(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
13 January 2022 (13.01.2022)

WIPO | PCT

(10) International Publication Number
WO 2022/010575 A1

(51) International Patent Classification:
G06Q 10/10 (2012.01) *G06F 40/295* (2020.01)

(21) International Application Number:
PCT/US2021/032130

(22) International Filing Date:
13 May 2021 (13.05.2021)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
16/925,453 10 July 2020 (10.07.2020) US

(71) Applicant: **MICROSOFT TECHNOLOGY LICENSING, LLC** [US/US]; One Microsoft Way, Redmond, Washington 98052-6399 (US).

(72) Inventors: **MOLAHALLI, Manish Shetty**; MICROSOFT TECHNOLOGY LICENSING, LLC, One Microsoft Way, Redmond, Washington 98052-6399 (US). **BANSAL, Chetan**; MICROSOFT TECHNOLOGY LICENSING, LLC, One Microsoft Way, Redmond, Washington 98052-6399 (US). **KUMAR, Sumit**; MICROSOFT TECHNOLOGY LICENSING, LLC, One Microsoft Way, Redmond, Washington 98052-6399 (US). **RAO, Nikitha**; MICROSOFT TECHNOLOGY LICENSING, LLC, One Microsoft Way, Redmond, Washington 98052-6399 (US). **NAGAPPAN, Nachiappan**; MICROSOFT TECHNOLOGY LICENSING, LLC, One Microsoft Way, Redmond, Washington 98052-6399 (US). **ZIMMERMANN, Thomas Michael Josef**; MICROSOFT TECHNOLOGY LICENSING, LLC, One Microsoft Way, Redmond, Washington 98052-6399 (US).

(74) Agent: **SWAIN, Cassandra T.** et al.; MICROSOFT TECHNOLOGY LICENSING, LLC, One Microsoft Way, Redmond, Washington 98052-6399 (US).

(54) Title: AUTOMATIC RECOGNITION OF ENTITIES RELATED TO CLOUD INCIDENTS

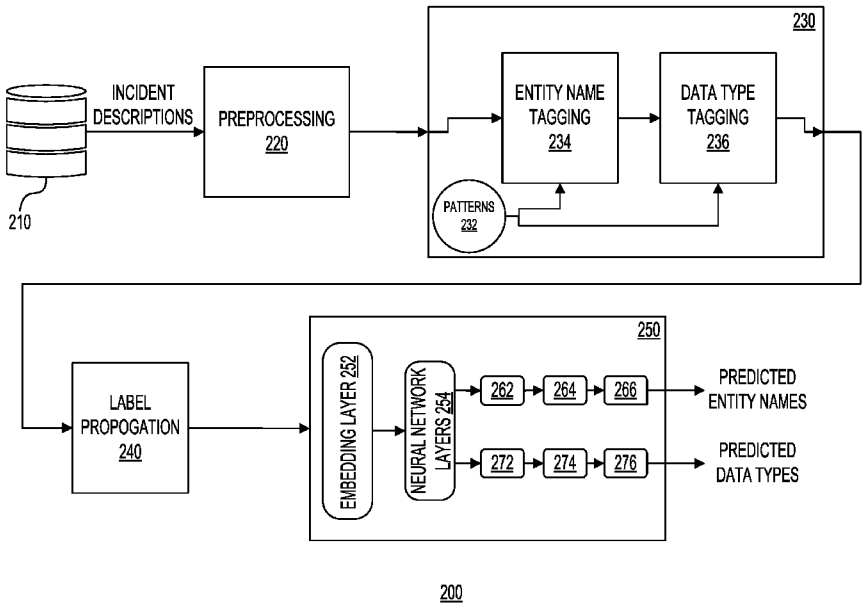


FIG. 2

(57) Abstract: Systems and methods for automatic recognition of entities related to cloud incidents are described. A method, implemented by at least one processor, for processing cloud incidents related information, including entity names and entity values associated with incidents having a potential to adversely impact products or services offered by a cloud service provider is provided. The method may include using at least one processor, processing the cloud incidents related information to convert at least words and symbols corresponding to a cloud incident into machine learning formatted data. The method may further include using a machine learning pipeline, processing at least a subset of the machine learning formatted data to recognize entity names and entity values associated with the cloud incident.

WO 2022/010575 A1

[Continued on next page]

(81) **Designated States** *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, IT, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

(84) **Designated States** *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*

Published:

— *with international search report (Art. 21(3))*

AUTOMATIC RECOGNITION OF ENTITIES RELATED TO CLOUD INCIDENTS

BACKGROUND

[0001] The public cloud includes a global network of servers that perform a variety of functions, including storing and managing data, running applications, and delivering content or services, such as streaming videos, provisioning electronic mail, providing office productivity software, or handling social media. The servers and other components may be located in data centers across the world. While the public cloud offers services to the public over the Internet, businesses may use private clouds or hybrid clouds. Both private and hybrid clouds also include a network of servers housed in data centers.

[0002] Managing cloud incidents is difficult because of the large size of the unstructured information related to cloud incidents.

SUMMARY

[0003] In one example, the present disclosure relates to a method, implemented by at least one processor, for processing cloud incidents related information, including entity names and entity values associated with incidents having a potential to adversely impact products or services offered by a cloud service provider. The method may include using the at least one processor, processing the cloud incidents related information to convert at least words and symbols corresponding to a cloud incident into machine learning formatted data. The method may further include using a machine learning pipeline, processing at least a subset of the machine learning formatted data to recognize entity names and entity values associated with the cloud incident.

[0004] In another example, the present disclosure relates to a system, including at least one processor, for processing cloud incidents related information, including entity names and entity values associated with incidents having a potential to adversely impact products or services offered by a cloud service provider. The system may be configured to using the at least one processor, process the cloud incidents related information to convert at least words and symbols corresponding to a cloud incident into machine learning formatted data. The system may further be configured to using a machine learning pipeline, process at least a subset of the machine learning formatted data to recognize entity names and entity values associated with the cloud incident.

[0005] In yet another example, the present disclosure relates to a method,

implemented by at least one processor, for processing cloud incidents related information, including entity names, entity values, and data types associated with incidents having a potential to adversely impact products or services offered by a cloud service provider. The method may include using the at least one processor, processing the cloud incidents related information to convert at least words and symbols corresponding to a cloud incident into machine learning formatted data. The method may further include using a first machine learning pipeline, as part of a first prediction task, processing at least a subset of the machine learning formatted data to recognize entity names and entity values associated with the cloud incident. The method may further include using a second machine learning pipeline, as part of a second prediction task, processing at least a subset of the machine learning formatted data to recognize data types associated with the cloud incident.

[0006] This Summary is provided to introduce a selection of concepts in a simplified form that are further described below in the Detailed Description. This Summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended to be used to limit the scope of the claimed subject matter.

BRIEF DESCRIPTION OF THE DRAWINGS

[0007] The present disclosure is illustrated by way of example and is not limited by the accompanying figures, in which like references indicate similar elements. Elements in the figures are illustrated for simplicity and clarity and have not necessarily been drawn to scale.

