(12)Indian Patent Application

(21) Application Number: 202141028707

(22) Filing Date: 25/06/2021 (43) Publication Date: 30/12/2022

(71) Applicant(s): L&T TECHNOLOGY SERVICES LIMITED

(72) Inventor(s): Madathil, Nithin
Khanum, Zaibanousheen
Chavan, Anusha
Suryanarayan, Surabhi Kudurgundi

(51) International Classifications: G06N 3/08 G06N 3/04 G06F 11/07 G06K 9/62 G06F 40/30

(54) Title: METHOD AND SYSTEM FOR TRAINING A MODEL TO DETECT ERRORS IN A LOG FILE

(57) Abstract: An Artificial Intelligence (AI) based method for training a model to detect errors in a log file of an application is disclosed. The method comprises selecting a plurality of log files pertaining to at least one application from one or more sources for error extraction, extracting a set of error logs from the plurality of log files pertaining to the at least one application, applying natural language processing (NLP) to the extracted set of errors and performing vectorization technique to convert the extracted set of errors into tokenized logs. The method further discloses clustering the tokenized logs to generate a feature table of errors for the at least one application. The feature table comprising a plurality of clusters along with respective label and each cluster comprising a list of errors and said feature table is used to test a new log file for error detection.

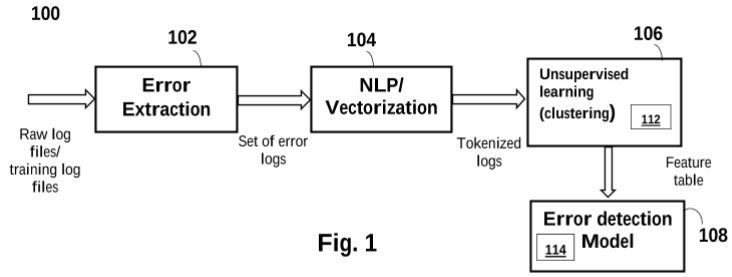

Fig. 1

FORM 2

THE PATENTS ACT 1970
(39 OF 1970)
&
The Patent Rules, 2003

Complete Specification

(See Section 10 and Rule 13)

1. TITLE OF THE INVENTION

METHOD AND SYSTEM FOR TRAINING A MODEL TO DETECT ERRORS IN A LOG FILE

2. APPLICANT(S)

(a) NAME : **L&T TECHNOLOGY SERVICES LIMITED**
(b) NATIONALITY : **INDIAN**
(c) ADDRESS : **DLF IT SEZ Park, 2nd Floor – Block 3**
1/124, Mount Poonamallee Road,
Ramapuram, Chennai – 600 089,
INDIA.

3. PREAMBLE TO THE DESCRIPTION

COMPLETE

The following specification describes the invention and the manner in which it is to be performed

DESCRIPTION

TECHNICAL FIELD

[0001] The present disclosure generally relates to the field of software testing. More specifically, it relates to training a model to detect errors from log files of an application.

BACKGROUND

[0002] There are numerous software testing tool available to detect any problem that occurs during coding and developing a new function or feature in a software product or in an application before it is delivered to the end users.

[0003] Precisely, to test any software application, using any of these tools, the automation scripts are executed on the application to generate the log files in the background of the application. Log file analysis is one of the most important aspects of performing effective and efficient software testing. At times when failures occur, developers/testers analyze these log files to know the error.

[0004] The defect triage procedure carried out in the software testing requires conducting a session with a triage team, which includes stakeholders like Product Manager, Testing Manager/Lead, Development Manager/Lead, and Business Analysts. The objective of this team is to evaluate the defects manually, assess them, and attach priorities and severity level.

[0005] Thus, the entire process of triaging is conducted based on human intelligence, thereby increasing the chances of missing out few errors in the log files. Further, the triaging process involve a lot of manual efforts that makes the process time consuming and expensive.

[0006] Therefore, there exists a need in the art to provide a method and a system which overcomes the above-mentioned problems by automatically testing the log files of an applications for errors, thereby reducing the dependency on human efforts and human intelligence.

OBJECTS OF THE PRESENT DISCLOSURE

[0007] Some of the objects of the present disclosure, which at least one embodiment herein satisfies are as listed herein below.

[0008] The primary object of the present disclosure is to train a model or a system to detect errors in a log file of an application.

[0009] Another object of the present disclosure is to completely automate the process of detecting errors in a log file of an application.

[0010] Yet another object of the present disclosure is to reduce the dependency of testing of application on human efforts and human intelligence, thereby eliminating triaging team.

[0011] Yet another object of the present invention is to provide a method and system that reduces the duplicate errors filed in defect tracking tool.

SUMMARY

[0012] The present disclosure overcomes one or more shortcomings of the prior art and provides additional advantages discussed throughout the present disclosure. Additional features and advantages are realized through the techniques of the present disclosure. Other embodiments and aspects of the disclosure are described in detail herein and are considered a part of the claimed disclosure.

[0013] In one non-limiting embodiment of the present disclosure, an Artificial Intelligence (AI) based method for training a model to detect errors in a log file of an application is disclosed. The method comprises selecting a plurality of log files pertaining to at least one application from one or more sources for error extraction and extracting a set of errors from the plurality of log files pertaining to the at least one application. Said method further discloses applying natural language processing (NLP) technique to the extracted set of errors to create an error file. The method further discloses assigning a token to each error, within the error file, and performing vectorization of the assigned tokens to convert, the errors within the error file, into tokenized logs. The method further discloses clustering the tokenized logs to generate a feature table of errors for the at least one application, the feature table comprising a plurality of clusters along with respective label and each cluster comprising a list of errors. Said method discloses that said feature table may be used to test a new log file for error detection.

[0014] In another non-limiting embodiment of the present disclosure, clustering the tokenized logs comprises defining the plurality of clusters from the tokenized logs based on a label associated

with the tokenized logs, wherein the label at least indicates a category of error and generating the feature table based on the plurality of cluster along with the respective label.

[0015] In yet another non-limiting embodiment of the present disclosure, for detecting the error in the new log file, the method discloses receiving the new log file for testing at least one application. The method further discloses applying NLP to the new log file, assigning tokens and performing vectorization of the at least one log file to generate the tokenized data. The method then moves to performing a look-up in the feature table for identifying at least one label for at least one error present in the tokenized data and displaying a list of errors corresponding to the at least one detected label.

[0016] In yet another non-limiting embodiment of the present disclosure, the NLP creates the error file by retaining only the entries relating to errors and removing remaining entries from the extracted set of errors.

[0017] In yet another non-limiting embodiment of the present disclosure, the method further comprises comparing the at least one identified label with a plurality of tickets and displaying at least one ticket matching the at least one detected label. In addition, the method discloses generating a new ticket, if the at least one label do not match the plurality of tickets.

[0018] In yet another non-limiting embodiment of the present disclosure, an Artificial Intelligence (AI) based system for training a model to detect errors in a log file of an application is disclosed. The system comprises one or more processors in communication with the memory and configured to select a plurality of log files pertaining to at least one application from one or more sources for error extraction. The one or more processors are further configured to extract a set of errors log from the plurality of log files pertaining to the at least one application. The one or more processors is further configured to apply natural language processing (NLP) technique to the extracted set of errors to create an error file. Furthermore, the one or more processors is configured assign a token to each error, within the error file, and perform vectorization of the tokens to convert, the errors within the error file, into tokenized logs. Finally, the one or more processor is configured to cluster the tokenized logs to generate a feature table of errors for the at least one application The feature table comprising a plurality of clusters along with respective label and each cluster comprising a list of errors and the feature table is used to test a new log file for error detection.

[0019] In yet another non-limiting embodiment of the present disclosure, to cluster the tokenized logs, the one or more processor is configured to define the plurality of clusters from the tokenized logs based on a label associated with the tokenized logs, wherein the label at least indicates a category of error. The one or more processors is further configured to generate the feature table based on the plurality of cluster along with the respective label.

[0020] In yet another non-limiting embodiment of the present disclosure, to detect errors in the new log file, the one or more processor is configured to receive the new log file for testing one of the at least one application. The one or more processor is further configured to apply NLP to the new log file, assign tokens and perform vectorization of the at least one log file to generate the tokenized data. Further, the one or more processor is configured to perform a look-up in the feature table to identify at least one label for at least one error present in the tokenized data and display a list of errors corresponding to the at least one detected label.

[0021] In yet another non-limiting embodiment of the present disclosure, the one or more processors is configured to use NLP to create the error file by retaining only the entries relating to errors and removing remaining entries from the extracted set of errors.

