IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON DEEP LEARNING: SECURITY AND FORENSICS RESEARCH ADVANCES AND CHALLENGES

Received March 5, 2020, accepted March 23, 2020, date of publication March 26, 2020, date of current version April 9, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2983435

Sentiment Analysis in a Forensic Timeline With Deep Learning

HUDAN STUDIAWAN[1,2], FERDOUS SOHEL[1], (Senior Member, IEEE), AND CHRISTIAN PAYNE[1]

[1]College of Science, Health, Engineering, and Education, Murdoch University, Perth, WA 6150, Australia
[2]Department of Informatics, Institut Teknologi Sepuluh Nopember, Surabaya 60111, Indonesia

Corresponding author: Hudan Studiawan (hudan.studiawan@murdoch.edu.au)

This work was supported by the Indonesia Lecturer Scholarship (BUDI) from the Indonesia Endowment Fund for Education (LPDP). This scholarship is a collaboration between the Ministry of Finance and Ministry of Research, Technology, and Higher Education of the Republic of Indonesia.

ABSTRACT A forensic investigator creates a timeline from a forensic disk image after an occurrence of a security incident. This procedure aims to acquire the time for all events identified from the investigated artifacts. An investigator usually looks for events of interest by manually searching the timeline. One of the sources from which to build a timeline is log files, and these events are often found in log messages. In this paper, we propose a sentiment analysis technique to automatically extract events of interest from log messages in the forensic timeline. We use a deep learning technique with a context and content attention model to identify aspect terms and the corresponding sentiments in the forensic timeline. Terms with negative sentiments indicate events of interest and are highlighted in the timeline. Therefore, the investigator can quickly examine the events and other activities recorded within the surrounding time frame. Experimental results on four public forensic case studies show that the proposed method achieves 98.43% and 99.64% for the F1 score and accuracy, respectively.

INDEX TERMS Forensic timeline, deep learning, context attention, content attention, sentiment analysis, event logs.

I. INTRODUCTION

Constructing a timeline of events is one of the most important steps in analyzing a piece of digital evidence [1]. For example, an investigator needs to build a forensic timeline from a forensic disk image right after a cyber security attack. The timestamps can be extracted from file systems and various log files. It is very critical for an investigator to understand when a suspicious event occurred and the activities surrounding that event [2].

The forensic timeline provides a general overview of the activities that occurred before, during, and after a particular security incident [3]. In addition, timeline construction is a part of event reconstruction. Event reconstruction is defined as processing a set of events and constructing a timeline of the events related to a forensic case [3].

An event of interest can be viewed as an anomaly in a forensic timeline. For instance, the investigator can analyze outlier files in a directory from digital evidence [4]. A new file created by an attacker is defined as an outlier in an investigated computer directory. The iterative z algorithm [5] has been used to detect this type of outlier. Marrington *et al.* [6] investigated temporal inconsistency in event log data from the Windows operating system. The timeline was constructed based on the Lamport relation, and the method determines out-of-sequence and missing events.

The investigator needs to check the forensic timeline for suspicious events. Existing methods need a manual search [2] or require inputting keywords that have a high likelihood of occurring [1]. The use of predefined rules does not provide flexibility because previously unseen messages may be recorded in log files. For an automatic process, there are several methods for generating a forensic timeline [8], [9] but not for finding events of interest in a timeline.

Furthermore, to support statistics-based anomaly detection for forensic purposes, natural language processing approaches, such as sentiment analysis, are needed [10]. On the other hand, an investigator can manually identify the

The associate editor coordinating the review of this manuscript and approving it for publication was Zhen Qin.

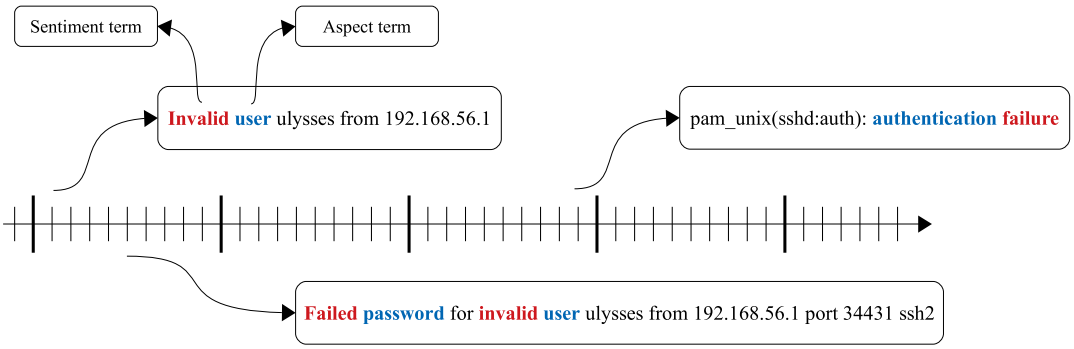

FIGURE 1. An illustration of aspect-based sentiment analysis in a forensic timeline. The log messages are extracted from an authentication log file [7].

events of interest in a forensic timeline. However, this is time-consuming; therefore, automatic detection of the events of interest is needed [11].

As one of the primary sources used to build a forensic timeline, the messages in event logs offer useful information about events and contain both positive and negative sentiments. The basic idea of this work is that the investigator can spot any event of interest from negative log messages because log files contain huge entries to analyze. We do not intend to remove the positive messages from the investigation context. After the investigator spots log messages with negative sentiments, the investigator can further examine the surrounding events described in the log messages.

As an illustration, there are log messages containing negative sentiments such as ''failed password'' or ''authentication failure'', as shown in Figure 1. The aspect term is shown in blue, while the sentiment term is indicated in red. The aspect term is the entity or subject being discussed in a log message. This type of timeline can be generated from a forensic disk image using the log2timeline tool [12].

Therefore, it is advantageous to examine the log files using aspect-based sentiment analysis. Unlike document-level or sentence-level sentiment analysis, aspect-based sentiment analysis aims to detect polarity in each aspect being examined [13]. This level of granularity enables us to extract more detailed information from log messages in the investigated forensic disk. From Figure 1, we can see that the investigator is better able to analyze a forensic timeline where the events of interest are highlighted through sentiment analysis results, especially the negative sentiments.

Aspect-level sentiment analysis involves two main steps: extracting the aspect from text data and then identifying the sentiment for each aspect, specifically positive or negative [13]. The most common data to be examined for its sentiment are social media data [14] and customer product reviews [15]. By analyzing sentiment, we can determine the users' and customers' preferences. In our case, the event logs can contain suspicious activities such as attempts to gain unauthorized access to computers, as illustrated in Figure 1. On the other hand, a normal system or normal user activities are viewed as positive sentiments.

The use of deep learning for forensic purposes was suggested in [11]. On the other hand, deep-learning-based techniques have been used to detect aspect-based sentiment [13], [16]. They also provide better performance compared to traditional machine learning methods. Specifically, we employ attention-based deep learning for sentiment detection [17]. We use two types of attention, namely, context attention and content attention, to achieve high accuracy.

Overall, the contributions of this paper are as follows.

1) This work proposes using aspect-based sentiment analysis in a forensic timeline. The events of interest are modeled as negative sentiments.
2) We use attention-based deep learning to detect sentiment from log messages, which are then displayed in a forensic timeline. This assists the forensic investigator in examining a timeline containing various messages.
3) Once the sentiment model is trained, it can be used to detect the events of interest in a forensic timeline automatically, and no further input is required.
4) The proposed method achieves very high performance, indicated by 98.43% and 99.64% for the F1 score and accuracy, respectively.

The organization of this paper is as follows. Section II covers the related research on forensic timeline analysis, sentiment analysis of event logs, and anomaly detection in a forensic timeline. Section III describes the main pipeline of the proposed method. The experimental results and analysis are given in Section IV. In Section V, we give the conclusions and future work of this paper.

