Proceedings of 12th IOE Graduate Conference
Peer Reviewed
ISSN: 2350-8914 (Online), 2350-8906 (Print)
Year: 2022 Month: October Volume: 12

Automated Log Parsing through Named Entity Recognition

Saloni Shikha [a], Arun Kumar Timalsina [b]

[a, b] *Department of Electronics and Computer Engineering, Pulchowk Campus, IOE, Tribhuvan University, Nepal*
[a] shikhasalonee@gmail.com, [b] t.arun@pcampus.edu.np

Abstract
Modern software-intensive systems generate millions of logs each day for troubleshooting purposes. These devices log their activities and events in some form resulting in exponential growth in the number of logs generated. These logs are in thousands of formats, some of which are structured while others are not. Extracting information from the structured logs is simple and can be done accurately using classical programming approaches. However, the remaining semi-structured logs are difficult to work with. Structured logs refer to the logs following specific format such as json, xml, cef and so on. Semi-structured logs here refers to the logs that follow some specific log template generated from the logging statement written by the system developer which might not necessarily have keys demarcated in the log itself. An example of a system whose logs are semi-structured is the Unix system. Unix logs come with common header formats with thousands of variants in the log body format. While using classical programmatic approach, whenever a new log format appears, extra effort has to be put to extract the information from the log which is a never-ending process. This research work aims to solve this through a deep learning-based named entity recognition algorithm. Name Entity Recognition, a modern Natural Language Processing (NLP) approach, is used to create a model trained with large amount of known data points that helps to identify and extract meaningful information from new logs of a variety of formats. The scope of this research work is to automate log parsing for such semi-structured logs generated from Unix processes. This project has studied creation of a Bidirectional Long Short-Term Memory(BLSTM) model to extract important values from these logs and assign them certain field names. The model was able to achieve a F-measure of 89% on the test set.

Keywords
Log parser, Long short-term memory, Named entity recognition, Regex-based parsers

1. Introduction

With growth in the number and complexity of devices in the world, the number of logs is increasing exponentially. Logs are being generated in a wide variety of formats which do not follow any particular format and the task of parsing these logs systematically and deriving useful meaning from these has become increasingly difficult. The most common and prevalent method of parsing of logs currently being used is regex-based parsers. The limitation of regex-based parsers is that regex-based parsers are very specific in their format and thus the parsers need to be created manually for each log format and must be upgraded with each change in format. Such parsers made for one type of log format are useless for logs that are generated in another format. Even a slight change in the format like during version change or even during data transfer an extra space character causes these kinds of parsers to fail. This has been an increasing concern in the field of security since log parsing is the first step before performing any kind of log analytics.

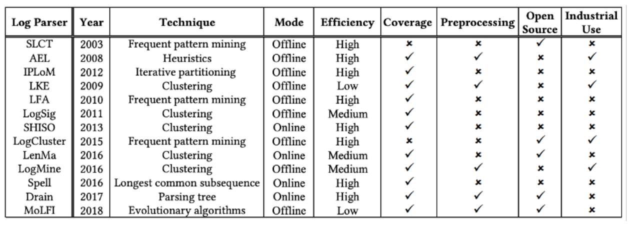
At present the cybersecurity industry is facing a huge expense of manpower in log analytics. Security information and event management (SIEM) applications require information extracted from the logs as the first step before any other kind of log analytics. Currently, most of the organisations that work on log analytics or any product relating to log analytics, for eg, SIEM industries, have employed manual signature-based methods for extracting information from the logs. This is not only time-taking and costly but also requires constant upgrading and support. A separate group of manpower is needed for supporting these never-ending modification in the log formats. Thus, the parsing of logs is a major challenge in the security industry which needs to be automated.

2. Related Literature

A structured log may be in some structured format such as xml, json, cef format where the key-value pairs can be extracted from the log itself. Parsing of such logs is not a complicated task that can be handled using programmatic approaches. However, in case of Unix logs, ssh logs, etc. where the fields of the log follow some specified format, the semantics and meaning of the fields may not be clear from the log itself. This requires documentation of the log and some parsing mechanism following the pattern stated in the documentation. However, limited research has been done in the field of automatically parsing semi-structured logs.