[0022] In yet another non-limiting embodiment of the present disclosure, the one or more processors is configured to compare the at least one identified label with a plurality of tickets and display at least one ticket matching the at least one detected label. The one or more processor is further configured to generate a new ticket if the at least one label is not matching the plurality of tickets.

[0023] The foregoing summary is illustrative only and is not intended to be in any way limiting. In addition to the illustrative aspects, embodiments, and features described above, further aspects, embodiments, and features will become apparent by reference to the drawings and the following detailed description.

BRIEF DESCRIPTION OF THE ACCOMPANYING DRAWINGS

[0024] The features, nature, and advantages of the present disclosure will become more apparent from the detailed description set forth below when taken in conjunction with the drawings in which like reference characters identify correspondingly throughout. Some embodiments of

system and/or methods in accordance with embodiments of the present subject matter are now described, by way of example only, and with reference to the accompanying figures, in which:

[0025] Fig. 1 shows an exemplary environment for training a model to test a log file of an application, in accordance with an embodiment of the present disclosure;

[0026] Fig. 2 shows an exemplary environment for detecting errors in a log file of an application, in accordance with an embodiment of the present disclosure;

[0027] Fig. 3 shows a flow chart illustrating an exemplary artificial intelligence (AI) based method for training a model to detect errors in a log file of an application, in accordance with an embodiment of the present disclosure;

[0028] Fig. 4 shows a flow chart illustrating an exemplary artificial intelligence (AI) based method for detecting errors in a log file of an application, in accordance with an embodiment of the present disclosure;

[0029] Fig. 5 shows a block diagram illustrating an artificial intelligence (AI) based system for training a model to detect errors in a log file of an application, in accordance with an embodiment of the present disclosure.

[0030] Fig's. 6(a) illustrate an exemplary raw log file, in accordance with an embodiment of the present disclosure.

[0031] Fig's 6(b)-6(f) exemplary output of blocks 102-106 disclosed in figure 1, in accordance with an embodiment of the present disclosure.

[0032] It should be appreciated by those skilled in the art that any block diagram herein represents conceptual views of illustrative systems embodying the principles of the present subject matter. Similarly, it will be appreciated that any flow charts, flow diagrams and the like represent various processes which may be substantially represented in computer readable medium and executed by a computer or processor, whether or not such computer or processor is explicitly shown.

DETAILED DESCRIPTION

[0033] The terms "comprise", "comprising", "include(s)", or any other variations thereof, are intended to cover a non-exclusive inclusion, such that a setup, system or method that comprises a

list of components or steps does not include only those components or steps but may include other components or steps not expressly listed or inherent to such setup or system or method. In other words, one or more elements in a system or apparatus proceeded by "comprises… a" does not, without more constraints, preclude the existence of other elements or additional elements in the system or apparatus.

[0034] In the following detailed description of the embodiments of the disclosure, reference is made to the accompanying drawings that form a part hereof, and in which are shown by way of illustration specific embodiments in which the disclosure may be practiced. These embodiments are described in sufficient detail to enable those skilled in the art to practice the disclosure, and it is to be understood that other embodiments may be utilized and that changes may be made without departing from the scope of the present disclosure. The following description is, therefore, not to be taken in a limiting sense.

[0035] Artificial Intelligence (AI) based method and AI based system for training a model to detect errors in a log file of an application is disclosed. The method comprises selecting a plurality of log files pertaining to at least one application from one or more sources for error extraction and extracting a set of errors from the plurality of log files pertaining to the at least one application by using a catalogue containing a list of pre-defined errors. The method further describes applying natural language processing (NLP) technique to the extracted set of errors to generate an error file. The method further includes assigning a token to each error, within the error file, and performing vectorization of the assigned tokens to convert the set of errors into tokenized logs. Finally, the method discloses clustering the tokenized logs to generate a feature table of errors for the at least one application. The feature table comprising a plurality of clusters along with respective label and each cluster comprising a list of errors, wherein said feature table is to test a new log file for error detection.

[0036] **Fig. 1** shows an exemplary environment 100 for training a model to test a log file of an application, in accordance with an embodiment of the present disclosure.

[0037] In an embodiment of the present disclosure, raw log files or training log files associated with or pertaining to one or more applications may be selected from various sources. The application may be a social media application such as Twitter, Instagram and Snapchat or to any of the OTT platform applications such as Netflix, Prime video, etc. It is to be noted said

application is not limited to above examples and any other application is well within the scope of the present disclosure.

[0038] The log files may be generated in background of an application by executing automatic scripts during the testing phase of an application. The log files may be present in any format known to a person skilled in the art. The log files may be stored locally or remotely. In one non-limiting embodiment of the present disclosure, the log files may be stored on a server and retrieved from the server for processing.

[0039] In an embodiment of the present disclosure, a log file may contain information about usage patterns, activities, and operations within the application. Further, the log file may contain one or more types of errors in it that are required to be identified and eliminated for smooth functioning of the application. Figure 6(a) shows an exemplary log file that is received as an input by the error extraction block 102 shown in figure 1. Those skilled in art will appreciate that said log file may include one or more errors. However, one such error i.e., *Android Runtime: FATAL EXCEPTION: main* is clearly indicated for ease of understanding in figure 6(a). Further, said log file may include other error, however the same are not shown to avoid confusion. Moreover, the log file shown in figure 6(a) is an exemplary log file shown for the purpose of explaining the invention and same shall not be construed limiting in any sense. Furthermore, as discussed above, the error extractor block 102 may be subjected, for error exctraction, to multiple such file at the same instance.

[0040] Moving further, the error extraction block 102 may be configured to apply an error extraction technique to the received log files to extract a set of errors from the log files. One or more known error extraction techniques may be used to remove unwanted data generated by executing automatic scripts on the application.

[0041] In one exemplary embodiment, the error extraction technique, applied by the error extraction block 102, may comprise parsing the log files and locating the set of errors by matching data present in the log files with a set of keywords and phrases. The matched set of key words and phrases may be used for further processing. In another exemplary embodiment, another technique, applied by the error extraction block 102, for extraction of errors from the log files may include using an error extractor (not shown) that contains a catalogue having an exhaustive list of pre-

defined errors relating to a plurality of application. Specifically, the received log files are parsed through the error extractor that may use the catalogue to extract errors contained therein.

[0042] However, the error extraction technique is not limited to above example and any other error extraction technique known to a person skilled in the art is well within the scope of the present disclosure. Figure 6(b) discloses output of an error extraction block (102) after performing error extraction process over a received log file as shown in figure 6(a). For example, figure 6(b) clearly shows how the error (i.e., *Android Runtime: FATAL EXCEPTION: main*) present in the log file shown in figure 6(a) is extracted by the error extraction block using one of the above discussed or other known techniques. Further, though not explicitly shown but the error extractor block 102 may include at least a processor for performing the operations such as error extraction.

[0043] Moving on, the extracted set of errors are then fed to a processing and vectorization block 104. The extracted set of errors received by the block 104 are processed using Natural Language Processing (NLP) technique to create an error file. Precisely, the NLP technique is applied to the extracted set of errors to remove stop words and clean the file with unwanted words. Those skilled in the art will appreciate that the extracted set of errors may also include entries other than errors, which may not be relevant for training of error detection model (discussed in detail in forthcoming paragraphs of the disclosure). Thus, such non-error entries are removed by applying NLP technique and a clean file, herein referred as the error file is created that only contains errors extracted from the received log files. Figure 6(c) discloses, by way of an example, an output of NLP block 104 reflecting an error file containing only the errors and not the other unwanted entries extracted, by the error extraction block 102, during the process of error extraction as shown in figure 6(b). Those skilled in the art will appreciate that the output of NLP block 104, as shown in figure 6(c), may include other errors in addition to *Android Runtime: FATAL EXCEPTION: main*, which may be extracted by the error extraction block 102 from same log file or other log files and the same is not explained for the sake of brevity.

[0044] In addition, block 104 may further be configured to assign a token to each error, within the error file, and perform vectorization of the assigned tokens to convert, the errors within the error file, into tokenized logs using one of the known vectorization techniques. In an aspect, in other words the NLP technique and vectorization technique may be used to process set of errors and

extract useful information such as keyword and phrases from the set of errors and then covert them into tokenized logs. In an exemplary embodiment, Figure 6(d) represents the output of the vectorization process performed at processing and vectorization block 104. As shown in figure 6(d) the processing and vectorization block 104 is configured to assign tokens to each error identified as shown in figure 6(c) and perform vectorization of the assigned tokens to convert, the errors within the error file, into tokenized logs. In an exemplary embodiment, the processing and vectorization block 104 may include one or more general purpose processor to perform the operations discussed in above paragraphs.