II. RELATED WORK

This section reviews the work related to forensic timeline analysis and sentiment analysis in event logs. Subsequently, we explain anomaly detection in a forensic timeline. We discuss anomaly detection because the negative sentiment found in log messages can be considered an anomaly or an event of interest as described in this paper.

A. FORENSIC TIMELINE ANALYSIS

One popular tool used to generate a forensic timeline is log2timeline [2], [12]. It creates a comprehensive timeline

from various log sources, such as browser logs and operating system logs. log2timeline produces a CSV file, which can be examined or visualized further by other forensic tools.

A recent tool to analyze a CSV file from log2timeline is Timeline2GUI [1]. It offers a graphical user interface (GUI) to investigate the timeline. Timeline2GUI is equipped with a search box enabling an investigator to look for important events based on a particular keyword. It can filter the events inside the CSV file generated from log2timeline with Timeline2GUI or with log2timeline itself.

Moreover, Timeline2GUI has two views, namely, a *reduced view* to obtain a general overview of all events and a *detailed view* for more detailed records of activity. It also uses a color highlighting to inform the investigator about events, such as red for program execution.

To build a forensic timeline, one can use other commercial tools such as EnCase and FTK, as demonstrated in [18]. As highlighted in [19], the creation of an event timeline from various artifacts in an operating system is a critical task for forensic investigation.

Esposito and Peterson [19] showed the analysis of a CSV output file from log2timeline with an SQL query. First, the CSV file is imported to a Microsoft Access database. Then, there are several queries for examining important events, namely, application activity, browsing history, access to recent documents, and executed programs [19]. A forensic timeline is also needed after events are modeled with semantic-based correlation [3] or ontology-based techniques [9] to investigate the association between events and their timeline.

There are several other tools to build a forensic timeline and its visualization, such as Zeitline [20] and CyberForensic TimeLab [21]. However, these tools are no longer updated, and more recent tools for visualization, such as Timeline2GUI [1], are available to practitioners and the digital forensic community.

B. SENTIMENT ANALYSIS IN EVENT LOGS

Since sentiment analysis is popularly applied to social media or product review data, we can apply it to log messages from an event log file. In software engineering, we can identify developer emotions from commit logs on the GitHub repository by using sentiment analysis [22]. This work uses the SentiStrength method, which is built based on a dictionary of sentiment words and a machine learning technique [23]. Guzman *et al.* [22] concluded that a software project with a larger number of developers tends to have more positive emotion polarity.

Other work on software commit logs indicated that a large number of changes in source code files resulted in negative sentiment in the commit messages [24]. The SentiStrength method is also used in this work [24]. In addition, these works [22], [24] concluded that the time to commit the source code to the repository would have negative sentiments when executed at the beginning of the week, specifically on Mondays or Tuesdays.

These aforementioned works attempted to use sentiment analysis in event log data. However, none of them have been applied to operating system logs for forensic purposes. To build a forensic timeline, we propose applying deep-learning-based sentiment analysis to system logs, which are extracted from a forensic disk image. Negative sentiments indicate an anomaly or an event of interest, which has to be investigated further.

C. ANOMALY DETECTION IN A FORENSIC TIMELINE

In terms of a temporal anomaly, Marrington *et al.* [6], [25] modeled Windows event logs based on the Lamport time relation. This technique can identify out-of-sequence and missing events. The advantage of this method is that it can run automatically to assist an investigation. However, it recognizes anomalies without correcting or giving a recommendation about the missing events.

The timestamp-based anomaly detection has also been developed in a virtual machine [26]. Similar to [6], [25], the authors used the Lamport model to detect anomalous events from virtual operating system logs. Another work proposed a temporal cross-reference in the file system and file metadata [27]. Different from existing methods, we propose a novel approach that involves using sentiment analysis to detect an anomaly or point of interest from system logs and then plotting them in a forensic timeline.

III. THE PROPOSED METHOD

A block diagram of the proposed method is shown in Figure 2. There are two main steps, specifically, a training step and an investigation step. In the training step, we first preprocess the log files to extract messages containing negative or positive sentiments. Second, we build word embeddings to represent text messages as a vector of numbers. Furthermore, there are two attention-based techniques in the main deep learning architecture, namely, a context attention layer and a content attention layer. We then use a softmax layer to determine whether a message sentiment is positive or negative. In the last step of the training phase, we save the sentiment model.

In the investigation step, we first build a forensic timeline from the forensic disk image. We then detect the sentiments using the model from the training step. Finally, the negative messages are displayed in a timeline to assist the forensic investigation. Each step is described in detail in the following subsections.

A. EVENT LOG PREPROCESSING

To obtain the sentiment of log messages, each log entry is split into separate entities. Several entities that are commonly found in a log entry include the timestamp, hostname, process name, and a message containing a short description about a particular event. An illustration of the entity parsing process for a log entry is depicted in Figure 3. At this stage, we extract all log messages from a log file to be analyzed for their sentiments.

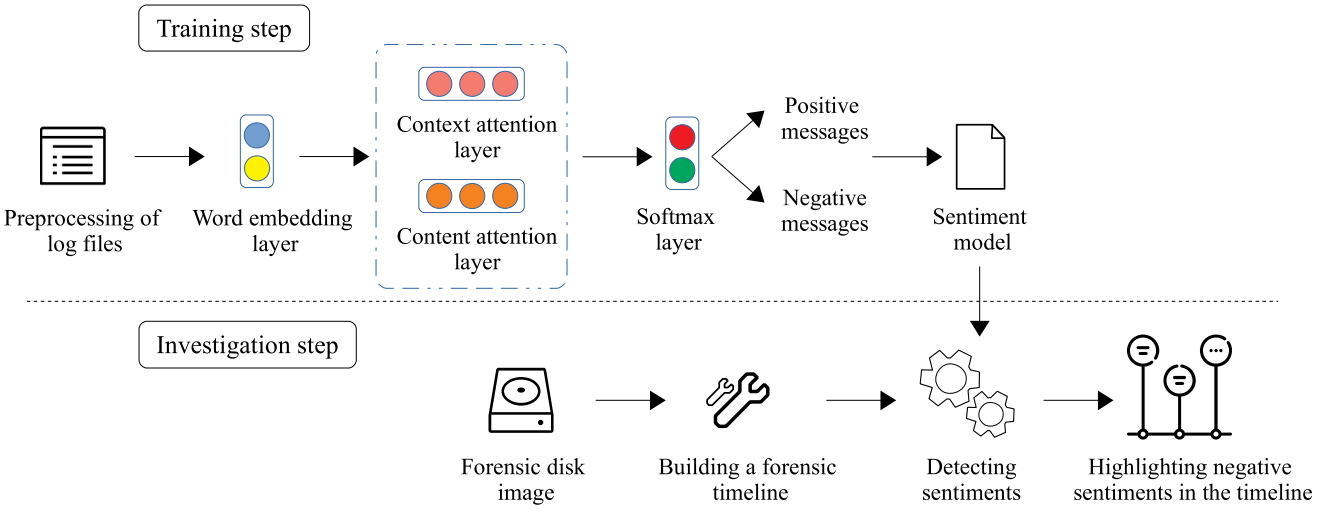

FIGURE 2. Proposed method for detecting aspect-based sentiment in a forensic timeline with deep learning.

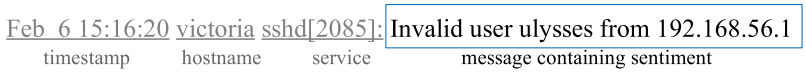

FIGURE 3. An illustration of parsing a log entry.