2.1 Log Clustering and Pattern Recognition

Research works have been done and some tools have been developed for this purpose since the early 2000s. Some research works have been conducted to examine viability of log analytics using deep learning [1]. The history of automated log parsing begins with Simple Logfile Clustering Tool (SLCT), which is one of the earliest tools that was designed with the purpose of finding clusters in the logfile where each cluster corresponds to some pattern which is found to occur frequently. Zhu et.al.[2] have performed benchmarking analysis of several old log parsing tools developed during research and performed a comparative analysis of the tools. A brief summary of the tools [3] has been shown in Fig 1.



Log Parser	Year	Technique	Mode	Efficiency	Coverage	Preprocessing	Open Source	Industrial Use
SLCT	2003	Frequent pattern mining	Offline	High	×	×	✓	×
AEL	2008	Heuristics	Offline	High	✓	✓	×	✓
IPLoM	2012	Iterative partitioning	Offline	High	✓	×	×	×
LKE	2009	Clustering	Offline	Low	✓	✓	×	✓
LFA	2010	Frequent pattern mining	Offline	High	✓	×	×	×
LogSig	2011	Clustering	Offline	Medium	✓	×	×	×
SHISO	2013	Clustering	Online	High	✓	×	×	×
LogCluster	2015	Frequent pattern mining	Offline	High	×	×	✓	✓
LenMa	2016	Clustering	Online	Medium	✓	×	✓	×
LogMine	2016	Clustering	Offline	Medium	✓	✓	×	✓
Spell	2016	Longest common subsequence	Online	High	✓	×	×	×
Drain	2017	Parsing tree	Online	High	✓	✓	✓	×
MoLFI	2018	Evolutionary algorithms	Offline	Low	✓	✓	✓	×

Figure 1: Summary of Historical Log Parsing Tools

Likewise, Joshi et. al. [4] have used the method of log clustering through locally sensitive signature where the similarity between log messages have been identified by parsing the messages followed by logically analysing the signature bit stream that has been associated with them. Log messages have been clustered based on their percentage similarity with the pattern signatures that have been stored in the database. If the similarity percentage is within a specified limit, the log is included in that cluster else a new signature is created and stored in the database for the pattern of this new log.

Drain [5] makes use of a fixed depth parse tree that is capable of parsing logs in a streaming manner in a timely fashion. Likewise, Logram [6] makes use of n-gram dictionaries for log parsing with spark nodes for scaling out efficiently.

2.2 Log Parsing as a Named Entity Recognition Problem

The extraction of entity types from log can be related to a Named Entity Recognition problem. For example, in a sentence ”User abcd authenticated successfully”, abcd is the name of user which is a variable, the log format being ”User ⟨ user⟩ authenticated successfully.” Determining that ”abcd” is the name of a user in this log, ”authenticated” being the action taking place and ”successful” being the status is a key challenge.

In a Named Entity Recognition problem, the relationship between adjacent words needs to be considered and utilised. Since BLSTM provides a proper representation of each field thereby catching the context of the surrounding words, BLSTM can learn the features and perform the task well.

Pokharel [7] has used a classifier model using both Naïve Bayes and Support Vector classifier and performed the task of log parsing through named entity recognition on Windows Security Event Logs. Likewise, Chiu et. al. [8] have used Bi-directional LSTM for Named Entity Recognition and have used CNN for performing character embedding which is then fed into the LSTM.

3. Methodology

3.1 System Architecture

A Bidirectional Long Short-Term Memory (BLSTM) has been used for extracting the named entities from the log messages. A Bidirectional LSTM is a derivation of Recurrent Neural Networks (RNN) which is useful for processing sequential data. Since event log parsing involves sequences of log fields such as timestamp followed by process name, process-id, service name, etc., BLSTM can give considerable results in this field. The reason for using

BLSTM over LSTM is that BLSTM is based on both context directions while LSTM does not consider both directions. Named Entity Recognition models have been found to give better results with char embedding compared to when only word embedding is used. Both word embedding and char embedding have been calculated before feeding into the LSTM model. This will later be tweaked and checked with various modification to determine the best configuration.