[0008] FIG. 1 is a block diagram of an incident lifecycle in accordance with one example;

[0009] FIG. 2 shows a block diagram of a machine learning pipeline for automatically extracting entity names and data types related to cloud incidents;

[00010] FIG. 3 is a block diagram of a system for performing methods associated with the present disclosure in accordance with one example;

[00011] FIGs. 4A and 4B show a deep learning model with a multi-head architecture in accordance with one example;

[00012] FIG. 5 shows a visual representation of the degree of attention paid to various parts of an incident description in accordance with one example;

[00013] FIG. 6 shows a system environment for implementing a machine learning pipeline of FIG. 2 for automatically extracting entity names and data types related to cloud

incidents in accordance with one example;

[00014] FIG. 7 shows a layout for an incident description in accordance with one example;

[00015] FIG. 8 shows another layout for an incident description in accordance with one example;

[00016] FIG. 9 shows a flow chart of a method for processing cloud incidents related information, including recognizing entity names and entity values in accordance with one example; and

[00017] FIG. 10 shows a flow chart of another method for processing cloud incidents related information, including recognizing entity names, entity values, and data types in accordance with one example.

DETAILED DESCRIPTION

[00018] Examples described in this disclosure relate to automatic recognition of entities related to cloud incidents. Certain examples relate to automatically recognizing entity names and data types related to cloud incidents using a machine learning pipeline. The public cloud includes a global network of servers that perform a variety of functions, including storing and managing data, running applications, and delivering content or services, such as streaming videos, electronic mail, office productivity software, or social media. The servers and other components may be located in data centers across the world. While the public cloud offers services to the public over the Internet, businesses may use private clouds or hybrid clouds. Both private and hybrid clouds also include a network of servers housed in data centers. Regardless of the arrangement of the cloud infrastructure, incidents requiring attention by the cloud service provider occur frequently.

[00019] Incident management includes activities such as automated triaging of incidents and incident diagnosis/detection. Structured knowledge extraction from incidents may require the use of machine learning. Machine learning may be used to extract information from sources, such as sources accessible via uniform resource links (e.g., web pages). In software artifacts like incidents, the vocabulary is not limited to the English language or other human languages. As an example, incidents' related information contains not just textual information concerning the incidents, but also information concerning entities such as GUIDs, Exceptions, IP Addresses, etc. Certain examples described in the present disclosure leverage a multi-task deep learning model for unsupervised knowledge extraction from information concerning incidents, such as cloud incidents. Advantageously, the unsupervised learning may eliminate the inefficiency of

annotating a large amount of training data.

[00020] In certain examples, a framework for unsupervised knowledge extraction from service incidents is described. As part of certain examples, the knowledge extraction problem is framed as a named-entity recognition task for extracting factual information related to the cloud incidents. Certain examples related to the present disclosure leverage structural patterns like key, value pairs and tables for bootstrapping the training data. Other examples relate to using a multi-task learning based Bi-LSTM-CRF model, which leverages not only the semantic context associated with the incident descriptions, but also the data-types associated with the extracted named entities. Experiments with this unsupervised machine learning based approach show good results with a high precision of 0.96. In addition, because the described systems and methods in the present disclosure are domain agnostic, they can be applied to other types of services and teams. Moreover, these systems and methods can be extended to other artifacts, including support tickets and logs. Using the knowledge extracted by the example approaches described herein, significantly more accurate models for downstream tasks like incident triaging can also be built.

[00021] FIG. 1 is a block diagram of an incident lifecycle 100 in accordance with one example. In this example, incident lifecycle 100 may broadly be classified into four phases: alerting phase 110, investigation phase 120, triaging phase 140, and resolution phase 150. In alerting phase 110, during an incident alert stage 112, an incident may be triggered when the service monitoring metrics fall below a predefined level in terms of the performance (e.g., slow response to a query), a slow transfer rate, a customer complaint or escalation, a system hang or crash, or the like. In general, telemetry systems deployed for monitoring services being offered via the cloud platform may collect telemetry data via various sensors. The monitoring of such sensor data may trigger an incident as part of incident alert stage 112. Once an incident alert is generated, as part of investigation phase 120, the information related to the incident alert may be stored in an incident database as part of incident creation stage 122. Investigation phase 120 may further include an escalation to team stage 124, during which the incident may then be escalated to a relevant team. In one example, the identification of the relevant team may be automatic (e.g., based on heuristics or component ownership). Investigation phase 120 may further include an investigation by the team stage 126. As part of this stage, the relevant team may investigate the incident(s), and as part of engagement or reassignment stage 128, may engage with the relevant stakeholders or re-route the incident(s) to the appropriate team.

As part of investigation phase 120, in problem identification stage 132, the cause(s) of the problem(s) that resulted in the incident alert(s) may be identified.

[00022] With continued refence to FIG. 1, once the appropriate team identifies the cause(s) of the problem(s) that resulted in the incident alert(s), the processing may move to triaging phase 140. In this phase, the incident(s) may be triaged according to any prioritization scheme. Next, in reporting error(s)/bug(s) stage 144, the appropriate error(s)/bug(s) related to the incident may be reported to the engineering teams. Next, in resolution phase 150, the incident may be resolved as part of incident resolution stage 152. Finally, as part of resolution phase 150, during fixing error(s)/bug(s) stage 154, any error(s) and/or bug(s) may be fixed such that the incidents caused by such error(s) and/or bug(s) do not recur. Other activities including root cause analysis may be pursued in parallel to ensure that incidents do not repeat in the future. Although FIG. 1 shows a certain number of phases as part of lifecycle 100 that are arranged in a certain manner, lifecycle 100 may include additional or fewer phases. In addition, although FIG. 1 shows a certain arrangement of stages within each phase, the phases may include additional or fewer stages, which may be arranged differently.

[00023] FIG. 2 shows a block diagram of a machine learning pipeline 200 for automatically extracting entity names and data types related to cloud incidents. Machine learning pipeline 200 may include a storage 210, which may store incident descriptions. As explained earlier, the incident descriptions may include various unstructured pieces of information that may be generated as a result of incident alerts. Storage 210 may also be used to store incident logs, telemetry data, and support tickets.

[00024] With continued reference to FIG. 2, in this example, machine learning pipeline 200 may include several components, including preprocessing 220, unsupervised data labeling 230, label propagation 240, and multi-task learning 250. Machine learning pipeline 200 may be implemented using both offline training components and online prediction components. Offline training components may be responsible for training of the various machine language models, validating the models, and publishing the validated models.

[00025] Still referring to FIG. 2, preprocessing 220 may be configured to process the incident descriptions and incident summaries, including applying a data cleaning process. Service incident descriptions and summaries may be created by various sources such as external customers, feature engineers, or automated monitoring systems. The incidents related information could be in various forms, such as textual statements,

conversations, stack traces, shell scripts, images, etc. While each of these types of unstructured information may be difficult to process, these descriptions contain useful information. In this example, preprocessing 220 may include several steps. As an example, first, the tables in the incident descriptions that have more than two columns may be pruned. In addition, the HTML tags may be removed using regexes and HTML parsers. As part of preprocessing 220, the incident descriptions and incidence summaries may be segmented into sentences using newline characters. Next, the individual sentences may be processed by cleaning up extra spaces and then they may be tokenized into words. The tokenization technique may be selected to handle even camel-case tokens (e.g., iPhone) and URLs as well.