Unlike the existing approaches, which generally use regular expressions to parse log files, we employ the nerlogparser tool [28]. It can automatically split each entity in a log entry using a pretrained deep learning model, namely, bidirectional long short-term memory. The output of the nerlogparser is a JSON file containing the entity names and values for all records in a log file or a dictionary data structure. We use the latter because it will be fed to the word embedding as the next layer of the proposed method.

Finally, in this stage, we define the preprocessed log messages, which contain sentiments such that $\mathbf{P} = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_{|\mathbf{P}|}\}$ and $\mathbf{p}_i = \{w_{i1}, w_{i2}, \ldots, w_{i|\mathbf{p}_i|}\}$, where w_i is a single word in a message $\mathbf{p}_i$, $|\mathbf{P}|$ is the total length of log messages, and $|\mathbf{p}_i|$ is the length of a particular log message $\mathbf{p}_i$.

B. WORD EMBEDDING LAYER

Since deep learning architectures can process only numbers, we convert each word in preprocessed log messages to a vector of numbers. This conversion technique is commonly known as word embedding. The embedding is the first layer in our proposed attention-based deep learning method.

We use a pretrained word embedding, namely, GloVe version glove.42B.300 [29]. GloVe produces the embedding value for each word based on statistical properties such as word occurrence in a certain context or a particular sentence. The name of this version indicates that it has been trained with a total of 42 billion words of web text data from Common Crawl [30], and 300 is the dimension of the embedding.

The advantage of using a pretrained embedding model is that it can include a relationship between the words and the context in a sentence. Since it has been trained on 42 billion words of web text data and the log files also contain readable texts, GloVe is a reasonable representation for text messages from event log data. In other words, the embedding vector values from the training results can be applied to the log files. In this embedding layer, we look up each word from log messages in the GloVe embedding. If an unknown word is found in the preprocessed log messages, we replace it with a random floating value in the range $[-0.1, 0.1]$.

The word length of messages in log entries can vary. Therefore, we apply the padding and truncating technique to ensure that the length of messages is the same. We pad a message if it is shorter than the embedding size, and we truncate it if otherwise. If the messages are the same size, the model training can be conducted in batches, thereby making it faster.

Formally, we define a preprocessed log message $\mathbf{p}$ as a sentence $\mathbf{s} = \{w_1, w_2, \ldots, w_i, \ldots, w_{i+L}, \ldots, w_{|\mathbf{s}|}\}$, where $|\mathbf{s}|$ is the length of the sentence and w_i to w_{i+L} are the aspect terms, where L is the length of the aspect words. The deep learning model aims to identify the sentiment of these aspect terms.

The word embedding matrix, $\mathbf{E} = \{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_{|\mathbf{P}|}\}$, is extracted from GloVe, where $\mathbf{E} \in \mathbb{R}^{d \times |V|}$, V is the vocabulary, and d is the word vector dimension. The word embedding of w_i is denoted as $\mathbf{e}_i \in \mathbb{R}^{d \times 1}$. In other words, it is a column from the embedding matrix $\mathbf{E}$. Therefore, we obtain an embedding vector $\mathbf{e} = \{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_{|\mathbf{s}|}\}$ to represent a sentence $\mathbf{s}$.

C. CONTEXT ATTENTION LAYER

An attention model is intended to obtain valuable information on an aspect sentiment from a sentence. The first attention

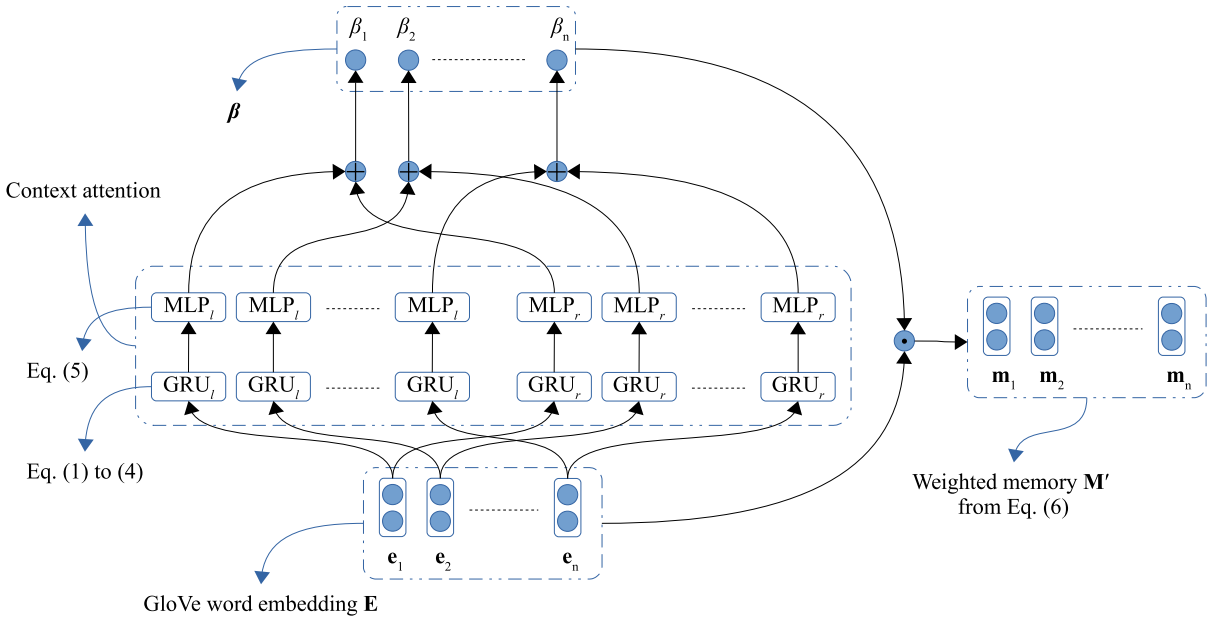

FIGURE 4. An illustration of the context attention procedure.

layer is the context attention. This layer considers three factors, namely, the word order, the aspect information, and the correlation between the word and the aspect [17].

To address the context attention, we use gated recurrent unit (GRU) networks [31] as the deep learning technique. Figure 4 depicts an illustration of the context attention procedure. The GRU learns the long-term dependencies in a sequence of words in a log message and the positional relationship between words in the whole log message. The GRU has two gates in its cell, namely, the reset gate and the update gate. The reset gate r is defined as:

$$r = \sigma(\mathbf{W}_r \mathbf{x}_t + \mathbf{U}_r \mathbf{h}_{t-1}), \tag{1}$$

where σ is the logistic sigmoid function, $\mathbf{x}$ is the input vector and $\mathbf{h}_{t-1}$ is the previous hidden state. Moreover, $\mathbf{W}_r$ and $\mathbf{U}_r$ are weight matrices to be optimized in the training step. The vector input $\mathbf{x}$ is the vector $\mathbf{E}$ from the word embedding layer. The reset gate manages the computation when the hidden state neglects the previous state and then adjusts with the current input. Moreover, the update gate z is denoted as:

$$z = \sigma(\mathbf{W}_z \mathbf{x}_t + \mathbf{U}_z \mathbf{h}_{t-1}), \tag{2}$$

where the notations are similar to the reset gate r. The update gate manages information from the previous state to the current one [31].

Furthermore, the activation unit h_t is defined by:

$$h_t = z \odot h_{t-1} + (1 - z) \odot \tilde{h}_t, \tag{3}$$

where $\tilde{h}_t$ is computed by:

$$\tilde{h}_t = \tanh(\mathbf{W}_{\tilde{h}_t} \mathbf{x}_t + \mathbf{U}_{\tilde{h}_t}(r \odot \mathbf{h}_{t-1})), \tag{4}$$

where $\odot$ is the element-wise product.