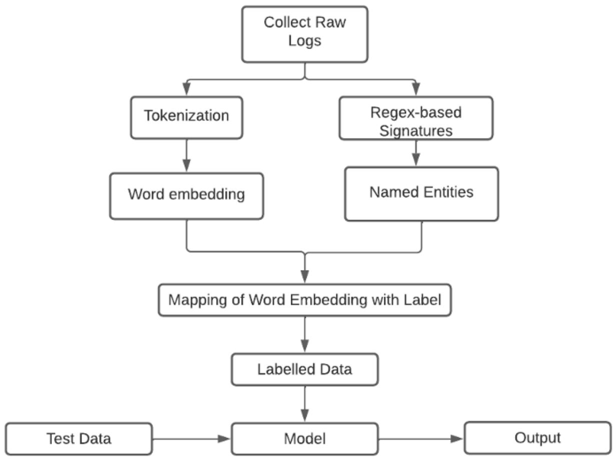


Figure 2: Project Methodology

3.2 System Block Diagram

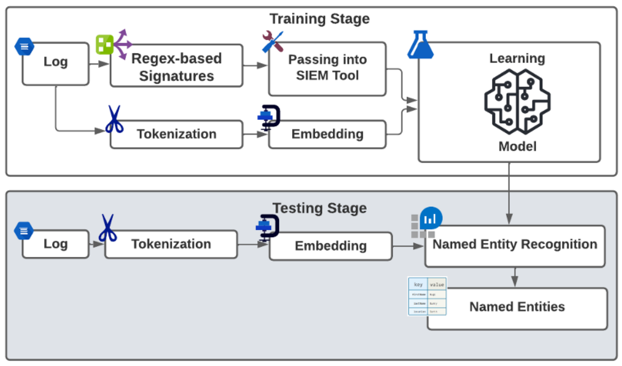


Figure 3: System Block Diagram

This shows the intended system block diagram of the proposed system. The workings of the different stages are explained in section 4.2.

3.3 Model Details

The model used is a Bidirectional LSTM model. The model was trained for 30 epochs with early stopping because 30 should be enough for the model to be trained. The model uses hyperparameters as shown in table 1.

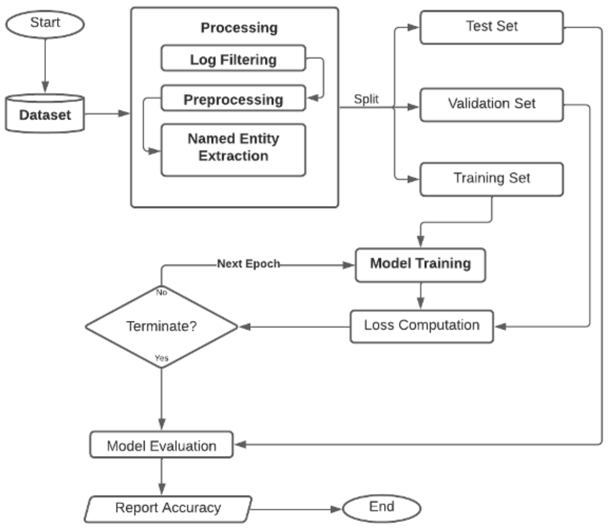


Figure 4: Flow Chart of Model Training

Parameter	Value
Number of epochs	30
Dropout	0.5
Batch size	20
Optimizer	adam
Learning Rate	0.001
Learning Rate Decay	0.9

Table 1: Model Implementation Details

3.4 Log Data Source

The dataset used for modelling evaluation of the project consists of semi-structured logs. Semi-structured logs here refer to the logs which do not follow specific format that is capable of explicitly representing key-value pairs within itself. For example, a JSON format log is structured since it explicitly contains the values as well as the keys for the values i.e. the description of what field the value represents. The logs are limited to Unix logs for this project. The dataset consists of Unix logs gathered from open-source databases. A few log sources from where the logs will be collected include:

i. Loghub:
Loghub, A Large Collection of System Log Datasets towards Automated Log Analytics [9], is an open-source dataset hosted in github which consists of 77GB of rawlogs from different sources including Unix. The logs are not sanitised, anonymized or modified in any way. This contains Unix logs collected over a time span of 263.9 days containing 25567 messages.

Selected logs are filtered from these based on the information they carry. This dataset also contains plenty of logs having no useful information in them. Likewise, logs that are repeated with only difference in specific fields such as user or datetime have been removed.

ii. Logpoint:
Unix logs have been collected from Logpoint. The logs from Logpoint include logs from a wide variety of Unix processes that is used in combination with the logs from Loghub. These logs have been anonymized in order to protect the data privacy.