[00026] Still referring to FIG. 2, unsupervised data labeling 230 may include identifying a set of entity names and then using the identified entity names as labels for tagging individual tokens in every incident description from a selected dataset. Identification of the set of entity names may include identifying patterns 232. Patterns 232 may include key value pairs (e.g., separated by a colon or a hyphen, such as key:value or key-value), tables, or any other data structure that can be used to represent relationships among keys, values, other such types of information. Patterns 232 may be extracted by identifying relationships in the incident descriptions. As an example, a key value pair in an incident description may be "Status code: 401." In this example, the text preceding the colon may be extracted as the entity name—Status code—and the text following the colon may be extracted as the entity value—401. In another example, another key value pair in an incident description may be "Problem type: VM not found." In this example, the text preceding the colon may be extracted as an entity name—Problem type—and the text following the colon may be extracted as the entity value—VM not found. Tables also occur quite frequently in the incident descriptions, especially the ones that are created by bots or by monitoring services. The text in the header tags '<th>' may be extracted as the entity name and the values in the corresponding rows may be extracted as entity values.

[00027] Entity names may correspond to various cloud services. Table 1 below shows an example of cloud services and related entity names.

Service Name	Related Entities
Visual Studio	{ Subscription Id, Vault Id,

	Secret Name, Version, Thumbprint, Service ID, Run Message, }

Bing	{
	Account,
	Resource Type,
	Resource,
	Current State,
	Namespace,
	Metric,
	Monitor,
	}

Exchange	{
	Subscription Id,
	Forest,
	Forest Type,
	Location,
	Machine,
	Rack,
	Monitoring Tenants
	}

Teams	{
	Tenant Name,
	Problem Description,
	Web/Desktop/Mobile App,
	Affected User,
	Object Id,
	Tenant Id,
	Tenant Id,
	}

Table 1

[00028] The initial candidate set of entity names and values may be noisy since pattern extraction 232 includes extracting almost all of the text that matches certain patterns. In certain examples, entity names may correspond to the category names (e.g., instance, people, location, etc.). To reduce noise in the initial candidate set, any entity names that contain symbols or numbers may be filtered out. To generate a more robust set of named-entities, n-grams (n: 1 to 3) may be extracted from the entity names of the candidates by selecting the top 100, or another number depending on the size of the data and other factors, most frequently occurring n-grams. In this process, less frequently used entity names (likely noisy candidate entity names) such as "token acquisition starts," may be pruned. Also with the n-gram analysis, a candidate entity such as ["My Subscription ID is", "6572"] may be transformed to ["Subscription ID", "6572"] since "Subscription ID" is a commonly occurring bi-gram in the candidate set.

[00029] Next, as part of data type tagging 236, for the refined entity name candidate set, the data type of the entity values may be determined. As an example, along with regexes, certain Python functions such as "isnumeric" may be used. The use of the data types may help improve the accuracy for the individual prediction tasks. An example set of data types may include the following data types: (1) basic types (e.g., numeric, Boolean, alphabetical, alphanumeric, non-alphanumeric); (2) complex types (e.g., GUID, URI, IP address, URL); and (3) other types (e.g., any data types that do not fit neatly into the basic or the complex types of data types). In one example, to arrive at the most likely data type, the data type may be determined for each instance of a named entity. Then, conflicts may be resolved by taking the most frequent type. For instance, if "VM IP" entity is most commonly specified as an IP Address but sometimes is specified as a Boolean, due to noise or dummy values, the data type may be resolved to be an IP Address. Table 2 below shows additional examples of entity names, the corresponding data types, and an example of each entity name.

Entity Name	Data Type	Example
Problem Type	Alphabetical	VNet Failure
Exception Message	Alphabetical	The VPN gateway deployment operation failed due to an intermittent error
Failed Operation Name	Alphabetical	Create and Mount Volume

Resource Id	URI	/resource/2aa3abc0-7986-1abc-a98b-443fd7245e6f-resourcegroups/cs-net/providers/network/frontdoor/
Tenant Id	GUID	4536dcd6-e2e1-3465-a22b-d25f62456233
Vnet Id	GUID	45ea1234-123b-7969-adaf-e0255045569e
Link with Details	URL	https://supportcenter.cloudx.com/caseoverview?srid=112
Device Name	Other	sab01-98cba-1d
Source IP	IP Address	198.168.0.1
Status Code	Number	500
Location	AlphaNumeric	eastus2

Table 2

[00030] Once the set of entity names is finalized, the incident descriptions may be parsed and each token in the incident descriptions may be tagged. As part of entity name tagging 234, unsupervised machine learning algorithms may be used to tag the incident descriptions with entity names. An example of a tagged sentence, which may be part of an incident description, is shown in Table 3 below.

Sentence	"VNetId : 4536dcd6-e2e1-3465-a22b-d25f62456233 has operation issue : delete"
Tagged Sentence	[VNetId, <O>] [:, <O>] [4536dcd6-e2e1-3465-a22b-d25f62456233, <V_NET_ID>] [has, <O>] [operation [issue, <O>] [:, <O>] [delete, <ISSUE>]

Table 3

[00031] In Table 3, <O>, which may be viewed as <Other> or <Outside> refers to tokens that are not entities. The tagged sentences, such as the one shown in Table 3, may be used to create a labeled dataset that can be used to train the machine learning models

used as part of multi-task learning 250.

[00032] Referring back to FIG. 2, machine learning pipeline 200 may further include label propagation 240. Unsupervised data labeling 230 allows bootstrapping of the training data using the pattern extraction. While this allows the generation of a seed dataset, the recall may suffer since the entities could occur inline within the incident descriptions without the key-value pair patterns or tabular patterns. The absence of any ground truth or any labeled data poses a problem. In one example, label propagation 240 may be used to solve this challenge. Label propagation 240 may use unsupervised machine learning techniques to label the incident descriptions, which may then be used to train a deep learning based model. In this example, to avoid over-fitting the model on the specific patterns, the labels may be diversified as part of this process.

[00033] In this example, the entity names and values extracted in the bootstrapping process and their types may be propagated to an entire corpus of incident descriptions. As an example, if the IP Address "127.0.0.1" was extracted as a "Source IP" entity, then all un-tagged occurrences of "127.0.0.1" in the corpus may be tagged as "Source IP." Certain corner cases may need to be handled differently. For instance, the aforementioned technique may not be usable for entities with the Boolean data type. As an example, an entity name may be "Is Customer Impacted" and the value may be "true" or "false." In this case, all occurrences of the word true or false cannot be labeled as corresponding to the entity "Is Customer Impacted." Label propagation 240 may also not work for all multi token entities, particularly the ones which are descriptive.

[00034] To the extent different occurrences of a particular value were tagged as different entities during bootstrapping, conflicts may be resolved using various techniques. As an example, an IP address (e.g., "127.0.0.1") can be "Source IP" in one incident while it may be "Destination IP" in another incident. In this example, during label propagation 240, such conflicts may be resolved based on popularity, (e.g., the value may be tagged with the entity name which occurs more frequently across the corpus). The frequency of occurrences may be tracked using histograms or other similar data structures.

[00035] Still referring to FIG. 2, machine learning pipeline 200 may further include multi-task learning 250. Multi-task learning 250 may automate the task of creating labeled data for deep learning models which can further generalize knowledge extraction. Multi-task learning may include an embedding layer 252. Incident descriptions may be converted to word level vectors using an embedding layer 252. As an example, an incident description may include words W1, W2, W3, and WN, which may be converted

into vectors for further processing. Multi-task learning 250 may further include shared neural network layers 254 and task-specific layers. Multi-task learning 250 may solve two entity recognition tasks simultaneously—entity name recognition task (l_1) and data type recognition task (l_2). The entity name prediction is treated as the main task and data type prediction is treated as the auxiliary task. In this example, entity name recognition may include the use of shared neural network layers 254 and layers labeled as 262, 264, and 266. In addition, in this example, data type recognition may include the use of shared neural network layers 254 and layers labeled as 272, 274, and 276. In this example, layers 262 and 272 may comprise a time distributed dense layer 460 of FIG. 4B; layers 264 and 274 may comprise an attention layers 470 of FIG. 4B; and layers 266 and 276 may comprise a conditional random fields (CRF) layer 480 of FIG. 4B.