To capture the context between the aspect term and its surrounding words in a log message, we employ two GRU networks, namely, GRU_l for the left context and GRU_r for the right context. GRU_l processes the log messages from left to right or in a forward direction to learn the aspect term and its sentiment. On the other hand, GRU_r moves from right to left or in a backward direction. This procedure is illustrated in Figure 4, where the context of word embedding from Section III B are learned for the left and the right context. Finally, GRU_l provides hidden state vectors $\mathbf{H}_l = \{\mathbf{h}_{i+L_l}, \ldots, \mathbf{h}_{i_l}, \mathbf{h}_{i-1}, \ldots, \mathbf{h}_1\}$, while GRU_r generates $\mathbf{H}_r = \{\mathbf{h}_{i_r}, \ldots, \mathbf{h}_{i+L_r}, \mathbf{h}_{i+L+1}, \ldots, \mathbf{h}_N\}$, where N is the number of processed words.

Moreover, we use a multilayer perceptron (MLP) to compute the left context attention weight β_l of $\mathbf{h}_l$, which is denoted as:

$$\beta_l = \sigma(\mathbf{W}\mathbf{h}_l) + b_l, \tag{5}$$

where $\mathbf{W}$ is the weight matrix to be optimized and b_l is a basic attention weight. We run the same calculation for the right context attention weight β_r. To obtain the attention weights for all aspects $\boldsymbol{\beta}_a$, we calculate the mean of the left attention weight $\boldsymbol{\beta}_l$ and right attention weight $\boldsymbol{\beta}_r$.

To produce the weighted memory, which contains the context attention, we define the memory $\mathbf{M}$, which is built from a stack of embedding vectors $\mathbf{E}$. It is defined as $\mathbf{M} = \{\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_{|\mathbf{M}|}\}$, where $\mathbf{M} \in \mathbb{R}^{d \times N}$. Finally, the weighted memory $\mathbf{M}' = (\mathbf{m}_{w1}, \mathbf{m}_{w2}, \ldots, \mathbf{m}_{wN})$ is denoted as:

$$\mathbf{m}_{wi} = \boldsymbol{\beta} \odot \mathbf{m}_i, \tag{6}$$

where $\boldsymbol{\beta}$ is the concatenation of $\boldsymbol{\beta}_l$, $\boldsymbol{\beta}_r$, and $\boldsymbol{\beta}_a$. The weighted memory $\mathbf{M}'$ contains information about contexts in input log messages and is fed to the content attention layer.

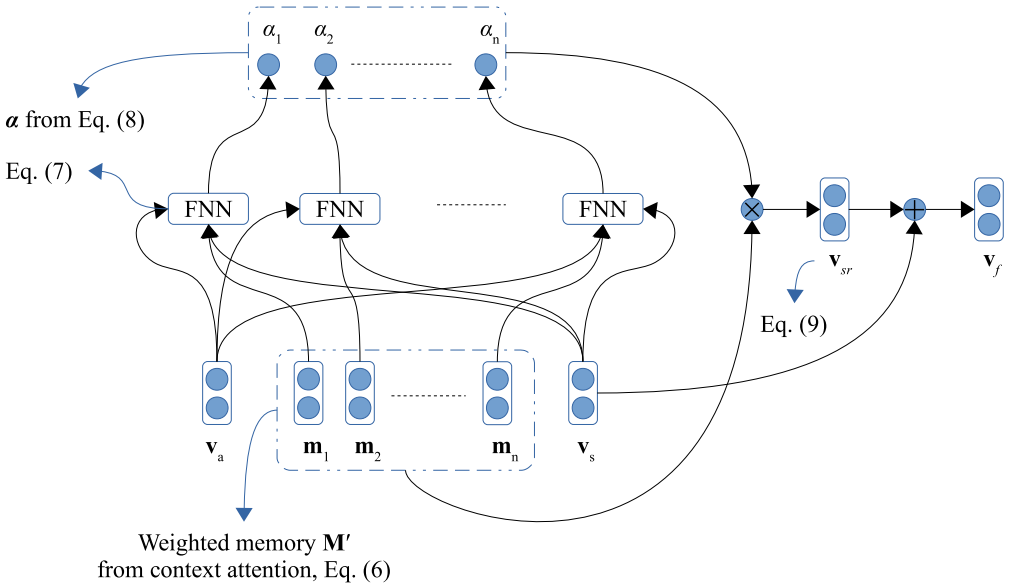

FIGURE 5. An illustration of the content attention procedure.

D. CONTENT ATTENTION LAYER

To more accurately detect the aspect sentiment, we use a second attention layer called the content attention layer that is used specifically for sentence representation. This representation improves the importance of a word for aspect sentiment within the whole content and enables us to capture the important sentiment features [17].

An illustrative example of content attention is shown in Figure 5 and explained in detail as follows. Note that in the bottom part of Figure 5, the weighted memory $\mathbf{M}'$ is generated from the previous procedure for the context attention. We define c_i as the attention weight of the memory slice $\mathbf{m}_i$. To calculate c_i, we use feedforward neural networks (FNNs) such that:

$$c_i = \mathbf{W} \tanh(\mathbf{W}\mathbf{m}_i + \mathbf{W}\mathbf{v}_a + \mathbf{W}\mathbf{v}_s), \tag{7}$$

where $\mathbf{m}_i$ is from $\mathbf{M}'$, $\mathbf{v}_a$ is the aspect representation, and $\mathbf{v}_s$ is the sentence representation. This c_i score shows the importance of word w_i to a sentiment aspect. After we calculate all weights $\mathbf{c} = \{c_1, c_2, \ldots, c_N\}$, the content attention weight α_i is defined by:

$$\alpha_i = \frac{\exp(c_i)}{\sum_{j=1}^{N} \exp(c_j)}. \tag{8}$$

At this stage, the attention weights produce an attention weight vector $\boldsymbol{\alpha} = \{\alpha_1, \alpha_2, \ldots, \alpha_N\}$ of the memory $\mathbf{M}'$. Note that the memory $\mathbf{M}'$ used in this layer is the weighted memory from the previous context attention layer. Furthermore, we can compute the aspects for sentence representation $\mathbf{v}_{sr}$:

$$\mathbf{v}_{sr} = \mathbf{M}'\boldsymbol{\alpha}, \tag{9}$$

where $\mathbf{v}_{sr} \in \mathbb{R}^d$. To improve the sentence representation to handle more complex log messages, we add the sentence representation from the embedding layer $\mathbf{v}_s$ and the sentence representation from the content attention layer $\mathbf{v}_{sr}$, resulting in $\mathbf{v}_f$. Finally, $\mathbf{v}_f$ is the final representation of the aspect, considering the context and sentiment information of the log messages from the previous context attention layer [17].

E. SOFTMAX LAYER

To build the deep learning model, we stack the embedding layer, content layer, and context attention layer. We use softmax as an output layer to identify the sentiment of each aspect found in the log messages. The softmax layer estimates a normalized distribution over two possible sentiments for each aspect term. The softmax is given by:

$$\text{softmax}(\mathbf{x}_i) = \frac{\exp^{x_i}}{\sum_{j=1}^{n} \exp^{x_j}}, \tag{10}$$

where $\mathbf{x}_i = \{x_1, x_2, \ldots, x_N\}$ is the vector output from the attention layer. In the model training step, we minimize the cross-entropy loss H, which is calculated by:

$$H(g, s) = -\sum_i g_i \, log(s_i), \tag{11}$$

where g is the ground-truth distribution and s is the estimated distribution from the softmax function.

F. BUILDING A FORENSIC TIMELINE

We use the log2timeline tool [12] to build a forensic timeline. Since we focus on sentiment analysis for operating system logs, we filter out other logs when constructing the timeline. The command for extracting the timeline including the log files is *log2timeline.py forensic-image.plaso forensic-image.dd*, where. plaso file is the plaso storage file format from log2timeline containing the extraction results from the forensic disk image. After that, we sort the timeline

TABLE 1. A list of public system log datasets used in this paper.