[0045] Moving ahead, as shown in Figure 1, the environment 100 may further include a unsupervised learning/training block 106. In an aspect, the unsupervised learning/training block 106 may comprise a dedicated processor 112. Said processor 112 of the unsupervised learning/training block 106 may be configured to apply an unsupervised learning technique to the tokenized logs. The unsupervised learning may include performing an iterative process and analyze tokenized logs without human intervention. The unsupervised learning involves implementation of machine learning for clustering and association.

[0046] In unsupervised learning/training block 106, the tokenized logs are processed, by the processor 112, to form a number of clusters as shown in figure 6(d) and each cluster is associated with a respective label, as shown in Figure 6(e). From figure 6(e) it is clear that each cluster comprises a list of errors and the label identifies a category or a type of error. For example, as shown in figure 6(e) the type of errors and the associated labels may be represented as label 0 representing network error, label 1 representing Apk not found error, label 2 representing App crash error, label 3 representing Playback error and label 4 may represent network interruption and so on . In an exemplary embodiment, the labels and associated errors shown in figure 6(e) are just for exemplary illustration to explain the invention and same shall not be construed limiting in any sense.

[0047] In the next block 108, i.e., error detection model, the clustering or categorization of the tokenized logs along with the respective label is done based on a reference table. In an aspect of the invention, the detection model 108 may also include a dedicated processor 114 to perform clustering or categorization of the tokenized logs along with the respective label. The reference table may comprise error syntax and its associated label. In an exemplary embodiment, each

cluster that is created after training is labelled with a name. The error detection model 108 handles the naming of clusters with set of defined names based on the application to which the log file relates. Further, the reference table may be used to map the tokenized logs against a respective type of error. The clusters and their respective labels are stored in a feature table, an example of which is shown in figure 6(f).

[0048] In an embodiment of the present disclosure, the error detection model 108 is trained with a large number of log files or training log files associated with numerous applications. The error detection model 108 may store the training data in the form of feature table. The error detection model 108 may use the feature table to detect a type or category of error in a new log file of the application to be tested. Thus, the training of the error detection model 108 with huge data set facilitates automating the process of detecting errors in a new log file and reducing the dependency of testing of application by human efforts and human intelligence.

[0049] In an embodiment, it is essential to understand that the blocks 102-108 disclosed in figure 1 are dedicated hardware units that may include one or more physical machine limitations such as error extractor, memory, processor etc. to carry out the desired objectives of the present invention.

[0050] **Fig. 2** shows an exemplary environment 200 for detecting errors in a log file of an application, in accordance with an embodiment of the present disclosure.

[0051] In an embodiment of the present disclosure, a new log file is tested for errors. The new log file may be converted into tokenized data using the NLP technique and the vectorization technique, by the processing and vectorizing block 104, discussed in above embodiments. The tokenized data may be fed to the error detection model 108 (discussed in disclosure of fig. 1 above) that comprises the feature table.

[0052] The detection model 108 uses the processor 114 The tokenized logs of each of the cluster in the feature table is compared with the tokenized data of the new log file to match the label for an error already present in the feature table with that of new log file. In view of said matching there are two possibilities. In first scenario, if a label corresponding to the error in the new log file is not detected in the feature table, it means that said error has not been identified in the log files during the training of the model. Thus, in said case the new log file may be forwarded to the

unsupervised learning block (discussed above in fig. 1) for training and generating an updated feature table.

[0053] In an exemplary aspect, once said log file is parsed through the model discussed in fig. 1, the error present in the new log file may form a new cluster in the feature table after application of unsupervised learning on the new log file. The updated feature table may facilitate detection of errors similar to the error present in the new log files generated in future.

[0054] In second scenario, if a label corresponding to the error in the new log file is detected, the detected label may be then forwarded to a ticketing tool 202. The ticketing tool may include a comparator 204 configured to compare the detected label with the plurality of tickets stored in memory 206 and display the ticket matching the detected label. In one non-limiting embodiment of the present disclosure, the ticketing tool may display the matched ticket with a comment.

[0055] In a further embodiment, if the label does not match the detected label, then a processor 208 of the ticketing tool 202 may be configured to generate a new ticket for the label. Thus, the processor 208 of the ticketing tool 202 may generate new ticket only in absence of ticket matching the label, thereby eliminating the chances of duplication of ticket in the ticketing tool. In one non-limiting embodiment of the present disclosure, JIRA may be used as the ticketing tool and the JIRA may comprise application programming interfaces (APIs) for searching a ticket associated with the label in the memory. However, the ticketing tool is not limited to above example and any other tool used for searching and generating a ticket is well within the scope of present disclosure.

[0056] **Fig. 3** shows a flow chart illustrating an exemplary artificial intelligence (AI) based method 300 for training a model to detect errors in a log file of an application, in accordance with an embodiment of the present disclosure.

[0057] At block 302, the method 300 discloses selecting a plurality of log files pertaining to at least one application from one or more sources for error extraction. The application may be a social media application such as Twitter, Instagram and Snapchat or to any of the OTT platform applications such as Netflix, Prime video, etc. It is to be noted said application is not limited to above examples and any other application is well within the scope of the present disclosure.

[0058] Each of the plurality of log files may be generated in background by executing automatic scripts during the testing phase of the respective application. Each log file may contain information about usage patterns, activities, and operations within the application.

[0059] The log files may be stored locally or remotely. In one non-limiting embodiment of the present disclosure, the log files may be stored on a server and retrieved from the server for processing. The log files may be present in any format known to a person skilled in the art.

[0060] At block 304, the method 300 discloses extracting a set of errors from the plurality of log files pertaining to the at least one application. An error extraction technique may be applied to the plurality of log files. The error extraction technique may be used to remove unwanted data generated by executing automatic scripts on the application.

[0061] In one embodiment, the error extraction technique may comprise parsing the plurality of log files and locating the set of error logs by matching data present in the log files with a set of keywords and phrases. The matched set of key words and phrases may be used for further processing. In another embodiment, the error extraction technique may comprise using a catalogue having an exhaustive list of pre-defined errors relating to an application. Specifically, the received log files are parsed through the error extractor that may use the catalogue to extract errors contained therein. However, the error extraction technique is not limited to above example and any other error extraction technique known to a person skilled in the art is well within the scope of the present disclosure.

[0062] At block 306, the method 300 discloses applying natural language processing (NLP) technique to the extracted set of errors to create an error file. Precisely, the NLP technique is applied to the extracted set of errors to remove stop words and clean the file with unwanted words. Those skilled in the art will appreciate that the extracted set of errors may also include entries other than errors, which may not be relevant for training of error detection model. Thus, such non-error entries are removed by applying NLP technique and a clean file, herein referred as the error file is created that only contains errors extracted from the received log files. In an embodiment of the present disclosure, techniques like stemming or lemmatization may be used for performing NLP operations over the extracted set of errors.

[0063] At block 308, the method 300 discloses assign a token to each error, within the error file, and performing vectorization of the assigned tokens to convert, the errors within the error file, into tokenized logs using one of the known vectorization techniques.

[0064] It is to be appreciated that vectorization technique may be applied to convert the errors in the form of words into vectors and generate them into tokenized logs, which are required to train the model for predicting error, as the model efficiently understand the data in the form of the tokenized logs that are in the vector form.

[0065] At block 308, the method 300 discloses clustering the tokenized logs to generate a feature table of errors for the at least one application. The feature table may comprise a plurality of clusters along with respective label and each cluster comprises a list of errors or a list of tokenized log grouped together. The feature table may be used to test a new log file for error detection during future testing.

[0066] In an embodiment of the present disclosure, the clustering may be implemented using unsupervised learning as discussed in above embodiments. The clustering of the tokenized logs comprises defining a plurality of clusters from the tokenized logs based on a label associated with the tokenized logs and generating the feature table based on the plurality of cluster along with the respective label. The label at least indicates a category of error or a type of error. The type of error may be application crash error, network error, playback error or any other error associated with application known to a person skilled in the art.

[0067] In one non-limiting embodiment of the present disclosure, k-means clustering technique may be used for clustering the tokenized logs. However, the clustering technique is not limited to above example and any other clustering technique is well within the scope of the present disclosure.