[00036] The losses may initially be calculated individually for both tasks, l_1 and l_2, and then combined into $loss_c$ using a weighted sum. The parameter $loss_{weights} = (\propto, \beta)$ may be used to control the importance between the main task and the auxiliary task as follows: $loss_c = \propto \times l_1 + \beta \times l_2$. During the training, multi-task learning 250 may aim to minimize the $loss_c$ but the individual losses are back-propagated to only those layers that produced the output. With such an approach, the lower level common layers are trained by both tasks, whereas the task specific layers are trained by individual losses. Additional details concerning various components of machine learning pipeline 200 are provided later with respect to FIGs. 4A and 4B. Although FIG. 2 shows certain components of machine learning pipeline 200 that are arranged in a certain manner, machine learning pipeline 200 may include additional or fewer components arranged differently. In addition, certain components of machine learning pipeline 200 may be used for training of the machine learning models and other components may be used for prediction tasks. Thus, machine learning pipeline 200 may include only one of these types of components or both of these types of components depending upon the functions being performed using such a pipeline.

[00037] FIG. 3 is a block diagram of a system 300 for performing methods associated with the present disclosure in accordance with one example. As an example, system 300 may be used to implement the various parts of machine learning pipeline 200 of FIG. 2. System 300 may include a processor(s) 302, I/O component(s) 304, memory 306, presentation component(s) 308, sensors 310, database(s) 312, networking interfaces 314, and I/O port(s) 316, which may be interconnected via bus 320. Processor(s) 302 may

execute instructions stored in memory 306. Processor(s) 302 may include CPUs, GPUs, ASICs, FPGAs, or other types of logic configured to execute instructions. I/O component(s) 304 may include components such as a keyboard, a mouse, a voice recognition processor, or touch screens. Memory 306 may be any combination of non-volatile storage or volatile storage (e.g., flash memory, DRAM, SRAM, or other types of memories). Presentation component(s) 308 may include displays, holographic devices, or other presentation devices. Displays may be any type of display, such as LCD, LED, or other types of display. Sensor(s) 310 may include telemetry or other types of sensors configured to detect, and/or receive, information (e.g., conditions associated with the various devices in a data center). Sensor(s) 310 may include sensors configured to sense conditions associated with CPUs, memory or other storage components, FPGAs, motherboards, baseboard management controllers, or the like. Sensor(s) 310 may also include sensors configured to sense conditions associated with racks, chassis, fans, power supply units (PSUs), or the like. Sensor(s) 310 may also include sensors configured to sense conditions associated with Network Interface Controllers (NICs), Top-of-Rack (TOR) switches, Middle-of-Rack (MOR) switches, routers, power distribution units (PDUs), rack level uninterrupted power supply (UPS) systems, or the like.

[00038] Still referring to FIG. 3, database(s) 312 may be used to store any of the data or files (e.g., incident descriptions or the like) as needed for the performance of methods described herein. Database(s) 312 may be implemented as a collection of distributed databases or as a single database. Network interface(s) 314 may include communication interfaces, such as Ethernet, cellular radio, Bluetooth radio, UWB radio, or other types of wireless or wired communication interfaces. I/O port(s) 316 may include Ethernet ports, Fiber-optic ports, wireless ports, or other communication ports.

[00039] Instructions corresponding to preprocessing 220, unsupervised data labeling 230, label propagation 240, and multi-task learning 250 and their respective constituent parts may be stored in memory 306 or another memory. These instructions when executed by processor(s) 302, or other processors, may provide the functionality associated with machine learning pipeline 200. The instructions corresponding to machine learning pipeline 200, and related components, could be encoded as hardware corresponding to an A/I processor. In this case, some or all of the functionality associated with the learning-based analyzer may be hard-coded or otherwise provided as part of an A/I processor. As an example, A/I processor may be implemented using a field programmable gate array (FPGA) with the requisite functionality. Other types of

hardware such as ASICs and GPUs may also be used. The functionality associated with machine learning pipeline 200 may be implemented using any appropriate combination of hardware, software, or firmware. Although FIG. 3 shows system 300 as including a certain number of components arranged and coupled in a certain way, it may include fewer or additional components arranged and coupled differently. In addition, the functionality associated with system 300 may be distributed or combined, as needed.

[00040] FIGs. 4A and 4B show a deep learning model 400 with a multi-head architecture in accordance with one example. In this example, deep learning model 400 may be used to implement certain aspects of multi-task learning 250 of FIG. 2. In this example, words and symbols extracted from an incident description may be first converted into a sequence of vectors. The sequence of vectors may be interpreted, both in a forward direction and in a reverse direction, by a Bi-directional Long Short-term Memory (LSTM) layer 430. The two prediction tasks may include entity name prediction and data type prediction. The two tasks may be handled in a way that some common parameters and layers (e.g., layers within the box 420 of FIG. 4) may be shared for both tasks, but there may also be task specific layers (e.g., separate time distributed dense layers 462 and 464, separate attention layers 472 and 474, and separate conditional random fields (CRF) layers 482 and 484). Although separate such layers are used, for ease of explanation, these layers are addressed using common reference numerals as shown in FIG. 4B: time distributed dense layer 460, attention layer 470, and CRF layer 480. Time distributed dense layer 460 may transpose the Bi-directional LSTM hidden vectors to the shape of the output labels. An attention mechanism (e.g., attention layer 470) may help the model bias the learning towards the more relevant sections of the sentences. In addition, in this example, a conditional random fields (CRF) layer 480 may produce a valid sequence of output labels. As shown in FIG. 4B, the output may include entity name prediction 492 and data type prediction 494. Back propagation using a combination of loss functions may be performed during training and the individual tag precision may be evaluated using recall and F1 metrics.

[00041] In certain examples, by using the underlying common information contained among related tasks multi-task learning may be used to improve generalization. In the context of classification or sequence labelling, the multi-task learning may improve the performance of individual tasks by learning them jointly. In certain examples described herein, named-entity recognition is the primary task. In this task, the machine learning models may primarily learn from context words that support occurrences of

entities. Incorporating a complimentary task of predicting the data type of a token may reinforce intuitive constraints, resulting in better training of the machine learning models. For example, in an input like "The SourceIPAddress is 127.0.0.1," the token 127.0.0.1 is identified more accurately by the machine learning models described herein, as the entity name "Source IP Address" because it is also identified as the data-type "IP Address", in parallel. In sum, the machine learning models supplement the intuition that all Source IP Addresses are of the data type IP addresses; thus, improving the model performance. Accordingly, in these examples data type prediction is used as the auxiliary task for the deep learning models. Various types of architectures may allow multi-task learning, including but not limited to, multi-head architectures, cross-snitch networks, and sluice networks. Certain examples described herein use a multi-head architecture, where the lower level features generated by the two neural network layers are shared, whereas the other layers are task specific.