Identifier	Description	# files	# lines	# positive	# negative	# lines with aspect
Casper	ext3 disk image from Digital Corpora [32]	15	11,086	9,874	1,212	1,844
Jhuisi	jhuisi host of DFRWS Challenge 2009 [33]	25	11,737	9,063	2,674	1,727
Nssal	nssal host of DFRWS Challenge 2009 [33]	40	107,093	91,349	15,732	47,953
Honey	Linux image from Honeynet Challenge 7 [7]	12	8,712	8,162	550	1,426

chronologically by accessing the plaso storage format with the command psort: *psort.py -w forensic-sorted.csv forensic-image.plaso*. Note that the CSV format contains the sorted forensic timeline based on the timestamp found in the log files. The psort tool is also included in the log2timeline toolsets.

We cannot access the plaso storage format directly; therefore, we have to use the psort tool to extract the timeline. In our case, we filter the messages extracted by the log2timeline tool to obtain only the operating system logs because the messages inside these logs contain sentiments that can assist in the investigation process.

The CSV file generated by the log2timeline tool provides 17 predefined and fixed fields, including the date, time, source, type, and description. For sentiment analysis, we focus on the description field containing the main message and description of a particular event from the systems. The sentiment model produced by the proposed deep learning method is applied to this description field. Therefore, when building a forensic timeline, we do not use the nerlogparser tool to parse the log entries. The nerlogparser is used only for building the deep learning model in the training step. For timeline visualization, we use the Timesketch tool [34], which is described in the experiment section.

IV. EXPERIMENTAL RESULTS AND ANALYSIS

In this section, we describe the four public forensic datasets used to test the proposed method and five other methods. We then discuss how we built the ground truth for sentiment analysis in the event log data as the primary source of a forensic timeline. Subsequently, we explain the software and settings used when conducting the experiments. We also compare the proposed method with five other deep learning techniques. Finally, we demonstrate the sentiment analysis results when displayed in a forensic timeline.

A. PUBLIC FORENSIC DATASETS

We use four public forensic datasets to evaluate the proposed method and compare the results with those of five other techniques. These datasets are displayed in Table 1. These datasets are chosen because they contain various security incidents. Therefore, the operating system records several events, which are associated with these incidents. We extract these log messages and analyze their sentiments with the proposed method. The negative sentiments refer to negative events in the log messages. Note that system log files are one of the main sources used for building a forensic timeline.

The first dataset is nps-2009-casper-rw, a forensic image from a bootable USB device and provided by Digital Corpora [32]. The disk file system is ext3 from a Linux Ubuntu distribution. The main event in this disk is that the user browsed through several US government websites. It provides 15 system log files with 11,086 lines in total, 9,874 positive events, and 1,212 negative events.

The second and third datasets are from The Digital Forensic Research Workshop (DFRWS) Challenge 2009 [33]. This conference is one of the most respected venues in the digital forensics research area. The 2009 Challenge was related to a case study about an attacker who illegally transferred secret data. There were two Linux hosts involved in this case, namely, ''jhuisi'' and ''nssal''. These two hosts were Sony PlayStation 3 (PS3) devices. ''jhuisi'' has 25 system log files with 11,737 log entries, while ''nssal'' is the largest dataset, with 91,349 and 15,732 positive and negative entries, respectively.

The last datasets are extracted from Honeynet Forensic Challenge 7 2011 [7]. This dataset also provides a disk image of a compromised Linux server. From 12 files, there are 8,162 and 550 positive and negative activities, respectively. We recovered the directory `/var/log/` from these forensic disk images for all datasets. From this directory, we extract several standard log files found in a Linux distribution, such as authentication logs, kernel logs, and syslog. These files are the artifacts commonly analyzed for investigation.

B. BUILDING THE GROUND TRUTH FOR SENTIMENT ANALYSIS

We build a list of negative sentiment terms and a list of aspect terms from the datasets. We have an initial dictionary containing several negative words from the log messages, such as ''invalid'', ''failed'', ''disconnect'', ''error'' and ''failure''. Examples of aspect terms include ''password'', ''user'', ''port'', ''authentication'', and ''session''. We iterate through each entry in the log files in the datasets. After that, we extract the aspect term from a message. Subsequently, we determine whether a message contains negative terms from the predefined list. If so, the message and its respective aspect term are flagged as a negative sentiment. Otherwise, the message is defined as a positive sentiment.

When a negative log message does not contain any predefined keywords, we add the keyword in that message manually to the dictionary. All labeled datasets, the complete list of negative terms, and the aspect terms are provided in a GitHub repository.[1] In the present datasets, the total number of negative sentiment terms is 90, while that of the aspect terms is 74.

[1] https://github.com/studiawan/logaspect

```
<?xml version="1.0" ?>
<sentences>
        …
        <sentence id="1728">
                <text>Invalid user ulysses from 192.168.56.1</text>
                <aspectTerms>
                        <aspectTerm term="user" polarity="negative" from="8" to="12"/>
                </aspectTerms>
        </sentence>
        …
</sentences>
```

FIGURE 6. XML format for log datasets based on SemEval-2014 format [35].

The ground truth is formatted in XML, as this is the common data format used in aspect-based sentiment datasets. An example of the XML format we use in the log datasets is shown in Figure 6. The XML format of the log entry depicted in Figure 3 is shown in Figure 6. The XML tags are based on SemEval-2014 Task 4 [35] as the most popular datasets for aspect-based sentiment analysis. In Figure 6, the aspect term is ''user',' and the sentiment identified is negative, as term the ''Invalid'' exists in the predefined list of negative sentiments. This format also records the start and end indices of the aspect term in the log message.

C. EXPERIMENTAL SETTINGS

For this experiment, we use Python 3.6 to implement the proposed method. In addition, we use Keras 2.2.4 [36] for the deep learning library and TensorFlow 1.7.0 [37] as its back-end engine. We also use scikit-learn 0.21.2 [38] to calculate the evaluation metrics and split the training and testing data. For the hardware, we use a computer with a 12-core CPU, 64 GB of RAM, and a GeForce GTX 1080 Ti 12 GB GPU. The output of the training phase is a model file that can be used directly by a forensic investigator to analyze sentiments in a forensic timeline. We split all datasets with the following proportions: 60% for training, 20% for development, and 20% for testing. Note that these proportions are applied to the proposed and all compared methods.

Furthermore, we use the following hyperparameters for the proposed method. The dimension of the word embedding is 300, with GloVe as the pretrained embedding. The dropout value is 0.1, with a batch size of 32. Furthermore, we run the training with 50 epochs, with a learning rate of 0.001. We use Adam [39] as the learning optimizer in the proposed architecture. Adam is suitable for general cases because it is efficient, needs only a small amount of RAM, and is invariant to gradient scaling. The training is stopped early if there is no improvement in the model accuracy after five iterations to prevent model overfitting.

For evaluation metrics, we use the precision, recall, F1, and accuracy. The precision measures the method's ability to identify negative sentiments and not label them as positive, while recall shows the performance when detecting positive sentiments. Moreover, F1 is a weighted average of the precision and recall. Finally, the accuracy score provides an overall measurement of the detection results for all positive and negative sentiments. These four evaluation metrics are discussed in the next section.

D. COMPARISON WITH OTHER METHODS

To demonstrate the superior performance of the proposed method, we compare the proposed method with nine other deep learning techniques: BRNN (bidirectional recurrent neural network) [40], BGRU (bidirectional gated recurrent unit) [41], BLSTM + CRF (bidirectional long short-term memory and conditional random field) [42], TD-LSTM (target-dependent long short-term memory) [43], and MemNet (deep memory network) [44]. Note that TD-LSTM and MemNet are attention-based deep learning but use only one attention model.