[0068] The clustering or categorization of the tokenized logs along with the respective label is done based on a reference table. The reference table comprises error syntax and its associated label. The reference table may be used to map the tokenized logs against a respective type of error. The clusters and their respective labels are stored in a feature table.

[0069] In an embodiment of the present disclosure, the unsupervised learning may be used to train an error detection model. The error detection model may store the feature table generated through

application of unsupervised learning on the tokenized logs. The error detection model may use the feature table to detect a type or category of error in a log file. Thus, the error detection model facilitates automating the process of detecting errors in a log file of an application and reducing the dependency of testing of application on human efforts and human intelligence.

[0070] In another embodiment of the present disclosure, the steps of method 300 may be performed in an order different from the order described above.

[0071] Fig. 4 shows a flow chart illustrating an exemplary artificial intelligence (AI) based method 400 for detecting errors in a log file of an application, in accordance with an embodiment of the present disclosure.

[0072] At block 402, the method 400 discloses receiving a new log file for testing one of the at least one application. At block 404, the method 400 discloses converting the new log file into tokenized data. The conversion of the new log file into tokenized data may comprise applying the NLP technique and the vectorization technique to the at least one log file to generate the tokenized data, as discussed in detail in steps 306 and 308 of figure 3 of the foregoing paragraphs and the repetition of same is avoided for the sake of brevity.

[0073] At block 406, the method 400 discloses performing a look-up in the feature table for identifying at least one label for at least one error present in the tokenized data based on the feature table. At block 408, the method 400 discloses displaying a list of errors corresponding to the at least one detected label.

[0074] In an embodiment of the present disclosure, the step of performing a look-up in the feature table for identifying the at least one label for at least one error present in the tokenized data based on the feature table may comprise comparing the tokenized logs of each of the cluster in the feature table with the tokenized data.

[0075] In first scenario, if a label corresponding to the tokenized data is not detected during the comparison, in the feature table, it means that the error has not been identified in the log files during the training of the model. Thus, the new log file may be forwarded to the unsupervised learning block 106 (discussed above in fig. 1) for training and generating an updated feature table.

[0076] The error present in the new log file may form a new cluster in the feature table after application of unsupervised learning on the new log file. The updated feature table may facilitate detection of errors similar to the error present in the new log files generated in future.

[0077] In second scenario, if a label corresponding to the tokenized data in the new log file is detected, the method 400 discloses comparing the at least one detected label with a plurality of tickets and displaying at least one ticket matching the at least one detected label. In one non-limiting embodiment of the present disclosure, the displayed ticket may include a comment.

[0078] If the at least one label do not match the plurality of tickets, the method 400 discloses generating a new ticket. Thus, the method 400 generates new ticket only in absence of ticket matching the label, thereby eliminating the chances of duplication of ticket in the ticketing tool.

[0079] In another embodiment of the present disclosure, the steps of method 400 may be performed in an order different from the order described above.

[0080] **Fig. 5** shows a block diagram illustrating an artificial intelligence (AI) based system 500 for training a model to detect errors in a log file of an application, in accordance with an embodiment of the present disclosure.

[0081] In an embodiment of the present disclosure, The AI based system 500 may comprise an I/O interface 502, one or more processors 404, memory 406, natural language toolkit (NLTK) library 508, AI/ML module 510, units 512 communicatively coupled with each other. The unit 512 may comprise a selection unit 514, an extraction unit 516, a conversion unit 518, a generation unit 520, a detection unit 522, and a ticketing unit 522 in communication with each other.

[0082] In an embodiment of the present disclosure, the selection unit 514, using the one or more processors 504, may be configured to select a plurality of log files pertaining to at least one application from one or more sources 550 for error extraction. Further, the selection unit 514 in combination with one or more processors 504 may be configured to receive, by means of the I/O interface 502, the selected raw log files from the one or more sources 550, wherein the one or more source 550 may include external storage or computing means. Further, the application may be a social media application such as Twitter, Instagram and Snapchat or to any of the OTT platform applications such as Netflix, Prime video, etc. It is to be noted said application is not

limited to above examples and any other application is well within the scope of the present disclosure.

[0083] Further, each of the plurality of log files may be generated in background by executing automatic scripts during the testing phase of the respective application. Each log file may contain information about usage patterns, activities, and operations within the application.

[0084] In an embodiment of the present disclosure, the one more source 550 may be present remotely to the AI based system 500. In one non-limiting embodiment, the one more source 550 may be present locally to the AI based system 450. The log files may be present in any format known to a person skilled in the art.

[0085] The extraction unit 516, using the one or more processors 504, may be configured to extract a set of error logs from the plurality of log files pertaining to the at least one application. The extraction unit 516 may apply an error extraction technique to the plurality of log files. The error extraction technique may be used to remove unwanted data generated by executing automatic scripts on the application.

[0086] In one embodiment, the error extraction technique may comprise parsing the plurality of log files and locating the set of error logs by matching data present in the log files with a set of keywords and phrases. The matched set of key words and phrases may be used for further processing. In another embodiment, the error extraction technique may comprise using a catalogue having an exhaustive list of pre-defined errors relating to an application. Specifically, the received log files are parsed through an error extractor (not shown) that may use the catalogue to extract errors contained therein. However, the error extraction technique is not limited to above example and any other error extraction technique known to a person skilled in the art is well within the scope of the present disclosure.

[0087] The conversion unit 518, using the one or more processors 504, may be configured to apply one or more natural language processing (NLP) technique to the extracted set of errors to create an error file. In an embodiment, the conversion unit 518 may use the NLTK library 508 to apply the natural language processing (NLP) technique and vectorization technique.

[0088] In an aspect, the NLP technique may be used to process the set of error logs and extract useful information such as keyword and phrases from the set of error logs. Precisely, the NLP

technique is applied to the extracted set of errors to remove stop words and clean the file with unwanted words. Those skilled in the art will appreciate that the extracted set of errors may also include entries other than errors, which may not be relevant for training of error detection model. Thus, such non-error entries are removed by applying NLP technique and a clean file, herein referred as the error file is created that only contains errors extracted from the received log files. In an embodiment of the present disclosure, techniques like stemming or lemmatization may be used for performing NLP operations over the extracted set of errors

[0089] . After the application of the NLP technique, the conversion unit 518 may be configured to assign a token to each error, within the error file, and perform vectorization of the assigned tokens to convert, the errors within the error file, into tokenized logs using one of the known vectorization techniques.

[0090] It is to be appreciated that the conversion unit 518 is used to convert the errors in the form of words into vectors and generate them into tokenized logs, which are required to train the model for predicting error, as the model efficiently understand the data in the form of the tokenized logs that are in the vector form.

[0091] Further the generation unit 520, of figure 1, using the one or more processors 504, may be configured to cluster the tokenized logs to generate a feature table of errors for the at least one application. The feature table comprises a plurality of clusters along with respective label and each cluster comprises a list of errors. The feature table is used to test a new log file for error detection.

[0092] The clustering may be implemented using unsupervised learning as discussed in above embodiments. To cluster the tokenized logs, the generation unit is configured to define a plurality of clusters from the tokenized logs based on a label associated with the tokenized logs and generate the feature table based on the plurality of cluster along with the respective label. The label at least indicates a category of error or a type of error. The type of error may be application crash error, network error, playback error or any other error associated with application known to a person skilled in the art.

[0093] In one non-limiting embodiment of the present disclosure, k-means clustering technique may be used for clustering the tokenized logs. However, the clustering technique is not limited to

above example and any other clustering technique is well within the scope of the present disclosure.

[0094] The clustering or categorization of the tokenized logs along with the respective label is done based on a reference table. The reference table comprises error syntax and its associated label. The reference table may be used to map the tokenized logs against a respective type of error. The clusters and their respective labels are stored in a feature table.

[0095] In an embodiment of the present disclosure, the unsupervised learning may be used to train an AI/Machine learning (ML) model 510. The AI/ML model 510 may store the feature table generated through application of unsupervised learning on the tokenized logs, in a memory (not shown). In an embodiment, the AI/ML model 510 may further include a dedicated GPU 526 that may be configured to perform look up operation in the feature table to detect a type or category of error in a log file. Thus, the error detection model also referred herein as ML model 510 may facilitate automating the process of detecting errors in a log file of an application and reducing the dependency of testing of application on human efforts and human intelligence.