3.5 Pre-processing

3.5.1 Tokenization

The raw data consists of multiple files containing log entries. An input of one sentence is taken, the sentence is split by space into smaller units i.e., words or terms. Each of these terms now acts as token for further processing. In this way, the log entries have now been processed into a sequence of tokens. However, these tokens fail to split the logs into single entities at times. For example, in the following log, the process and process_id are separate entities that have separate existence but are treated as one because of the tokenization method used.
Feb 12 10:54:25 ftpd[27480]: lost connection to 192.168.5.250 [192.168.5.250]
Here, ftpd is the process and 27480 is the process_id when we drill down further but the entire ftpd[27480] has been treated as process here. This will be properly tokenized and processed in the future.

3.5.2 Word and Character Embedding

The tokens that have been generated from raw log entries are now converted into a sequence of word embeddings. GloVe[10] has been used as the pretrained word embedding. GloVe has been used to initialize the look-up dictionary required for word embedding. GloVe has been used as the word embedding since it uses Wikipedia data which is representative enough to get the context between words present in the message part of a log. The vocabulary used in its training includes common words, thus this is suitable to be used in log data as well. Here, every word gets represented as a unique embedding.

3.6 Evaluation Methodology for NER System

For the authenticity of Named Entity Recognition(NER) systems, its thorough evaluation is necessary. There are many prevalent classification techniques and the evaluation system should be chosen that is best for the technique. Performance of NER systems are compared with the actual annotation provided by linguists, in this project, the actual labels added by human-written signatures or rules.
Precision (P) and recall (R) are the metrics that are mostly used to measure the performance of information extraction systems such as NER system. While precision mostly deals with substitution and insertion errors, recall is mostly about substitution and deletion errors. Thus, a single performance metric that can deal with all the three types of errors simultaneously is desirable, which is the F-measure. F-measure is the weighted combination of Precision and Recall. In our case, the metrics are given by:

$$Precision = \frac{Correct\ Responses}{Correct + Incorrect + Missing\ Responses} \tag{1}$$

$$Recall = \frac{Number\ of\ correct\ responses}{Correct + Incorrect + Spurious\ Responses} \tag{2}$$

$$F-measure = \frac{2 * Precision * Recall}{Precision + Recall} \tag{3}$$

Here, response refers to the result/labelling provided by our NER system. If we consider the correct label (tagged by human, in our case human-written signature) to be the answer key,
Correct $\rightarrow$ Instances where Response = Answer key.
Incorrect $\rightarrow$ Instances where Response $\neq$ Answer key.
Missing $\rightarrow$ Instances where Response is unlabelled but Answer key is labelled.
Spurious $\rightarrow$ Instances where Response is labelled but Answer key is unlabelled.

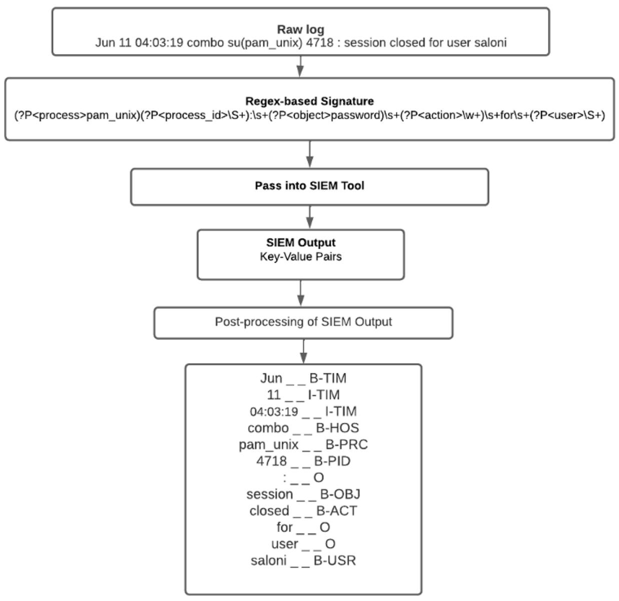
3.7 Tools and Technologies

Python, Numpy, Tensorflow, Nltk, Matplotlib, Jupyter Notebook, Google Colab, SIEM Tool, Microsoft Office, Google Docs

4. Results and Discussion

4.1 Experimental Setup

The data format was text format. The log definition reference for the logs was used as the standard semantics for the log messages. The field-value mapping was generated using the log definition and a standard log was built using a SIEM tool which was used to create the labelled data. The validation was done against this standard labelled data which was then used to calculate the performance metrics. The outcome of the project model was compared with the generated labelled data to validate the result and calculate performance metrics.



```
Jun 11 04:03:19 combo su(pam_unix) 4718 : session closed for user saloni
```

```
(?P<process>pam_unix)(?P<process_id>\S+):\s+(?P<object>password)\s+(?P<action>\w+)\s+for\s+(?P<user>\S+)
```