In addition, we compare the proposed method with other aspect-based sentiment detection methods, namely, the attention network (AN), word-aspect attention network (WAAN), lexicon-aware word-aspect attention network (LWAAN), and interactive lexicon-aware word-aspect attention network (ILWAAN) [45]. These methods are based on the attention neural networks and incorporate sentiment lexicon information.

The metric values in the experimental results were from the testing dataset. The experimental results for each dataset when analyzed with the aforementioned methods and the proposed method are shown in Table 2 for the Casper dataset, Table 3 for the Jhuisi dataset, Table 4 for the Nsaal dataset, and Table 5 for the Honey dataset. These methods have been used to detect sentiment in other datasets, such as social media or product reviews. However, in these experiments, we evaluate these methods using the system log data. Note that the system log data are one of the main sources of data when constructing a forensic timeline.

When applied to the four datasets, the proposed method achieved the best performance in detecting sentiments, as indicated by the best mean F1 and accuracy values, with 98.430% and 99.635%, respectively. This superior performance is mostly due to the use of two attention layers in the deep learning model. The difference in the performance between the BRNN and BGRU is not significant because the

TABLE 2. Comparison of the proposed method with other methods on the Casper dataset (%).

Method	Precision	Recall	F1	Accuracy
BRNN [40]	91.827	89.531	90.665	97.466
BGRU [41]	91.918	90.625	91.267	97.570
BLSTM + CRF [42]	86.474	88.906	87.673	97.027
TD-LSTM [43]	96.124	94.714	95.406	98.684
MemNet [44]	97.297	66.667	73.611	94.737
AN [45]	94.365	94.000	92.094	94.000
WAAN [45]	94.809	94.500	92.987	94.500
LWAAN [45]	95.256	94.500	93.224	94.500
ILWAAN [45]	99.202	90.000	94.042	98.500
Proposed method	**99.575**	**95.000**	**97.155**	**99.211**

TABLE 3. Comparison of the proposed method with other methods on the Jhuisi dataset (%).

Method	Precision	Recall	F1	Accuracy
BRNN [40]	96.203	94.508	95.348	97.875
BGRU [41]	95.698	94.508	95.099	97.738
BLSTM + CRF [42]	94.861	95.648	95.253	97.578
TD-LSTM [43]	98.966	93.023	95.727	98.165
MemNet [44]	98.797	91.860	94.961	97.859
AN [45]	96.769	96.750	96.570	96.750
WAAN [45]	97.211	97.250	97.158	97.250
LWAAN [45]	91.428	90.500	87.736	90.500
ILWAAN [45]	99.160	93.878	96.315	98.500
Proposed method	**99.825**	**98.837**	**99.324**	**99.694**

TABLE 4. Comparison of the proposed method with other methods on the Nssal dataset (%).

Method	Precision	Recall	F1	Accuracy
BRNN [40]	98.765	98.007	98.384	99.315
BGRU [41]	99.335	99.269	99.302	99.710
BLSTM + CRF [42]	99.431	99.417	99.424	99.733
TD-LSTM [43]	99.703	99.439	99.570	99.770
MemNet [44]	98.128	99.317	98.711	99.298
AN [45]	98.684	98.688	98.672	98.688
WAAN [45]	98.996	99.000	98.993	99.000
LWAAN [45]	99.041	99.042	99.033	99.042
ILWAAN [45]	99.874	99.484	99.677	99.833
Proposed method	**99.988**	**99.935**	**99.961**	**99.979**

TABLE 5. Comparison of the proposed method with other methods on the Honey dataset (%).

Method	Precision	Recall	F1	Accuracy
BRNN [40]	92.657	91.471	92.060	98.302
BGRU [41]	94.577	92.964	93.763	98.475
BLSTM + CRF [42]	95.484	94.670	95.075	98.961
TD-LSTM [43]	99.474	85.000	90.912	98.973
MemNet [44]	81.963	94.113	86.964	97.945
AN [45]	99.048	84.333	91.081	84.333
WAAN [45]	93.356	94.000	93.677	94.000
LWAAN [45]	94.994	94.667	94.830	94.667
ILWAAN [45]	90.736	94.655	92.598	99.000
Proposed method	**99.823**	**95.000**	**97.280**	**99.658**

GRU represents an improvement over the RNN. For example, it achieved 97.466% and 97.570% accuracy for the BRNN and BGRU on the Casper dataset, as depicted in Table 2. The bidirectional approach captures only the left and right context of each word in a log message. These two directions provide information about the sequence of words in a log message. In addition, the bidirectional architecture represents each word and captures the context from the surrounding words in a log entry.

BLSTM + CRF is slightly better than the BRNN and BGRU because it uses a CRF as the last layer. LSTM is also an improved version of the RNN. The CRF considers the most optimal path in the neural networks through all possible label sequences. It assumes that each aspect is dependent on each other. Therefore, it provides an increase in accuracy compared to the BRNN and BGRU. In Table 4 and Table 5, BLSTM + CRF produces the highest F1 score compared to that of the BRNN and BGRU, with 99.424% and 95.075%, respectively. However, the BRNN, BGRU, and BLSTM+CRF take into account only the left and right context. In the proposed method, we use two types of attention that consider both content and context to achieve better accuracy.

TD-LSTM performs quite well on the Casper dataset, as indicated by the second-best F1 score of 95.406% (Table 2); it also achieves the third-best score on the Nssal dataset, with 99.570% (Table 4). TD-LSTM models the left and right contexts with the targets. In other words, TD-LSTM models the previous and following contexts as aspect-dependent features and can capture the interactions between aspects and contexts in the log message data. However, the recall and F1 score on the Honey dataset are lower than those of the other methods, with 85.000% and 90.912%, respectively. The reason is that the majority of sentiments in this dataset are negative. TD-LSTM is unable to identify them because targets are represented twice and are focused specifically on the second representation.

For the Casper and Honey datasets, MemNet demonstrated the poorest performance compared to the other methods. For instance, it shows an F1 score of 73.611% (Table 2) and 86.964% (Table 5). This is because it does not use a recurrent neural network model; thus, it is unable to capture the context between words. It uses multi-hop attention, which is not really applicable to log message data. In addition, MemNet builds the knowledge directory based on the embedding vectors of individual words [44]. Therefore, it is harder to learn the aspect terms provided in more complicated contexts.

The AN, WAAN, and LWAAN are simpler variants of the ILWAAN; thus, they are similar in terms of their deep learning architectures [45]. Therefore, they show similar performances on all datasets. The AN is the baseline model; it builds an aspect and its context separately using LSTM networks. The WAAN does not incorporate sentiment lexicon information. The LWAAN is different when calculating aggregation-level attention. Finally, the ILWAAN is a complete model that includes multiple attention mechanisms and sentiment lexicon information. Note that all of the evaluation results are very high, as shown in Table 2 to Table 5. This is because the log datasets are not as complex and complicated as regular text data. Log messages tend to only be short and consist only of several words or phrases.