[0096] In an embodiment of the present disclosure, the detection unit 522 may be configured to receive the new log file for testing one of the at least one application, convert the new log file into tokenized data, perform a look-up in the feature table to identify at least one label for at least one error present in the tokenized data based on the feature table, and display a list of errors corresponding to the at least one detected label.

[0097] In an embodiment of the present disclosure, to convert the new log file into tokenized data, the conversion unit 518 may be configured to apply the NLP technique and the vectorization technique to the at least one log file to generate the tokenized data as described in above embodiments.

[0098] In an embodiment of the present disclosure, to detect the at least one label for at least one error present in the tokenized data based on the feature table the detection unit 522 may be configured to compare the tokenized logs of each of the cluster in the feature table with the tokenized data.

[0099] In first scenario, if a label corresponding to the tokenized data is not detected during the comparison, in the feature table, it means that the error has not been identified in the log files

during the training of the model. Thus, the new log file may be forwarded to the AI/ML model 510 for training and generating an updated feature table.

[0100] The error present in the new log file may form a new cluster in the feature table after application of unsupervised learning on the new log file. The updated feature table may facilitate detection of errors similar to the error present in the new log files generated in future.

[0101] In an embodiment of the present disclosure, the ticketing unit 524 may be configured to compare the at least one detected label with a plurality of tickets and display at least one ticket matching the at least one detected label. In one non-limiting embodiment of the present disclosure, the displayed ticket may include a comment.

[0102] The ticketing unit 524 may be configured to generate a new ticket if the at least one label do not match the plurality of tickets. Thus, the system 500 generates new ticket only in absence of ticket matching the label, thereby eliminating the chances of duplication of ticket in the ticketing tool.

[0103] In an embodiment, the units 514-524 may be dedicated hardware units capable of executing one or more instructions stored in the memory 506 for performing the operations of the AI based system 500. In another embodiment, the units 514-524 may be software modules stored in the memory 506 which may be executed by the one or more processors 504 for performing the operations of the AI based system 500.

[0104] The illustrated steps are set out to explain the exemplary embodiments shown, and it should be anticipated that ongoing technological development will change the manner in which particular functions are performed. These examples are presented herein for purposes of illustration, and not limitation. Further, the boundaries of the functional building blocks have been arbitrarily defined herein for the convenience of the description. Alternative boundaries can be defined so long as the specified functions and relationships thereof are appropriately performed. Alternatives (including equivalents, extensions, variations, deviations, etc., of those described herein) will be apparent to persons skilled in the relevant art(s) based on the teachings contained herein. Such alternatives fall within the scope and spirit of the disclosed embodiments.

[0105] Furthermore, one or more computer-readable storage media may be utilized in implementing embodiments consistent with the present disclosure. A computer-readable storage

medium refers to any type of physical memory on which information or data readable by a processor may be stored. Thus, a computer-readable storage medium may store instructions for execution by one or more processors, including instructions for causing the processor(s) to perform steps or stages consistent with the embodiments described herein. The term "computer-readable medium" should be understood to include tangible items and exclude carrier waves and transient signals, i.e., are non-transitory. Examples include random access memory (RAM), read-only memory (ROM), volatile memory, nonvolatile memory, hard drives, CD ROMs, DVDs, flash drives, disks, and any other known physical storage media.

[0106] Suitable processors include, by way of example, a processor, a special purpose processor, a conventional processor, a digital signal processor (DSP), a plurality of microprocessors, one or more microprocessors in association with a DSP core, a controller, a microcontroller, Application Specific Integrated Circuits (ASICs), Field Programmable Gate Arrays (FPGAs) circuits, any other type of integrated circuit (IC), and/or a state machine.

ADVANTAGES OF THE PRESENT DISCLOSURE

[0107] Exemplary embodiments discussed above may provide certain advantages. Though not required to practice aspects of the disclosure, these advantages may include those provided by the following features.

[0108] In an embodiment, the present disclosure completely automate the process of detecting errors in a log file of an application.

[0109] In an embodiment, the present disclosure reduces the dependency of testing of application on human efforts and human intelligence, thereby eliminating triaging team.

[0110] In an embodiment, the present disclosure reduces the duplicate errors filed in defect tracking tool.

WE CLAIM:

1. An Artificial Intelligence (AI) based method for training a model to detect errors in a log file of an application, the method comprising:

selecting a plurality of log files pertaining to at least one application from one or more sources for error extraction;

extracting a set of errors from the plurality of log files pertaining to the at least one application;

applying Natural Language Processing (NLP) to the extracted set of errors to create an error file;

assigning a token to each error, within the error file, and performing vectorization of the assigned tokens to convert, the errors within the error file, into tokenized logs; and

clustering the tokenized logs to generate a feature table of errors for the at least one application, wherein the feature table comprises a plurality of clusters along with respective label and each cluster comprising a list of errors,

wherein said feature table is used to test a new log file for error detection.

2. The method of claim 1, wherein clustering the tokenized logs comprises:

defining the plurality of clusters from the tokenized logs based on a label associated with the tokenized logs, wherein the label at least indicates a category of error; and

generating the feature table based on the plurality of cluster along with the respective label.

3. The method of claim 1, wherein for detecting the error in the new log file, the method further comprising:

receiving the new log file for testing at least one application;

applying NLP to the new log file, assigning tokens and performing vectorization of the at least one log file to generate the tokenized data;

performing a look-up in the feature table for identifying at least one label for at least one error present in the tokenized data; and

displaying a list of errors corresponding to the at least one detected label.

4. The method of claim 1, wherein the NLP creates the error file by retaining only the entries relating to errors and removing remaining entries from the extracted set of errors

5. The method of claim 3, further comprising:

comparing the at least one identified label with a plurality of tickets; and

displaying at least one ticket matching the at least one detected label; or

generating a new ticket if the at least one label do not match the plurality of tickets.

6. An Artificial Intelligence (AI) based system for training a model to detect errors in a log file of an application, the system comprising:

a memory,

one or more processors communicatively coupled to the memory, the one or more processors configured to:

select a plurality of log files pertaining to at least one application from one or more sources for error extraction;

extract a set of errors from the plurality of log files pertaining to the at least one application;

apply Natural Language Processing (NLP) to the extracted set of errors to create an error file;

assign a token to each error, within the error file, and perform vectorization of the tokens to convert, the errors within the error file, into tokenized logs; and

cluster the tokenized logs to generate a feature table of errors for the at least one application, wherein the feature table comprises a plurality of clusters along with respective label and each cluster comprising a list of errors,

wherein said feature table is used to test a new log file for error detection.

7. The system of claim 6, wherein to cluster the tokenized logs, the one or more processors is configured to:

define the plurality of clusters from the tokenized logs based on a label associated with the tokenized logs, wherein the label at least indicates a category of error; and

generate the feature table based on the plurality of cluster along with the respective label.

8. The system of claim 6, wherein to detect errors in the new log file, the one or more processors is further configured to:

receive the new log file for testing at least one application;

apply NLP to the new log file, assign tokens and perform vectorization of the at least one log file to generate the tokenized data;

perform a look-up in the feature table to identify at least one label for at least one error present in the tokenized data; and

display a list of errors corresponding to the at least one detected label.

9. The system of claim 8, wherein the one or more processors is configured to use NLP to create the error file by retaining only the entries relating to errors and removing remaining entries from the extracted set of errors.

10. The system as claimed in claim 8, wherein the one or more processors is further configured to:

compare the at least one identified label with a plurality of tickets; and

display at least one ticket matching the at least one detected label; or

generate a new ticket if the at least one label do not match the plurality of tickets.

Dated this 24th day of June 2022

-- Digitally Signed--
Bhanu Prasad
(INPA No: 3253)
Manager, IPR Dept.,
L&T Technology Services Limited,
DLF 3rd Block, 2nd Floor,
Manapakkam, Chennai - 600089.

ABSTRACT

METHOD AND SYSTEM FOR TRAINING A MODEL TO DETECT ERRORS IN A LOG FILE

An Artificial Intelligence (AI) based method for training a model to detect errors in a log file of an application is disclosed. The method comprises selecting a plurality of log files pertaining to at least one application from one or more sources for error extraction, extracting a set of error logs from the plurality of log files pertaining to the at least one application, applying natural language processing (NLP) to the extracted set of errors and performing vectorization technique to convert the extracted set of errors into tokenized logs. The method further discloses clustering the tokenized logs to generate a feature table of errors for the at least one application. The feature table comprising a plurality of clusters along with respective label and each cluster comprising a list of errors and said feature table is used to test a new log file for error detection.