[00042] As noted previously, the entity name prediction is treated as the main task and data type prediction is treated as the auxiliary task. The losses are initially calculated individually for both tasks, l_1 and l_2, and then combined into $loss_c$ using a weighted sum. The parameter $loss_{weights} = (\propto, \beta)$ may be used to control the importance between the main and the auxiliary task as follows: $loss_c = \propto \times\ l_1 + \beta \times l_2$. During the training, deep learning model 400 aims to minimize the $loss_c$ but the individual losses are back-propagated to only those layers that produced the output. With such an approach, the lower level common layers are trained by both tasks, whereas the task specific layers are trained by individual losses.

[00043] With continued reference to FIG. 4A, in this example, incident descriptions 402 may be converted to word level vectors using a pre-trained embedding layer 410. As an example, an incident description may include words W1, W2, W3, and WN, which may be converted into vectors for further processing. Pre-trained embedding layer 410 may be implemented as a GloVe embedding layer or a word2vec embedding layer. GloVe relates to a model that captures linear substructure relations in a global corpus of words, revealing regularities in syntax as well as semantics. The GloVe model, trained on five different corpora, covers a vast range of topics and tokens. In this example, in a preferred embodiment, the 100 dimension version of GloVe may be used to create pre-trained embedding layer 410 with the pre-trained GloVe weights.

[00044] Vector size may be a 768-dimension vector or a 1024-dimension vector.

Additional operations, including position embedding, sentence embedding, and token masking may also be performed as part of pre-trained embedding layer 410. Position embedding may be used to identify token positions within a sequence. Sentence embedding may be used to map sentences to vectors. Token masking may include replacing a certain percentage of the words in each sequence with a mask token. These vectors may improve the performance of the prediction tasks being performed using deep learning model 400. In this example, these vectors may act as characteristic features in named entity recognition being performed using deep learning model 400.

[00045] Still referring to FIG. 4A, Bi-directional LSTM network 430 may be implemented as one or more Recurrent Neural Networks (RNNs). An RNN maintains historic information extracted from a sequence or a series like data. This feature may enable an RNN-based model to make predictions at a certain time step, conditional to viewed history. Thus, an RNN may take a sequence of vectors $(x_1, x_2, \ldots, x_n)$ as input and return as sequence of vectors $(h_1, h_2, \ldots, h_3)$ that encodes information at every time step. Although RNNs are capable of encoding and learning dependencies that are spread over long time steps, at times they may fail to do so; this is because RNNs tend to be biased towards more recent updates in a long sequence of situations.

[00046] In one example, Long Short-term Memory (LSTM) networks may be used to capture long range dependencies using several gates. These gates may control a portion of the input and pass to the memory cell, and the portion from the previous hidden state to forget. An example LSTM network may comprise a sequence of repeating RNN layers or other types of layers. Each layer of the LSTM network may consume an input at a given time step, e.g., a layer's state from a previous time step, and may produce a new set of outputs or states. In the case of using the LSTM, a single chunk of content may be encoded into a single vector or multiple vectors. As an example, a word or a combination of words (e.g., a phrase, a sentence, or a paragraph) may be encoded as a single vector. Each chunk may be encoded into an individual layer (e.g., a particular time step) of an LSTM network. In this example, Bi-directional LSTM network 430 may include a first LSTM network 440 and a second LSTM network 450. LSTM network 440 may be configured to process a sequence of words from left to right and LSTM network 450 may be configured to process a sequence of words from right to left. LSTM network 440 may include LSTM cell 442, LSTM cell 444, LSTM cell 446, and LSTM cell 448, which may be coupled to receive inputs and to provide outputs, as shown in FIG. 4A. LSTM network 450 may include LSTM cell 452, LSTM cell 454, LSTM cell 456, and LSTM cell 458,

which may be coupled to receive inputs and to provide outputs, as shown in FIG. 4A. In addition, as shown in FIG. 4A, both LSTM cell 442 and LSTM cell 452 may provide their output to hidden layer H1 453. Likewise, both LSTM cell 444 and LSTM cell 454 may provide their output to hidden layer H2 455. Similarly, both LSTM cell 446 and LSTM cell 456 may provide their output to hidden layer H3 457. Finally, both LSTM cell 448 and LSTM cell 458 may provide their output to hidden layer HN 459.

[00047] An example LSTM layer may be described using a set of equations, such as the ones below:

$$f_t = \sigma(W_f \cdot [h_{t-1}x_t] + b_c)$$
$$i_t = \sigma(W_f \cdot [h_{t-1}x_t] + b_i)$$
$$\tilde{c}_t = tanh(W_c \cdot [h_{t-1}x_t] + b_c)$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$$
$$o_t = \sigma(W_o \cdot [h_{t-1}x_t] + b_o)$$
$$h_t = o_t \circ \tanh(c_t)$$

In this example, in the above equations σ is the element wise sigmoid function and $\circ$ represents Hadamard product (element-wise). In this example, f_t, i_t, and o_t are forget, input, and output gate vectors respectively, and c_t is the cell state vector. Using the above equations, given a sentence as a sequence of real valued vectors $(x_1, x_2, \ldots, x_n)$, the LTSM (e.g., LTSM network 440 of FIG. 4A) computes $\vec{h}_t$ that represents the leftward context of the word at the current time step t. In this example, a word at the current time step t, receives context from other words that occur on either sides. Thus, a second LSTM (e.g., LSTM network 450 of FIG. 4A) interprets the same sequence in reverse, returning $\overleftarrow{h}_t$ at each time step. In this example, this combination of forward and backward LSTMs corresponds to Bi-directional LSTM network 430. The final representation of the word may be produced by concatenating the left and right context, $h_t = [\vec{h}_t; \overleftarrow{h}_t]$. In this example, inside each LSTM layer, the inputs and hidden states may be processed using a combination of vector operations (e.g., dot-product, inner product, or vector addition) or non-linear operations, if needed.

[00048] The instructions corresponding to the machine learning system could be encoded as hardware corresponding to an A/I processor. In this case, some or all of the functionality associated with the learning-based analyzer may be hard-coded or otherwise provided as part of an A/I processor. As an example, A/I processor may be implemented using an FPGA with the requisite functionality.

[00049] Any of the learning and inference techniques such as Linear Regression, Support Vector Machine (SVM) set up for regression, Random Forest set up for regression, Gradient-boosting trees set up for regression and neural networks may be used. Linear regression may include modeling the past relationship between independent variables and dependent output variables. Neural networks may include artificial neurons used to create an input layer, one or more hidden layers, and an output layer. Each layer may be encoded as matrices or vectors of weights expressed in the form of coefficients or constants that might have been obtained via off-line training of the neural network. Neural networks may be implemented as Recurrent Neural Networks (RNNs), Long Short Term Memory (LSTM) neural networks, or Gated Recurrent Unit (GRUs). All of the information required by a supervised learning-based model may be translated into vector representations corresponding to any of these techniques.

[00050] With reference to FIG. 4B, deep learning model 400 for entity name and data type prediction from the incident descriptions or other sources of incidents' related information may include additional layers, including a time distributed dense layer 460, an attention layer 470, and a conditional random fields (CRF) layer 480. Time distributed dense layer 460 may transpose the Bi-directional LSTM hidden vectors to the shape of the output labels. Attention layer 470 may help the model bias it is learning towards the more relevant sections of the sentences. In addition, CRF layer 480 may produce a valid sequence of output labels. As shown in FIG. 4B, each of these layers may process outputs received from bi-directional LSTM network 430.