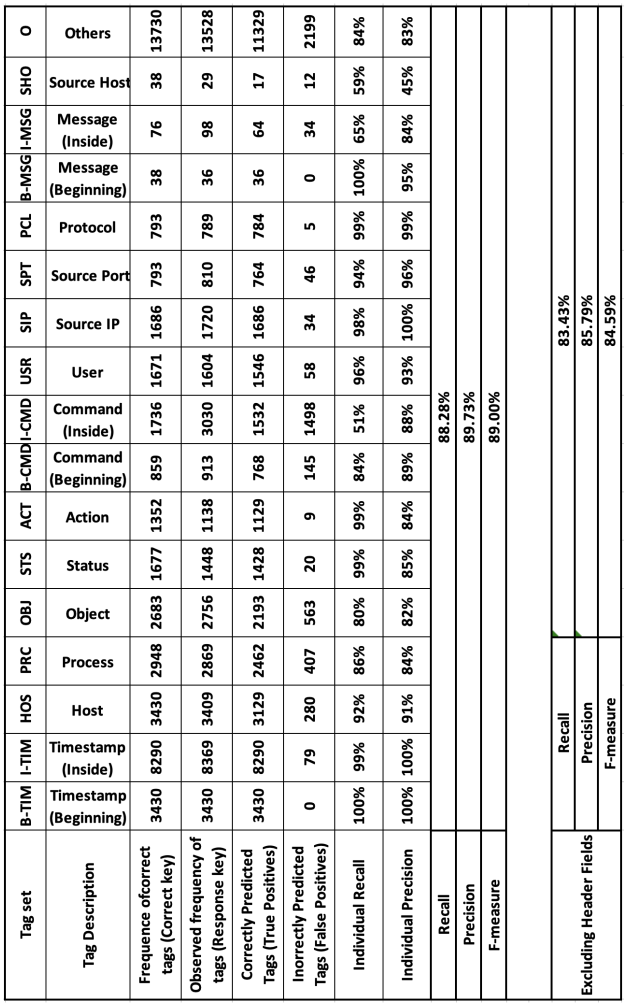
```
Jun _ _ B-TIM
11 _ _ I-TIM
04:03:19 _ _ I-TIM
combo _ _ B-HOS
pam_unix _ _ B-PRC
4718 _ _ B-PID
: _ _ O
session _ _ B-OBJ
closed _ _ B-ACT
for _ _ O
user _ _ O
saloni _ _ B-USR
```

Figure 5: Preprocessing steps for creation of training and testing set

4.2 Creation of Training and Testing Dataset

This project concentrates on building a machine learning model capable of predicting named entities from input raw logs. However, the dataset consists of log entries containing many different log formats. Majority of the task is preparation of suitable data for training the neural network model. Exploration, preprocessing, and annotation of the dataset is necessary to be fed into the model as input. Regex-based signatures have been used to extract the entity names from the data. The logs are passed through these signatures which classify its meaningful entities into its respective entity-name. In other words, key-value pairs are generated from the log entries in this step. The values which do not contain significant information are simply tagged as O. The dataset is splitted into 60%, 20% and 20% for training, validation and testing sets respectively.

The fields and corresponding tags used for the fields are in mentioned in detail in figure 6.



Tag set		B-TIM	I-TIM	HOS	PRC	OBJ	STS	ACT	B-CMD	I-CMD	USR	SIP	SPT	PCL	B-MSG	I-MSG	SHO	O
Tag Description		Timestamp (Beginning)	Timestamp (Inside)	Host	Process	Object	Status	Action	Command (Beginning)	Command (Inside)	User	Source IP	Source Port	Protocol	Message (Beginning)	Message (Inside)	Source Host	Others
Frequence ofcorrect tags (Correct key)		3430	8290	3430	2948	2683	1677	1352	859	1736	1671	1686	793	793	38	76	38	13730
Observed frequency of tags (Response key)		3430	8369	3409	2869	2756	1448	1138	913	3030	1604	1720	810	789	36	98	29	13528
Correctly Predicted Tags (True Positives)		3430	8290	3129	2462	2193	1428	1129	768	1532	1546	1686	764	784	36	64	17	11329
Inorrectly Predicted Tags (False Positives)		0	79	280	407	563	20	9	145	1498	58	34	46	5	0	34	12	2199
Individual Recall		100%	99%	92%	86%	80%	99%	99%	84%	51%	96%	98%	94%	99%	100%	65%	59%	84%
Individual Precision		100%	100%	91%	84%	82%	85%	84%	89%	88%	93%	100%	96%	99%	95%	84%	45%	83%
Recall										88.28%								
Precision										89.73%								
F-measure										89.00%								
Excluding Header Fields	Recall											83.43%						
	Precision											85.79%						
	F-measure											84.59%						