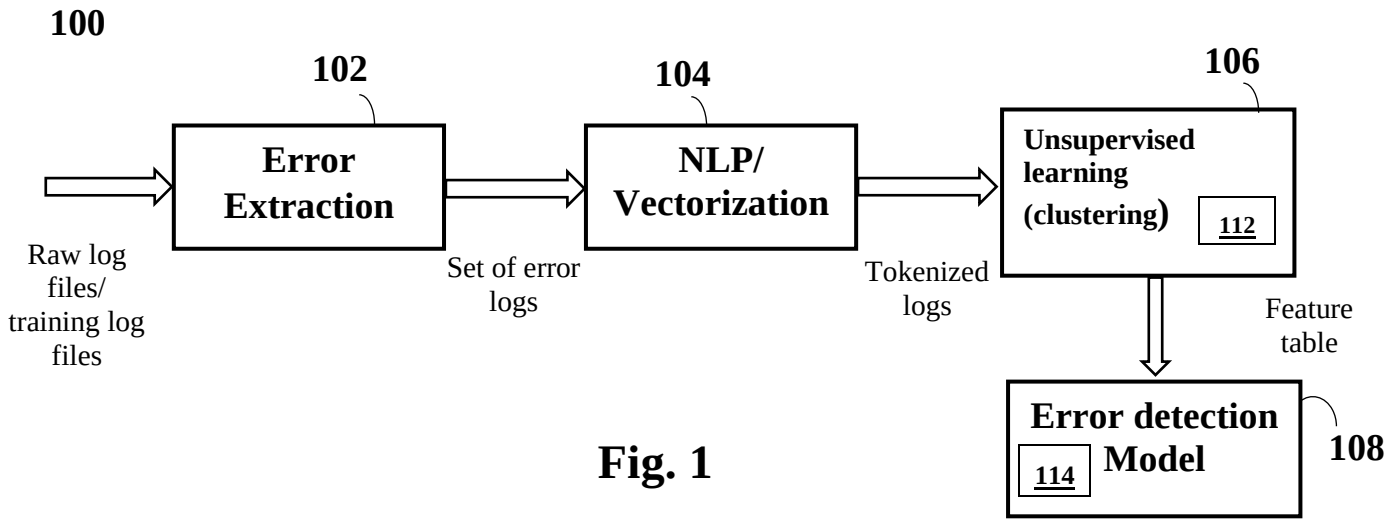

Fig. 1

-- Digitally Signed--

Bhanu Prasad
(INPA No: 3253)
Manager, IPR Dept.,
L&T Technology Services Limited,
DLF 3rd Block, 2nd Floor, Manapakkam,
Chennai - 600089.

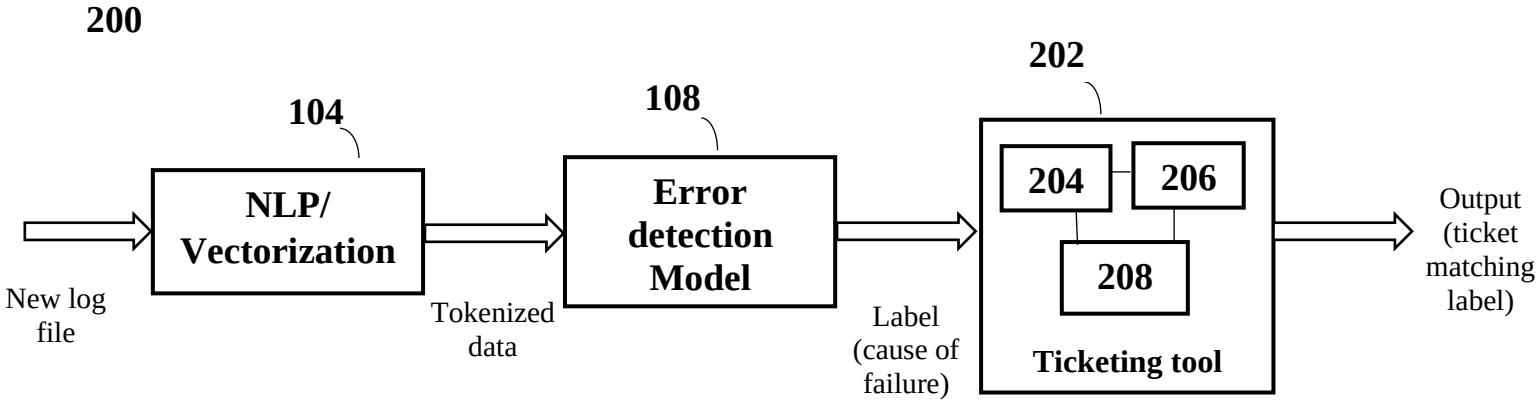

Fig. 2

-- Digitally Signed--

Bhanu Prasad
(INPA No: 3253)
Manager, IPR Dept.,
L&T Technology Services Limited,
DLF 3rd Block, 2nd Floor,
Manapakkam, Chennai - 600089.

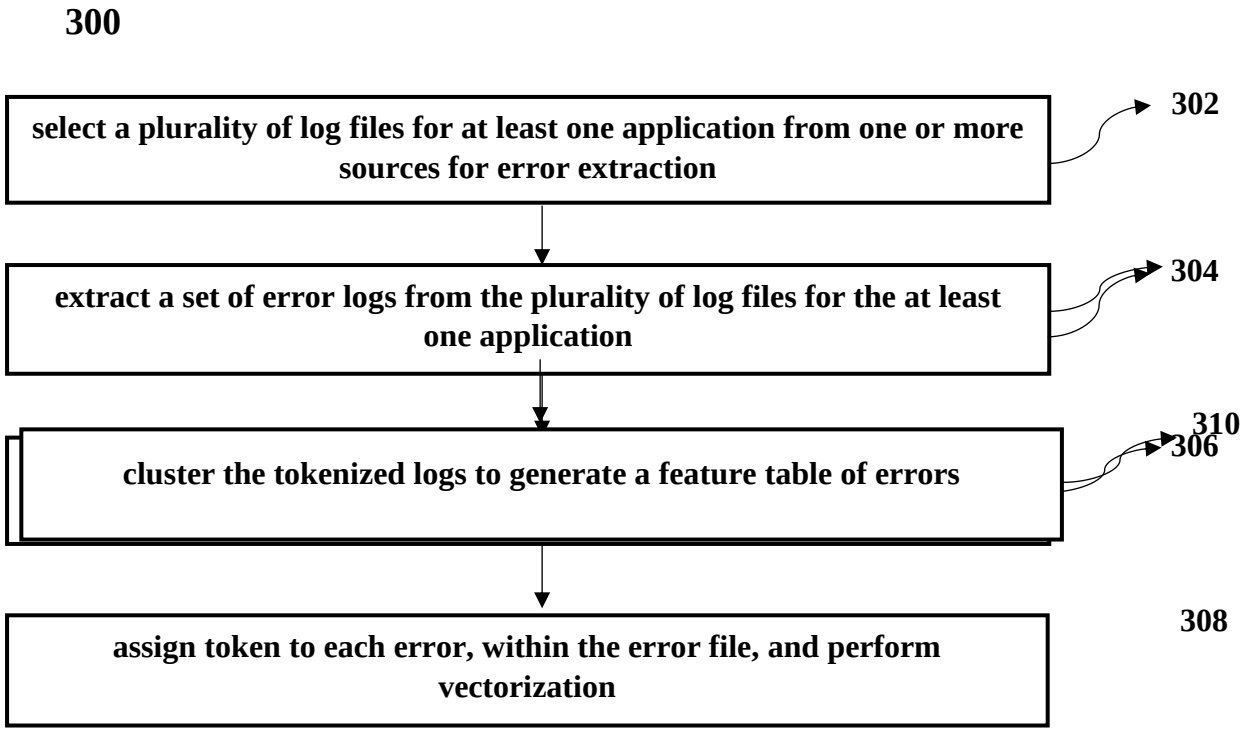

FIG. 3

-- Digitally Signed--

Bhanu Prasad
(INPA No: 3253)
Manager, IPR Dept.,
L&T Technology Services Limited,
DLF 3rd Block, 2nd Floor,
Manapakkam, Chennai - 600089.