[00051] Still referring to FIG. 4B, time distributed dense layer 460 may be trained to reshape the vectors received from bi-directional LSTM network 430. In this example, attention layer 470 may be implemented by using the Bidirectional Encoder Representations from Transformers (BERT) model. Attention layer 470 may take as input the hidden states from Bi-directional LSTM network 430, after these inputs have been transposed to output dimensions using time distributed dense layer 460. In this example, attention layer 460 may be implemented at the words level as a neural layer, with a weight parameter W_a. In one example, let $h = (h_1, h_2, \ldots, h_T)$ be the input to the attention layer 470, the attention weights and final representation h^* of the sentence is formed as follows:

$$scores = W_a^T h$$

$$\alpha = softmax(scores)$$

$$r = h\alpha^T$$

$$h^* = tanh\,(r)$$

[00052] In the example equations shown above, the $softmax$ and $tanh$ functions are applied element-wise on the input vectors. The values corresponding to h and h^* may be concatenated and passed to the next layer. In one example, attention layer 460 may include transformers corresponding to the BERT model. Transformers may convert input sequences into output sequences using self-attention. Transformers may be configured to have either 12 or 24 hidden (h) layers. Transformers may include fully-connected network (FCN) layers, including the FCN (Query), FCN (Key), and FCN (Value) layers.

[00053] Referring now to FIG. 5, a visual representation 500 of the degree of attention paid to various parts of an incident description is shown in accordance with one example. The attention vector α for a test sentence, shown in FIG. 5, illustrates that the attention layer learns to give more emphasis to tokens that have a higher likelihood of being entities. The degree of attention varies from lower to higher. In this example, the different degrees of attention, from a lower degree of attention to a higher degree of attention, are shown as 510, 520, 530, 540, 550, 560, and 570. In case of long sequences, the different degrees of attention to certain sections of the sequence, which are more likely to contain entities, helps improve the sensitivity of deep learning model 400.

[00054] Referring back to FIG. 4B, the use of the hidden state representations (h_t) as word features to make independent tagging decisions at the word level may still leave the issue of inherent dependencies across the output labels unaddressed. For example, the entity names and corresponding values may have contextual or other types of constraints. Similarly, data types may be constrained in terms of the data types that are usable with certain entity names. In one example, by learning these dependencies and generalizing them to sentences without such constraints, the tagging decisions may be jointly modeled using conditional random fields as part of CRF layer 480.

[00055] To explain one example implementation of CRF layer 480, consider an input sequence $X = (x_1, x_2, \ldots, x_3)$ and an output sequence $y = (y_1, y_2, \ldots, y_n)$, where n is the number of words in the sentence. Assuming, for this example, P is the matrix of the probability scores of shape $n \times k$, where k is the number of distinct tags in the output of bi-directional LSTM network 430, including the dense and attention layers. In other words, in this example $P_{i,j}$ is a score that the i^{th} word corresponds to the j^{th} tag. In this example, as part of CRF layer 480, first a score is computed for the output sequence, y, using the example equation below:

$$s(X, y) = \sum_{i=0}^{n} A\, y_i, y_{i+1} + \sum_{i=0}^{n} P\, i, y_i$$

where A represents the matrix of transition scores. Thus, in this example, $A_{i,j}$ is the score for the transition from tag_i to tag_j. Then the score is converted to a probability for the sequence y to be the right output using a softmax over Y (all possible output sequences) using the example equation below:

$$p(y|X) = \frac{e^{s(X,y)}}{\sum_{y' \in Y} e^{s(X,y)}}$$

[00056] In this example, the model corresponding to CRF layer 480 learns by maximizing the log-probability of the correct y. While extracting the tags for the input, the output sequence with the highest score is predicted using the following example equation:

$$y^* = argmax\, p(y'|X)$$
$$y' \in Y$$

[00057] Thus, in this example implementation of CRF layer 480, CRF layer 480 and attention layer 470 push the model towards learning a valid sequence of tags. As an example, for a sentence that includes the entity name subscription ID and the entity value 12345 (separated by a colon), attention layer 470 may tag the colon as a tenant ID.

[00058] In one example, the hyper-parameters for the deep learning models may be set as follows: word embedding size is set to 100, the hidden LSTM layer size is set to 200 cells, and the maximum length of a sequence is limited to 300. These example hyper-parameters may be used with all models. The machine learning models may be trained using any set of computing resources, including using system 300 of FIG. 3. Each computing resource may be implemented using any number of graphics processing units (GPUs), computer processing units (CPUs), memory (e.g., SRAM or other types of memory), or field programmable gate arrays (FPGAs). Application Specific Integrated Circuits (ASICs), Erasable and/or Complex programmable logic devices (PLDs), Programmable Array Logic (PAL) devices, and Generic Array Logic (GAL) devices may also be used to implement the computing resources. In addition, although FIG. 4B describes the use of the BERT model for attention layer 460, any serializable neural network model may be partitioned and used.

[00059] FIG. 6 shows a system environment for implementing a machine learning pipeline 200 for automatically extracting entity names and data types related to cloud

incidents in accordance with one example. In this example, system environment 600 may correspond to a portion of a data center. As an example, the data center may include several clusters of racks including platform hardware, such as server nodes, storage nodes, networking nodes, or other types of nodes. Server nodes may be connected to switches to form a network. The network may enable connections between each possible combination of switches. As used in this disclosure, the term data center may include, but is not limited to, some or all of the data centers owned by a cloud service provider, some or all of the data centers owned and operated by a cloud service provider, some or all of the data centers owned by a cloud service provider that are operated by a customer of the service provider, any other combination of the data centers, a single data center, or even some clusters in a particular data center. System environment 600 may include server1 610 and serverN 630. System environment 600 may further include data center related functionality 660, including deployment/monitoring 670, directory/identity services 672, load balancing 674, data center controllers 676 (e.g., software defined networking (SDN) controllers and other controllers), and routers/switches 678. Server1 610 may include host processor(s) 611, host hypervisor 612, memory 613, storage interface controller(s) (SIC(s)) 614, cooling 615, network interface controller(s) (NIC(s)) 616, and storage disks 617 and 618. ServerN 630 may include host processor(s) 631, host hypervisor 632, memory 633, storage interface controller(s) (SIC(s)) 634, cooling 635, network interface controller(s) (NIC(s)) 636, and storage disks 637 and 638.

[00060] With continued reference to FIG. 6, server1 610 may be configured to support virtual machines, including VM1 619, VM2 620, and VMN 621. The virtual machines may further be configured to support applications, such as APP1 622, APP2 623, and APPN 624. ServerN 630 may be configured to support virtual machines, including VM1 639, VM2 640, and VMN 641. The virtual machines may further be configured to support applications, such as APP1 642, APP2 643, and APPN 644. Each of server1 610 and serverN 630 may also support various types of services, including file storage, application storage, and block storage for the various tenants of the cloud service provider responsible for managing system environment. In this example, system environment 600 may be enabled for multiple tenants using the Virtual eXtensible Local Area Network (VXLAN) framework. Each virtual machine (VM) may be allowed to communicate with VMs in the same VXLAN segment. Each VXLAN segment may be identified by a VXLAN Network Identifier (VNI).