Figure 6: Performance Metrics of Named Entity Recognition Output

4.3 Results and Discussion

Various evaluation metrics have been calculated. Precision, recall and F1-score for each tag were computed. Macro averages of all these three metrics have also been computed for the model. The actual distribution of the different tags in the test dataset is shown in figure 7 along with the model prediction statistics.

The model shows high precision and recall for the header fields since they follow pre-defined format in terms of their structure and position in the logs. Header fields mostly include fields such as Timestamp (including date, month, time of the day), host and

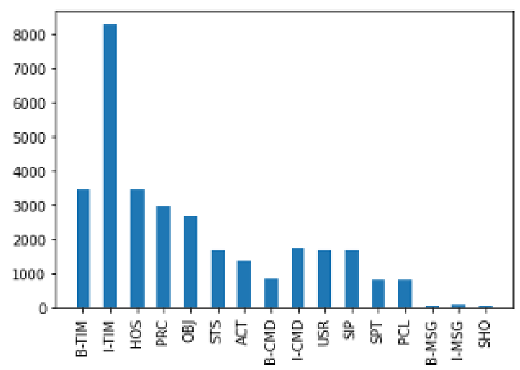


Figure 7: Occurrence Frequency of Fields in Test Data Set

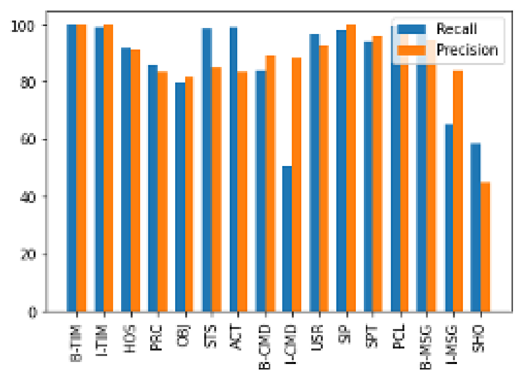


Figure 8: Tag-wise Comparison of Performance Metrics

process. However, the model is found to often miss parsing of a few fields important from the perspective of security such as the 'authentication_method', 'authentication_type'. Likewise, since Command field has a large variation in its value, the model seems to be giving a lot of false positives for the field Command i.e. CMD. As a result, the individual recall for this field is extremely low.

The overall performance metrics of the model with all fields included are shown in table 3:

Recall	88.28%
Precision	89.73%
F-measure	89.00%

Table 2: Overall Performance Metrics

The prediction of header fields is comparatively more accurate compared to other fields since these fields are relatively constant in the logs with a constant position as well. Since the performance metrics are highly affected by the correct prediction of the header fields, another performance metrics with these fields excluded have been calculated which is shown in table 3.

Recall	83.42%
Precision	85.78%
F-measure	84.59%

Table 3: Performance Metrics Excluding Header Fields

The performance metrics for individual tags has been plotted and compared in figure 8.

The variation of train and validation loss of the model with epoch has been plotted in figure 9.

Advantage of the model over regex-based signatures:

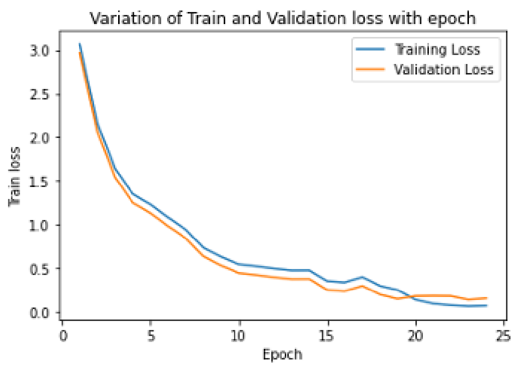


Figure 9: Variation of Train and Validation loss of the Final Model

The model gives more flexibility over rule-based or regex-based signatures in that when log samples containing formats slightly varying from the logs present in the training set, the model successfully captures the fields from it. For example, regex-based parsers fail to capture multi-word usernames when they expect a single word user. If a regex-based parser is designed to parse a log,
Feb 12 19:29:32 nssal-ps3 sshd[5713]: Failed password for invalid user brenda from 208.87.243.236 port 49922 ssh2

the system might fail working on the log,
Feb 12 19:29:32 nssal-ps3 sshd[5713]: Failed password for invalid user brenda jones from 208.87.243.236 port 49922 ssh2

because of an extra word in the username. However, such types of inconsistencies are found to be overcome through the model since it can adjust minor syntactical changes that occur with different version of Unix logs or with slight difference in the logging format.

5. Conclusion

This research work has made use of deep learning for named entity recognition in Unix raw logs to extract useful information in the logs. The basic pipeline steps required to process the raw logs has been explored including signature generation for commonly occuring patterns. Thus this research work showed that one of the deep learning methods BLSTM can be used to predict named entities from logs just like with natural language leaving space for further research.

With these contributions, this project has shown high hopes for Natural Language Processing approaches in an automated system that could perform an automated field extraction from Unix logs. Bringing these kinds of implementation from concept to real implementation in the cyber security industries will need the cooperation of all stakeholders, including researchers from academia and industry experts. Nevertheless, a lot of improvement has to be made to develop the automated technology worth being used in the practical cyber-security industry since false negatives can get critical security threats go unnoticed leading to catastrophic consequences.

6. Future Enhancements

This work has made use of Natural Language Processing for log parsing. A remarkable enhancement in this work would be to have a system that would make use of a hybrid approach, i.e. an approach that makes use of both: field parsing through Named Entity Recognition and log clustering through some clustering algorithm. Log clustering can group log messages into clusters thereby providing large-scale analytics from logs whereas log parsing through named-entity recognition or any other algorithm provides fine grain details from the logs.

If an anomalous event, for example, an event concerning multiple authentication failures from the same user within a short time span takes place, a log clustering system would detect the event as an outlier. However, we would require a log parsing system to find out which user's account was trying to be compromised. Thus, since the two systems are complementary to each other, a hybrid approach would be considered more complete. This can be one of the future directions of the work.

Acknowledgments

The authors would like to extend their sincere thanks to Logpoint for providing the logs along with a SIEM tool that aided in creating training and testing set though label generation.

References

[1] Casey Lorenzen, Rajeev Agrawal, and Jason King. Determining viability of deep learning on cybersecurity log analytics. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 4806–4811. IEEE, 2018.

[2] Jieming Zhu, Shilin He, Jinyang Liu, Pinjia He, Qi Xie, Zibin Zheng, and Michael R Lyu. Tools and benchmarks for automated log parsing. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pages 121–130. IEEE, 2019.

[3] Hossein Hamooni, Biplob Debnath, Jianwu Xu, Hui Zhang, Guofei Jiang, and Abdullah Mueen. Logmine: Fast pattern recognition for log analytics. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 1573–1582, 2016.

[4] Basanta Joshi, Umanga Bista, and Manoj Ghimire. Intelligent clustering scheme for log data streams. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 454–465. Springer, 2014.

[5] Pinjia He, Jieming Zhu, Zibin Zheng, and Michael R Lyu. Drain: An online log parsing approach with fixed depth tree. In *2017 IEEE international conference on web services (ICWS)*, pages 33–40. IEEE, 2017.

[6] Hetong Dai, Heng Li, Che Shao Chen, Weiyi Shang, and Tse-Hsun Chen. Logram: Efficient log parsing using n-gram dictionaries. *IEEE Transactions on Software Engineering*, 2020.

[7] Prabhat Pokharel. Information extraction using named entity recognition from log messages. *Masters thesis*, 2018.

[8] Jason PC Chiu and Eric Nichols. Named entity recognition with bidirectional lstm-cnns. *Transactions of the association for computational linguistics*, 4:357–370, 2016.

[9] Shilin He, Jieming Zhu, Pinjia He, and Michael R Lyu. Loghub: a large collection of system log datasets towards automated log analytics. *arXiv preprint arXiv:2008.06448*, 2020.

[10] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.