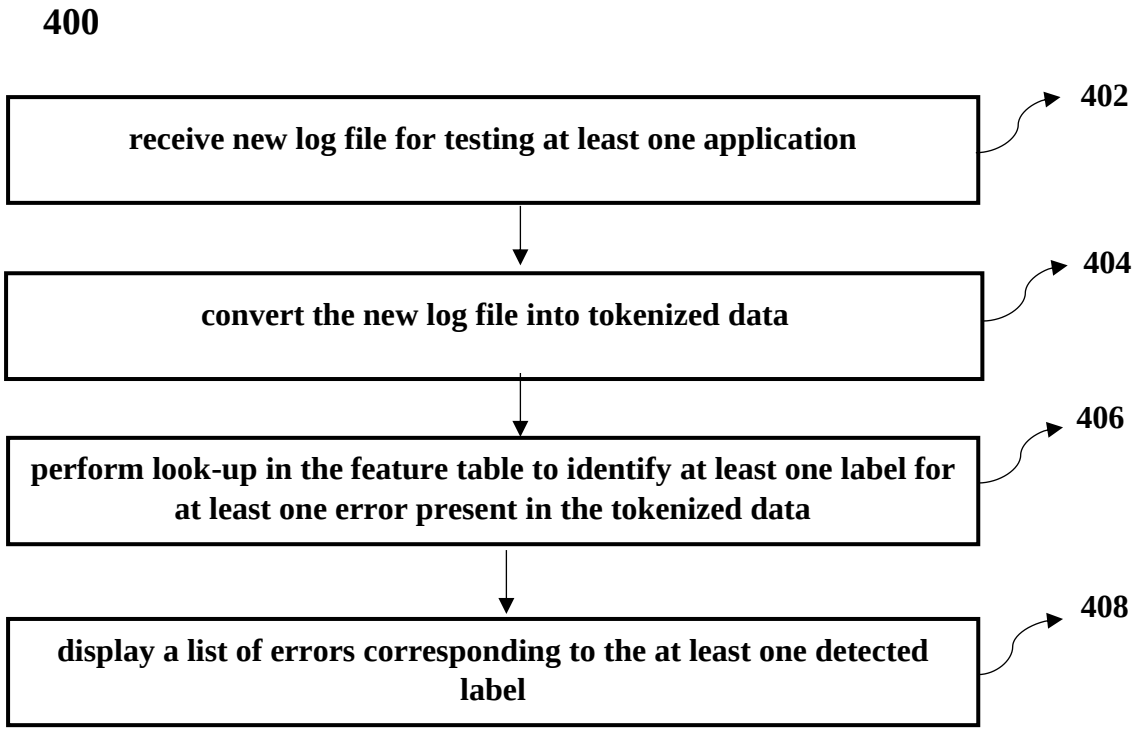

FIG. 4

-- Digitally Signed--

Bhanu Prasad
(INPA No: 3253)
Manager, IPR Dept.,
L&T Technology Services Limited,
DLF 3rd Block, 2nd Floor,
Manapakkam, Chennai - 600089.

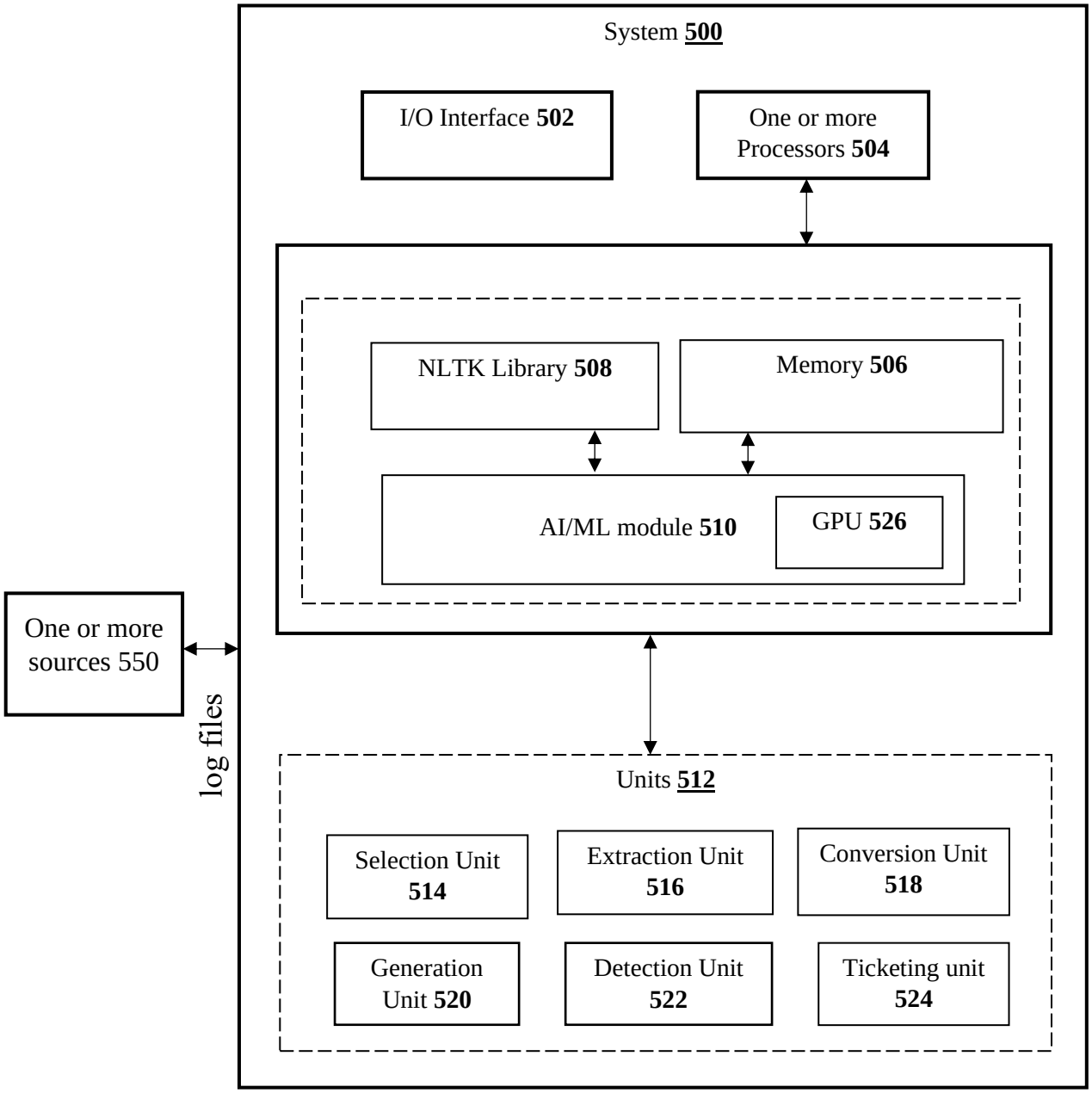

FIG. 5

-- Digitally Signed--

Bhanu Prasad
(INPA No: 3253)
Manager, IPR Dept.,
L&T Technology Services Limited,
DLF 3rd Block, 2nd Floor,
Manapakkam, Chennai - 600089.

```
07-12 11:45:05.277 27022 27038 E AndroidRuntime:        at android.app.UiAutomation.throwIfNotConnectedLocked(UiAutomation.java:1112)
07-12 11:45:05.277 27022 27038 E AndroidRuntime:        at android.app.UiAutomation.disconnect(UiAutomation.java:286)
07-12 11:45:05.277 27022 27038 E AndroidRuntime:        at android.app.Instrumentation.finish(Instrumentation.java:244)
07-12 11:45:05.277 27022 27038 E AndroidRuntime:        at androidx.test.runner.MonitoringInstrumentation.finish(MonitoringInstrumentation.java:364)
07-12 11:45:05.277 27022 27038 E AndroidRuntime:        at androidx.test.runner.AndroidJUnitRunner.finish(AndroidJUnitRunner.java:412)
07-12 11:45:05.277 27022 27038 E AndroidRuntime:        at androidx.test.runner.AndroidJUnitRunner.onStart(AndroidJUnitRunner.java:401)
07-12 11:45:05.277 27022 27038 E AndroidRuntime:        at android.app.Instrumentation$InstrumentationThread.run(Instrumentation.java:2160)
07-15 03:43:32.143 10112 10112 E AndroidRuntime: FATAL EXCEPTION: main
07-15 03:43:32.143 10112 10112 E AndroidRuntime: Process: com.ericsson.mediaroom.tv3client, PID: 10112
07-15 03:43:32.143 10112 10112 E AndroidRuntime: java.lang.NullPointerException: Attempt to invoke virtual method 'void
java.util.concurrent.CountDownLatch.countDown()' on a null object reference
07-15 03:43:32.143 10112 10112 E AndroidRuntime:        at ericsson.tv.cloud.client.android.StreamPlayerController.completeCloseVideo
(StreamPlayerController.java:846)
07-15 03:43:32.143 10112 10112 E AndroidRuntime:        at ericsson.tv.cloud.client.android.StreamPlayerController.access$500
(StreamPlayerController.java:54)
07-15 03:43:32.143 10112 10112 E AndroidRuntime:        at ericsson.tv.cloud.client.android.StreamPlayerController$3.run(StreamPlayerController.java:881)
07-15 03:43:32.143 10112 10112 E AndroidRuntime:        at android.os.Handler.handleCallback(Handler.java:873)
07-15 03:43:32.143 10112 10112 E AndroidRuntime:        at android.os.Handler.dispatchMessage(Handler.java:99)
07-15 03:43:32.143 10112 10112 E AndroidRuntime:        at android.os.Looper.loop(Looper.java:214)
07-15 03:43:32.143 10112 10112 E AndroidRuntime:        at android.app.ActivityThread.main(ActivityThread.java:7080)
07-15 03:43:32.143 10112 10112 E AndroidRuntime:        at java.lang.reflect.Method.invoke(Native Method)
07-15 03:43:32.143 10112 10112 E AndroidRuntime:        at com.android.internal.os.RuntimeInit$MethodAndArgsCaller.run(RuntimeInit.java:494)
07-15 03:43:32.143 10112 10112 E AndroidRuntime:        at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:965)
07-15 17:12:50.906 12295 12311 E AndroidRuntime: FATAL EXCEPTION: Instr: androidx.test.runner.AndroidJUnitRunner
07-15 17:12:50.906 12295 12311 E AndroidRuntime: Process: io.appium.uiautomator2.server, PID: 12295
07-15 17:12:50.906 12295 12311 E AndroidRuntime: java.lang.IllegalStateException: UiAutomation not connected!
```