[00061] Deployment/monitoring 670 may interface with a sensor API that may

allow sensors to receive and provide information via the sensor API. Software configured to detect or listen to certain conditions or events may communicate via the sensor API any conditions associated with devices that are being monitored by deployment/monitoring 670. Remote sensors or other telemetry devices may be incorporated within the data centers to sense conditions associated with the components installed therein. Remote sensors or other telemetry may also be used to monitor other adverse signals in the data center and feed the information to deployment/monitoring 670. As an example, if fans that are cooling a rack stop working then that may be sensed by the sensors and reported to the deployment/monitoring 670. Although FIG. 6 shows system environment 600 as including a certain number of components arranged and coupled in a certain way, it may include fewer or additional components arranged and coupled differently. In addition, the functionality associated with system environment 600 may be distributed or combined, as needed. Moreover, although FIG. 6 shows VMs, other types of compute entities, such as containers, micro-VMs, microservices, unikernels for serverless functions, may be supported by the host servers in a like manner.

[00062] FIG. 7 shows a layout 700 for an incident description in accordance with one example. Layout 700 may correspond to an incident description being displayed, or otherwise being communicated, to a person/team assigned to address the incident at issue. Layout 700 may include user interface elements to allow interaction. As an example, the following menu options may be associated with layout 700 of the example incident description: Details 702, Diagnostics 704, Notifications 706, Postmortem 708, Activity Log (History) 710, and Similar Incidents 712. When a user selects Details 702 menu option, the information displayed in box 720 may be displayed. The example incident description shown in layout 700 relates to an issue with a virtual machine (VM) in a failed state. Additional details associated with the incident description are shown in box 720. Although example layout 700 shows certain aspects associated with an incident description, other incident descriptions may have a different layout and may include information other than shown in layout 700. Table 4, below, shows entity names and entity values for layout 700.


```
{
“cloud”: [ “cloudx” ],
“grant permission”: [ “true” ],
“instance_id”: [ “45ea1234-123b-7969-adaf-e0255045569e” ],
```

“tenant_id”: [“2aa3abc0-7986-1abc-a98b-443fd7245e6f”],
“ip_addres”: [“192.168.0.1”],
“issue”: [“vm in failed state, unable to delete vm or perform any activity”],
“product_subscription_id”: [“4536dcd6-e2e1-3465-a22b-d25f62456233”]
“resource_group”: [“tl”],
“link_with_details”:[https://supportcenter.cloudx.com/caseoverview?srid=1123],
}

Table 4

[00063] FIG. 8 shows another layout 800 for an incident description in accordance with one example. Layout 800 may correspond to another incident description being displayed, or otherwise being communicated, to a person/team assigned to address the incident at issue. Layout 800 may also include user interface elements to allow interaction. As an example, similar to layout 700, the following menu options may be associated with layout 800 of the example incident description: Details 802, Diagnostics 804, Notifications 806, Postmortem 808, Activity Log (History) 810, and Similar Incidents 812. When a user selects Details 802 menu option, the information displayed in box 820 may be displayed. The example incident description shown in layout 800 relates to an issue with an error associated with a virtual network (Vnet). Additional details associated with the incident description are shown in box 820. Although example layout 800 shows certain aspects associated with an incident description, other incident descriptions may have a different layout and may include information other than shown in layout 800. Table 5, below, shows entity names and entity values for layout 800.

{
“ask”: [“please remove the orphaned resources related to this”],
“problem_type” [“cannot delete v net”],
“product_subscription_id”: [“45ea123-123b-7969-adaf-e0255045569e”],
“v_net_id”: [“4536dcd6-e2e1-3465-a22b-d25f62456123”],
“v_net_name”: [“wa-vnet”],
“v_net_region”: [“east56usind”],
}

Table 5

[00064] FIG. 9 shows a flow chart 900 of a method, implemented by at least one processor, for processing cloud incidents related information, including entity names and

entity values associated with incidents having a potential to adversely impact products or services offered by a cloud service provider. Step 910 may include using the at least one processor (e.g., processor(s) 302 of FIG. 3), processing the cloud incidents related information to convert at least words and symbols corresponding to a cloud incident description into machine learning formatted data. As explained earlier, with respect to FIGs. 2-4B, using a pre-trained embedding layer (e.g., pre-trained embedding layer 410) words and symbols corresponding to the cloud incident may be converted into machine learning formatted data. As an example, the words and symbols may be converted into vector data for processing by neural networks.

[00065] Step 920 may include using a machine learning pipeline, processing at least a subset of the machine learning formatted data to recognize entity names and entity values associated with the cloud incident. As explained earlier, with respect to FIGs. 2-4B, the machine learning formatted data (e.g., vector data) may be processed to recognize entity names and entity values.

[00066] FIG. 10 shows a flow chart 1000 of a method, implemented by at least one processor, for processing cloud incidents related information, including entity names, entity values, and data types associated with incidents having a potential to adversely impact products or services offered by a cloud service provider. Step 1010 may include using the at least one processor (e.g., processor(s) 302 of FIG. 3), processing the cloud incidents related information to convert at least words and symbols corresponding to a cloud incident description into machine learning formatted data. As explained earlier, with respect to FIGs. 2-4B, using a pre-trained embedding layer (e.g., pre-trained embedding layer 410) words and symbols corresponding to the cloud incident may be converted into machine learning formatted data. As an example, the words and symbols may be converted into vector data for processing by neural networks.

[00067] Step 1020 may include using a first machine learning pipeline, as part of a first prediction task, processing at least a subset of the machine learning formatted data to recognize entity names and entity values associated with the cloud incident. As explained earlier, with respect to FIGs. 2-4B, at least a subset of the machine learning formatted data (e.g., vector data) may be processed to recognize entity names and entity values.

[00068] Step 1030 may include using a second machine learning pipeline, as part of a second prediction task, processing at least a subset of the machine learning formatted data to recognize data types associated with the cloud incident. As explained earlier, with respect to FIGs. 2-4B, the machine learning formatted data (e.g., vector data) may be

processed to recognize data types.

[00069] In one example, machine learning pipeline 200 and the corresponding deep learning model for entity name recognition and data type recognition may be deployed as part of system environment 600. As an example, machine learning pipeline 200 and the corresponding deep learning model may be deployed as a REST API (e.g., a REST API developed using the Python Flask web app framework). The REST API may offer a POST endpoint which takes the incident description as input and returns the recognized entities in JSON format. The deployment of the REST API in system environment 600 advantageously allows automatically scaling up of the service in response to demand variation. This enables the service to be cost efficient since the majority of the incidents are created during the day. In addition, deployment and monitoring tools in conjunction with machine learning pipeline 200 may enable application monitoring, as part of which service latency or failure issues may be communicated via alerts.

[00070] By efficiently recognizing entity names, entity values, and data types, systems and methods described in the present disclosure may enable other applications, as well. As an example, these systems and methods may be used for incident triaging. Advantageously, the recognized entity names and the recognized data types may reduce the feature space because a significant amount of unstructured information in the incident descriptions is not helpful. This may further help in creating incident summaries that are concise and yet informative for a service team. As a result, instead of parsing the verbose incident descriptions, the service team member may quickly analyze the concise summary and act on it, as required, per service agreements and protocols.

[00071] In addition, automated health checks may also be performed, alleviating the need for the service team member to review detailed telemetry data and logs. As an example, oversubscription (or undersubscription) of resources may be automatically identified using the automated health checks.