On all evaluation metrics, the proposed method achieves the best value. The proposed method performs well, with overall F1 and accuracy scores of 98.43% and 99.64%, respectively. The reason is that we use two attention

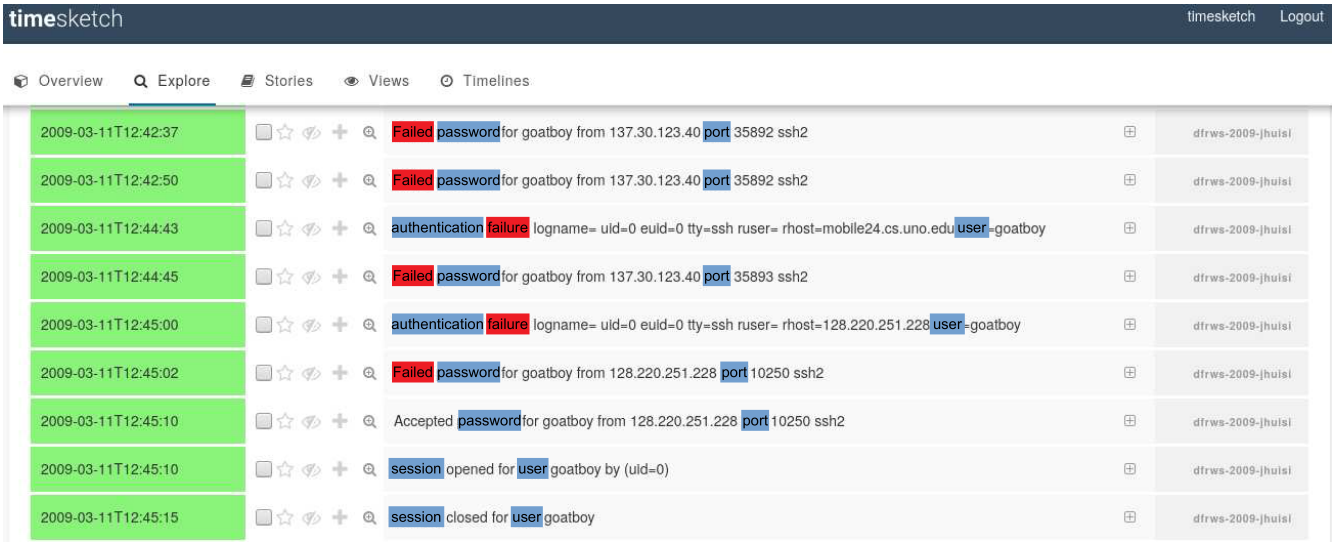

FIGURE 7. A plot of the sentiment analysis results to a Timesketch forensic timeline from an auth.log [33]. The investigator can easily spot the events of interest.

techniques to model aspect-based sentiment analysis. The context attention can capture the word position and information on the context and aspect. In addition, this attention also considers the correlations between each aspect in a sentence. The second attention, which addresses content, is able to represent the sentence well. Based on the conducted experiments, it has been proven that these two attention models can increase the performance of the aspect-based sentiment analysis in terms of model accuracy.

E. DISPLAYING NEGATIVE SENTIMENTS ON A FORENSIC TIMELINE

After extracting system logs from a forensic disk image using the log2timeline tool, we run the proposed method for sentiment analysis. In this investigation step, we detect the sentiments using the model from the training step. Finally, the negative messages are displayed in a timeline to assist the forensic investigation. We use the Timesketch tool [34] to plot negative sentiments on a forensic timeline. We use Timesketch because it is the latest and most improved application for displaying the activity timeline. It is a forensic tool that allows the investigative examination of a timeline. As shown in Figure 7, we use tabular-based views to display the negative sentiments of log messages. Blue denotes the aspect word, while red indicates its sentiment.

An example visualization of the forensic timeline is shown in Figure 7 with a portion of the case of DFRWS 2009. In the DFRWS 2009 case, a user is suspected of transferring an illegal file, which is a drug recipe in this case. Based on the scenario solution provided in [33], the user ''goatboy'' is responsible for this recipe file of illegal drug recipes in exchange for a subscription to Mardi Gras images and a video library. This type of view shown in Figure 7 gives a forensic investigator a general indication of the timeline before they examine more specific surrounding events.

A portion of the forensic timeline for the auth.log file from the DFRWS 2009 jhuisi host [33] is shown in Figure 7. The forensic investigator can easily distinguish the events of interest from among all events, as they are highlighted. In Figure 7, the events of interest are ''authentication failure'' and ''failed password'' for the user ''goatboy'', which have negative sentiments. The investigator can then examine these events and their surrounding events to identify possible incidents caused by an attacker.

V. CONCLUSIONS AND FUTURE WORK

In conclusion, we propose an aspect-based sentiment analysis to identify the events of interest in a forensic timeline. The sentiment analysis is applied to log messages as one of the primary sources to construct a forensic timeline. We apply the attention-based deep learning method, an architecture with two types of attention, namely, context and content attention. We finally plot the sentiment analysis results to a forensic timeline using the Timesketch tool. This plot is easy to navigate, as it is based on a web-based interface. The proposed method achieves 98.43% and 99.64% for the F1 score and accuracy, respectively, when evaluated on four public forensic datasets.

In the future, we will combine the sentiment-based approach for a forensic timeline with statistics-based anomaly detection [10]. This combination is expected to provide even greater assistance to investigators in discovering events of interest or anomalies in a forensic timeline. The sentiment analysis proposed in this paper can also be applied to a forensic timeline visualization tool such as Timeline2GUI [1]. For instance, suspicious events appear in red in the detailed view of Timeline2GUI so that the investigator can quickly notice them. Finally, we also plan to apply the proposed method to log entries, which have been grouped automatically [46]. The results can help investigators in

analyzing log clustering results and in making sense of the events that have occurred after a particular security incident.