Figure 6(a)

```
The file is : lesmils.txt
02-23 12:09:10.916 16625 16625 E AndroidRuntime: FATAL EXCEPTION: main

02-23 12:10:17.217 25510 25510 E AndroidRuntime: FATAL EXCEPTION: main

02-23 14:41:24.663 25731 25731 E AndroidRuntime: FATAL EXCEPTION: main

02-23 14:43:27.561  1497  1497 E AndroidRuntime: FATAL EXCEPTION: main

02-23 14:50:56.738  2001  2001 E AndroidRuntime: FATAL EXCEPTION: main

02-23 12:09:10.916 16625 16625 E AndroidRuntime: FATAL EXCEPTION: main

02-23 12:10:17.217 25510 25510 E AndroidRuntime: FATAL EXCEPTION: main

02-23 14:41:24.663 25731 25731 E AndroidRuntime: FATAL EXCEPTION: main
```

Figure 6(b)

```
['connectivityservice networkagentinfo wifi event_network_info_changed going from connected to disconnected', 'androidruntime f
atal exception main', 'playmovies guide page has failed true', 'playmovies guide page has failed true', 'dex2oat ss apk not fou
nd at data data in startv hotstar cache', 'dex2oat ss apk not found at data data in startv hotstar cache', 'dex2oat ss apk not
found at data data in startv hotstar cache', 'dex2oat ss apk not found at data data in startv hotstar cache', 'connectivityserv
ice networkagentinfo wifi event_network_info_changed going from connected to disconnected', 'dex2oat ss apk not found at data d
ata in startv hotstar cache', 'dex2oat ss apk not found at data data in startv hotstar cache', 'dex2oat ss apk not found at dat
a data in startv hotstar cache', 'dex2oat ss apk not found at data data in startv hotstar cache', 'dex2oat ss apk not found at
data data in startv hotstar cache', 'dex2oat ss apk not found at data data in startv hotstar cache', 'connectivityservice netwo
rkagentinfo wifi event_network_info_changed going from connected to disconnected', 'dex2oat ss apk not found at data data in st
artv hotstar cache', 'dex2oat ss apk not found at data data in startv hotstar cache', 'dex2oat ss apk not found at data data in
startv hotstar cache', 'dex2oat ss apk not found at data data in startv hotstar cache', 'androidruntime fatal exception main',
'androidruntime fatal exception main', 'androidruntime fatal exception main', 'androidruntime fatal exception main', 'androidru
ntime fatal exception main', 'androidruntime fatal exception main', 'androidruntime fatal exception main', 'androidruntime fata
l exception main', 'androidruntime fatal exception main', 'androidruntime fatal exception main', 'androidruntime fatal exceptio
n main', 'fullvideofragment playback failed no_content', 'fullvideofragment playback failed no_content', 'msplayer nativeplayer
setplaybackstate stopped stopped', 'msplayer nativeplayer setplaybackstate stopped stopped', 'msplayer nativeplayer onerror net
work', 'msplayer nativeplayer onerror network', 'msplayer nativeplayer setplaybackstate stopped stopped', 'fullvideofragment pl
ayback failed unknown', 'fullvideofragment playback failed unknown', 'fullvideofragment playback failed unknown', 'fullvideofra
```

Figure 6(c)

-- Digitally Signed--

Bhanu Prasad
(INPA No: 3253)
Manager, IPR Dept.,
L&T Technology Services Limited,
DLF 3rd Block, 2nd Floor,
Manapakkam, Chennai - 600089.

```
[[0.        0.        0.        ... 0.        0.        0.34480503]
 [0.5       0.        0.        ... 0.        0.        0.        ]
 [0.        0.        0.        ... 0.        0.        0.        ]
 ...
 [0.        0.2211601 0.2211601 ... 0.        0.        0.        ]
 [0.        0.2211601 0.2211601 ... 0.        0.        0.        ]
 [0.        0.        0.        ... 0.        0.        0.34480503]]
```

Figure 6(d)

Clustering

```
2,1,androidruntime fatal exception main
2,20,androidruntime fatal exception main
2,29,androidruntime fatal exception main
2,21,androidruntime fatal exception main
2,22,androidruntime fatal exception main
2,24,androidruntime fatal exception main
2,26,androidruntime fatal exception main
2,27,androidruntime fatal exception main
2,23,androidruntime fatal exception main
3,38,fullvideofragment playback failed unknown
3,40,fullvideofragment playback failed unknown
3,41,fullvideofragment playback failed unknown
3,42,fullvideofragment playback failed unknown
3,43,fullvideofragment playback failed unknown
3,32,fullvideofragment playback failed no_content
3,62,fullvideofragment playback failed player_error
3,31,fullvideofragment playback failed no_content
3,39,fullvideofragment playback failed unknown
4,0,connectivityservice networkagentinfo wifi event_network_info_changed going from connected to disconnected
4,55,connectivityservice networkagentinfo mobile lte event_network_info_changed going from connected to disconnected
4,54,connectivityservice networkagentinfo wifi event_network_info_changed going from connected to disconnected
4,66,connectivityservice networkagentinfo wifi event_network_info_changed going from connected to disconnected
4,47,connectivityservice networkagentinfo wifi event_network_info_changed going from connected to disconnected
4,45,connectivityservice networkagentinfo wifi event_network_info_changed going from connected to disconnected
4,15,connectivityservice networkagentinfo wifi event_network_info_changed going from connected to disconnected
4,8,connectivityservice networkagentinfo wifi event_network_info_changed going from connected to disconnected
5,53,msplayer nativeplayer setplaybackstate stopped stopped
5,51,msplayer nativeplayer setplaybackstate stopped stopped
5,50,msplayer nativeplayer setplaybackstate stopped stopped
5,49,msplayer nativeplayer setplaybackstate stopped stopped
5,37,msplayer nativeplayer setplaybackstate stopped stopped
5,34,msplayer nativeplayer setplaybackstate stopped stopped
5,52,msplayer nativeplayer setplaybackstate stopped stopped
5,33,msplayer nativeplayer setplaybackstate stopped stopped
6,3,playmovies guide page has failed true
6,2,playmovies guide page has failed true
```

Labelling

```
label_names = {0: 'Network Error', 1: 'Apk Not Found Error', 2: 'App crash', 3: 'Playback Error', 4: 'Network Interruption',
```

Figure 6(e)

```
[2, 6]
['App crash', 'Playback Error']
```

Figure 6(f)

-- Digitally Signed--

Bhanu Prasad
(INPA No: 3253)
Manager, IPR Dept.,
L&T Technology Services Limited,
DLF 3rd Block, 2nd Floor,
Manapakkam, Chennai - 600089.