springer.

[27] Jeff Z. Pan and Ian Horrocks. 2002. Reasoning in the SHOQ($\mathbf{D_n}$) Description Logic. In *Proceedings of the 2002 International Workshop on Description Logics (DL2002)*, Ian Horrocks and Sergio Tessaris (Eds.).

[28] Jeff Z. Pan and Ian Horrocks. 2003. Web Ontology Reasoning with Datatype Groups. In *Proc. of the International Semantic Web Conference*. 47–63.

[29] Jeff Z. Pan, Giorgos B. Stamou, Vassilis Tzouvaras, and Ian Horrocks. 2005. f-SWRL: A Fuzzy Extension of SWRL. In *Artificial Neural Networks: Formal Models and Their Applications - ICANN*. 829–834.

[30] Jeff Z. Pan, Giorgos Stoilos, Giorgos Stamou, Vassilis Tzouvaras, and Ian Horrocks. 2006. f-SWRL: A Fuzzy Extension of SWRL. *Journal of Data Semantic* (2006), 28–46.

[31] Antonio Pecchia, Ingo Weber, Marcello Cinque, and Yu Ma. 2020. Discovering process models for the analysis of application failures under uncertainty of event logs. *Knowledge-Based Systems* 189 (2020). https://doi.org/10.1016/j.knosys.2019.105054

[32] Kexin Pei, Zhongshu Gu, Brendan Saltaformaggio, Shiqing Ma, Fei Wang, Zhiwei Zhang, Luo Si, Xiangyu Zhang, and Dongyan Xu. 2016. HERCULE: Attack story reconstruction via community discovery on correlated log graph. *ACM International Conference Proceeding Series* 5-9-Decemb, 3 (2016), 583–595. https://doi.org/10.1145/2991079.2991122

[33] Aditya Pingle, Aritran Piplai, Sudip Mittal, Anupam Joshi, James Holt, and Richard Zak. 2019. Relext: Relation extraction using deep learning approaches for cybersecurity knowledge graph improvement. *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2019* (2019), 879–886. https://doi.org/10.1145/3341161.3343519 arXiv:1905.02497

[34] Guilin Qi, Qiu Ji, Jeff Z. Pan, and Jianfeng Du. 2011. Extending Description Logics with Uncertainty Reasoning in Possibilistic Logic. *International Journal of Intelligent Systems* 26(4) (2011).

[35] Guilin Qi, Jeff Z. Pan, and Qiu Ji. 2007. A Possibilistic Extension of Description Logics. In *Proc. of 2007 International Workshop on Description Logics (DL2007)*.

[36] Wilhelm Schickard-lnstitute, Ulrich Gntzer, Wilhelm Schickard-lnstitute, Daimlerchrysler Ag, and F T Ad. [n. d.]. Algorithms for Association Rule Mining - A General Survey and Comparison. 2, 1 ([n. d.]), 58–64.

[37] Murat Sensoy, Achille Fokoue, Jeff Z. Pan, Timothy J. Norman, Yuqing Tang, Nir Oren, and Katia P. Sycara. 2013. Reasoning about uncertain information and conflict resolution through trust revision. In *Proc. of the International conference on Autonomous Agents and Multi-Agent Systems, AAMAS*, Maria L. Gini, Onn Shehory, Takayuki Ito, and Catholijn M. Jonker (Eds.). 837–844.

[38] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. 2007. Pellet: A practical OWL-DL reasoner. *Web Semantics* 5, 2 (2007), 51–53. https://doi.org/10.1016/j.websem.2007.03.004

[39] Giorgos Stoilos, Giorgos B. Stamou, and Jeff Z. Pan. 2006. Handling Imprecise Knowledge with Fuzzy Description Logic. In *the Proc. of the 2006 International Workshop on Description Logics (DL2006)*.

[40] Hudan Studiawan, Christian Payne, and Ferdous Sohel. 2017. Graph clustering and anomaly detection of access control log for forensic purposes. *Digital Investigation* 21 (2017), 76–87. https://doi.org/10.1016/j.diin.2017.05.001

[41] Jan Svacina, Jackson Raffety, Connor Woodahl, Brooklynn Stone, Tomas Cerny, Miroslav Bures, Dongwan Shin, Karel Frajtak, and Pavel Tisnovsky. 2020. On Vulnerability and Security Log analysis: A Systematic Literature Review on Recent Trends. *ACM International Conference Proceeding Series* (2020), 175–180. https://doi.org/10.1145/3400286.3418261

[42] Ding Yuan, Haohui Mai, Weiwei Xiong, Lin Tan, Yuanyuan Zhou, and Shankar Pasupathy. 2010. SherLog: Error diagnosis by connecting clues from run-time logs. *International Conference on Architectural Support for Programming Languages and Operating Systems - ASPLOS* (2010), 143–154. https://doi.org/10.1145/1736020.1736038

[43] Jieming Zhu, Shilin He, Jinyang Liu, Pinjia He, Qi Xie, Zibin Zheng, and Michael R. Lyu. 2019. Tools and Benchmarks for Automated Log Parsing. *Proceedings - 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice, ICSE-SEIP 2019* (2019), 121–130. https://doi.org/10.1109/ICSE-SEIP.2019.00021 arXiv:1811.03509

[44] De Qing Zou, Hao Qin, and Hai Jin. 2016. UiLog: Improving Log-Based Fault Diagnosis by Log Analysis. *Journal of Computer Science and Technology* 31, 5 (2016), 1038–1052. https://doi.org/10.1007/s11390-016-1678-7