REFERENCES

[1] M. Debinski, F. Breitinger, and P. Mohan, "Timeline2GUI: A Log2Timeline CSV parser and training scenarios," *Digit. Invest.*, vol. 28, pp. 34–43, Mar. 2019.
[2] K. Guðjónsson, "Mastering the super timeline with log2timeline," Inf. Secur. Reading Room, SANS Inst., Bethesda, MD, USA, Tech. Rep., 2010.
[3] Y. Chabot, A. Bertaux, C. Nicolle, and M.-T. Kechadi, "A complete formalized knowledge representation model for advanced digital forensics timeline analysis," *Digit. Invest.*, vol. 11, pp. S95–S105, Aug. 2014.
[4] B. D. Carrier and E. H. Spafford, "Automated digital evidence target definition using outlier analysis and existing evidence," in *Proc. Digit. Forensic Res. Workshop*, 2005, pp. 1–10.
[5] C.-T. Lu, D. Chen, and Y. Kou, "Algorithms for spatial outlier detection," in *Proc. 3rd IEEE Int. Conf. Data Mining*, Nov. 2003, pp. 597–600.
[6] A. Marrington, I. Baggili, G. Mohay, and A. Clark, "CAT detect (computer activity timeline detection): A tool for detecting inconsistency in computer activity timelines," *Digit. Invest.*, vol. 8, pp. S52–S61, Aug. 2011.
[7] G. Arcas, H. Gonzales, and J. Cheng. (2011). *Challenge 7 of the Honeynet Project Forensic Challenge 2011—Forensic Analysis of a Compromised Server*. [Online]. Available: https://old.honeynet.org/challenges/2011_7_compromised_server
[8] C. Hargreaves and J. Patterson, "An automated timeline reconstruction approach for digital forensic investigations," *Digit. Invest.*, vol. 9, pp. S69–S79, Aug. 2012.
[9] Y. Chabot, A. Bertaux, C. Nicolle, and T. Kechadi, "Automatic timeline construction and analysis for computer forensics purposes," in *Proc. IEEE Joint Intell. Secur. Informat. Conf.*, Sep. 2014, pp. 276–279.
[10] H. Studiawan, C. Payne, and F. Sohel, "Graph clustering and anomaly detection of access control log for forensic purposes," *Digit. Invest.*, vol. 21, pp. 76–87, Jun. 2017.
[11] H. Studiawan, F. Sohel, and C. Payne, "A survey on forensic investigation of operating system logs," *Digit. Invest.*, vol. 29, pp. 1–20, Jun. 2019.
[12] J. Metz and K. Guðjónsson. (2019). *Log2Timeline Plaso: Super Timeline All the Things*. [Online]. Available: https://github.com/log2timeline/plaso
[13] J. Zhou, J. X. Huang, Q. Chen, Q. V. Hu, T. Wang, and L. He, "Deep learning for aspect-level sentiment classification: Survey, vision, and challenges," *IEEE Access*, vol. 7, pp. 78454–78483, 2019.
[14] L. Li, Y. Wu, Y. Zhang, and T. Zhao, "Time+User dual attention based sentiment prediction for multiple social network texts with time series," *IEEE Access*, vol. 7, pp. 17644–17653, 2019.
[15] S. Zhang and H. Zhong, "Mining users trust from E-Commerce reviews based on sentiment similarity analysis," *IEEE Access*, vol. 7, pp. 13523–13535, 2019.
[16] H. H. Do, P. Prasad, A. Maag, and A. Alsadoon, "Deep learning for aspect-based sentiment analysis: A comparative review," *Expert Syst. Appl.*, vol. 118, pp. 272–299, Mar. 2019.
[17] Q. Liu, H. Zhang, Y. Zeng, Z. Huang, and Z. Wu, "Content attention model for aspect based sentiment analysis," in *Proc. World Wide Web Conf. World Wide Web (WWW)*, 2018, pp. 1023–1032.
[18] C. Bryce, "Timeline creation and analysis guides," Senator Patrick Leahy Center Digit. Invest., Burlington, Vermont, Tech. Rep., 2013.
[19] S. Esposito and G. Peterson, "Creating super timelines in Windows investigations," in *Proc. 9th IFIP Int. Conf. Digit. Forensics*, 2013, pp. 135–144.
[20] F. Buchholz and C. Falk, "Design and implementation of Zeitline: A forensic timeline editor," in *Proc. Digit. Forensic Res. Conf.*, 2005, pp. 1–7.
[21] J. Olsson and M. Boldt, "Computer forensic timeline visualization tool," *Digit. Invest.*, vol. 6, pp. S78–S87, Sep. 2009.
[22] E. Guzman, D. Azócar, and Y. Li, "Sentiment analysis of commit comments in GitHub: An empirical study," in *Proc. 11th Work. Conf. Mining Softw. Repositories (MSR)*, 2014, pp. 352–355.
[23] M. Thelwall, K. Buckley, G. Paltoglou, D. Cai, and A. Kappas, "Sentiment strength detection in short informal text," *J. Amer. Soc. Inf. Sci. Technol.*, vol. 61, no. 12, pp. 2544–2558, Dec. 2010.
[24] V. Sinha, A. Lazar, and B. Sharif, "Analyzing developer sentiment in commit logs," in *Proc. 13th Int. Workshop Mining Softw. Repositories (MSR)*, 2016, pp. 520–523.
[25] A. Marrington, G. Mohay, A. Clark, and H. Morarji, "Dealing with temporal inconsistency in automated computer forensic profiling," Dept. Sci. Technol., Queensland Univ. Technol., Brisbane, QLD, Australia, Tech. Rep., 2009.
[26] S. Thorpe and I. Ray, "Detecting temporal inconsistency in virtual machine activity timelines," *J. Inf. Assurance Secur.*, vol. 7, no. 1, pp. 24–31, 2012.
[27] X. Ding and H. Zou, "Time based data forensic and cross-reference analysis," in *Proc. ACM Symp. Appl. Comput. (SAC)*, 2011, pp. 185–190.
[28] H. Studiawan, F. Sohel, and C. Payne, "Automatic log parser to support forensic analysis," in *Proc. 16th Austral. Digit. Forensics Conf.*, 2018, pp. 1–10.
[29] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 1532–1543.
[30] The Common Crawl Foundation. (2019). *Common Crawl*. [Online]. Available: http://commoncrawl.org/
[31] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN Encoder–Decoder for statistical machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 1724–1734.
[32] S. Garfinkel. (2009). *NPS-2009-Casper-RW: An Ext3 File System From a Bootable USB*. [Online]. Available: http://downloads.digitalcorpora.org/corpora/drives/nps-2009-casper-rw/
[33] E. Casey and G. G. Richard, III. (2009). *DFRWS Forensic Challenge 2009*. [Online]. Available: http://old.dfrws.org/2009/challenge/index.shtml
[34] J. Berggren. (2019). *Timesketch: Collaborative Forensic Timeline Analysis*. [Online]. Available: https://github.com/google/timesketch
[35] M. Pontiki, D. Galanis, J. Pavlopoulos, H. Papageorgiou, I. Androutsopoulos, and S. Manandhar, "SemEval-2014 task 4: Aspect based sentiment analysis," in *Proc. 8th Int. Workshop Semantic Eval. (SemEval)*, 2014, pp. 27–35.
[36] F. Chollet. (2015). *Keras*. [Online]. Available: https://keras.io
[37] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, M. Devin, J. Dean, S. Ghemawat, G. Irving, M. Isard, and M. Kudlur, "TensorFlow: A system for large-scale machine learning," in *Proc. 12th USENIX Conf. Operating Syst. Design Implement.*, 2016, pp. 265–283.
[38] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Oct. 2011.
[39] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2015, pp. 1–15.
[40] O. Irsoy and C. Cardie, "Opinion mining with deep recurrent neural networks," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 720–728.
[41] S. Jebbara and P. Cimiano, "Aspect-based sentiment analysis using a two-step neural network architecture," in *Proc. Semantic Web Challenges*, 2016, pp. 153–167.
[42] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, 2016, pp. 260–270.
[43] D. Tang, B. Qin, X. Feng, and T. Liu, "Effective LSTMs for target-dependent sentiment classification," in *Proc. 26th Int. Conf. Comput. Linguistics*, 2016, pp. 3298–3307.
[44] D. Tang, B. Qin, and T. Liu, "Aspect level sentiment classification with deep memory network," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2016, pp. 214–224.
[45] H.-T. Nguyen and L.-M. Nguyen, "ILWAANet: An interactive lexicon-aware word-aspect attention network for aspect-level sentiment classification on social networking," *Expert Syst. Appl.*, vol. 146, May 2020, Art. no. 113065.
[46] H. Studiawan, C. Payne, and F. Sohel, "Automatic graph-based clustering for security logs," in *Proc. Int. Conf. Adv. Inf. Netw. Appl.*, 2019, pp. 914–926.

HUDAN STUDIAWAN received the bachelor's and master's degrees from the Institut Teknologi Sepuluh Nopember, Indonesia, in 2009 and 2011, respectively. He is currently pursuing the Ph.D. degree with Murdoch University, Australia. His current research interests are digital forensics and machine learning.

FERDOUS SOHEL (Senior Member, IEEE) received the Ph.D. degree from Monash University, Australia.

He is currently an Associate Professor in information technology with Murdoch University, Australia. Prior to joining Murdoch University, in 2015, he was a Research Assistant Professor/Research Fellow of the School of Computer Science and Software Engineering, The University of Western Australia, from January 2008 to 2015. His research interests include computer vision, image processing, pattern recognition, multimodal biometrics, scene understanding, robotics, and video coding. He is a member of the Australian Computer Society. He is a recipient of the prestigious Discovery Early Career Research Award (DECRA) funded by the Australian Research Council, two WA-State-Government-funded competitive grants on shark hazard mitigation and digital pathology to improve cancer diagnosis, the VC Early Career Investigators Award (UWA), and the Best Ph.D. Thesis Medal from Monash University.

CHRISTIAN PAYNE received the Ph.D. degree from Murdoch University, Australia, in 2009. He is an Adjunct Lecturer with the School of Engineering and Information Technology, Murdoch University. He has previously taught in the areas of introductory programming and information security. His research interests include computer security, applied cryptography, and legal issues in technology.