[00072] In conclusion, the present disclosure relates to a method, implemented by at least one processor, for processing cloud incidents related information, including entity names and entity values associated with incidents having a potential to adversely impact products or services offered by a cloud service provider. The method may include using the at least one processor, processing the cloud incidents related information to convert at least words and symbols corresponding to a cloud incident into machine learning formatted data. The method may further include using a machine learning pipeline, processing at least a subset of the machine learning formatted data to recognize entity

names and entity values associated with the cloud incident.

[00073] The method may further include using the machine learning pipeline, jointly processing at least a second subset of the machine learning formatted data with the at least the subset of the machine learning formatted data to recognize data types associated with the cloud incident. The method may further include using a multi-task learning layer, processing both the subset of the machine learning formatted data and the second subset of the machine learning formatted data to generate output data.

[00074] The method may further include: (1) using a first time distributed dense layer, reshaping a first subset of the output data, wherein the first subset of the output data corresponds to entity names and entity values, to generate a first set of reshaped data and (2) using a second time distributed dense layer reshaping a second subset of the output data, wherein the second subset of the output data corresponds to data types, to generate a second set of reshaped data. The method may further include: (1) using a first attention layer, processing the first set of reshaped data, emphasizing a first set of tokens more likely to be entity names or entity types and (2) using a second attention layer, processing the second set of reshaped data, emphasizing a second set of tokens more likely to be data types.

[00075] The method may further include (1) using learned constraints associated with entity names and entity values, helping recognize the entity names and the entity values associated with the cloud incident, and (2) using learned constraints associated with data types, helping recognize the data types associated with the cloud incident. The method may further include generating a seed database of tagged entity names and tagged entity values by unsupervised tagging of entity names and entity values based on patterns extracted from cloud incidents related information. The method may further include using unsupervised label propagation of the tagged entity names and the tagged entity values, to generate training data for training the machine learning pipeline.

[00076] In another example, the present disclosure relates to a system, including at least one processor, for processing cloud incidents related information, including entity names and entity values associated with incidents having a potential to adversely impact products or services offered by a cloud service provider. The system may be configured to using the at least one processor, process the cloud incidents related information to convert at least words and symbols corresponding to a cloud incident into machine learning formatted data. The system may further be configured to using a machine learning pipeline, process at least a subset of the machine learning formatted data to recognize

entity names and entity values associated with the cloud incident.

[00077] The system may further be configured to jointly process at least a second subset of the machine learning formatted data with the at least the subset of the machine learning formatted data to recognize data types associated with the cloud incident. The system may further be configured to using a multi-task learning layer, process both the subset of the machine learning formatted data and the second subset of the machine learning formatted data to generate output data.

[00078] The system may further be configured to: (1) using a first time distributed dense layer, reshape a first subset of the output data, wherein the first subset of the output data corresponds to entity names and entity values, to generate a first set of reshaped data and (2) using a second time distributed dense layer reshape a second subset of the output data, wherein the second subset of the output data corresponds to data types, to generate a second set of reshaped data . The system may further be configured to: (1) using a first attention layer, process the first set of reshaped data, emphasizing a first set of tokens more likely to be entity names or entity types and (2) using a second attention layer, process the second set of reshaped data, emphasizing a second set of tokens more likely to be data types. The system may further be configured to: (1) using learned constraints associated with entity names and entity values, help recognize the entity names and the entity values associated with the cloud incident, and (2) using learned constraints associated with data types, help recognize the data types associated with the cloud incident.

[00079] In yet another example, the present disclosure relates to a method, implemented by at least one processor, for processing cloud incidents related information, including entity names, entity values, and data types associated with incidents having a potential to adversely impact products or services offered by a cloud service provider. The method may include using the at least one processor, processing the cloud incidents related information to convert at least words and symbols corresponding to a cloud incident into machine learning formatted data. The method may further include using a first machine learning pipeline, as part of a first prediction task, processing at least a subset of the machine learning formatted data to recognize entity names and entity values associated with the cloud incident. The method may further include using a second machine learning pipeline, as part of a second prediction task, processing at least a subset of the machine learning formatted data to recognize data types associated with the cloud incident.

[00080] The method may further include using a multi-task learning layer, processing both the first subset of the machine learning formatted data and the second subset of the machine learning formatted data to generate output data. The method may further include: (1) using a first time distributed dense layer, reshaping a first subset of the output data, wherein the first subset of the output data corresponds to entity names and entity values, to generate a first set of reshaped data and (2) using a second time distributed dense layer reshaping a second subset of the output data, wherein the second subset of the output data corresponds to data types, to generate a second set of reshaped data.

[00081] The method may further include: (1) using a first attention layer, processing the first set of reshaped data, emphasizing a first set of tokens more likely to be entity names or entity types and (2) using a second attention layer, processing the second set of reshaped data, emphasizing a second set of tokens more likely to be data types. The method may further include: (1) using learned constraints associated with entity names and entity values, helping recognize the entity names and the entity values associated with the cloud incident, and (2) using learned constraints associated with data types, helping recognize the data types associated with the cloud incident. The method may further include: (1) generating a seed database of tagged entity names and tagged entity values by unsupervised tagging of entity names and entity values based on patterns extracted from cloud incidents related information, and (2) using unsupervised label propagation of the tagged entity names and the tagged entity values to generate training data for training the machine learning pipeline.

[00082] It is to be understood that the methods, modules, and components depicted herein are merely exemplary. Alternatively, or in addition, the functionality described herein can be performed, at least in part, by one or more hardware logic components. For example, and without limitation, illustrative types of hardware logic components that can be used include Field-Programmable Gate Arrays (FPGAs), Application-Specific Integrated Circuits (ASICs), Application-Specific Standard Products (ASSPs), System-on-a-Chip systems (SOCs), Complex Programmable Logic Devices (CPLDs), etc. In an abstract, but still definite sense, any arrangement of components to achieve the same functionality is effectively "associated" such that the desired functionality is achieved. Hence, any two components herein combined to achieve a particular functionality can be seen as "associated with" each other such that the desired functionality is achieved, irrespective of architectures or inter-medial components. Likewise, any two components

so associated can also be viewed as being "operably connected," or "coupled," to each other to achieve the desired functionality.

[00083] The functionality associated with some examples described in this disclosure can also include instructions stored in a non-transitory media. The term “non-transitory media” as used herein refers to any media storing data and/or instructions that cause a machine to operate in a specific manner. Exemplary non-transitory media include non-volatile media and/or volatile media. Non-volatile media include, for example, a hard disk, a solid-state drive, a magnetic disk or tape, an optical disk or tape, a flash memory, an EPROM, NVRAM, PRAM, or other such media, or networked versions of such media. Volatile media include, for example, dynamic memory such as DRAM, SRAM, a cache, or other such media. Non-transitory media is distinct from, but can be used in conjunction with transmission media. Transmission media is used for transferring data and/or instruction to or from a machine. Exemplary transmission media include coaxial cables, fiber-optic cables, copper wires, and wireless media, such as radio waves.

[00084] Furthermore, those skilled in the art will recognize that boundaries between the functionality of the above described operations are merely illustrative. The functionality of multiple operations may be combined into a single operation, and/or the functionality of a single operation may be distributed in additional operations. Moreover, alternative embodiments may include multiple instances of a particular operation, and the order of operations may be altered in various other embodiments.

[00085] Although the disclosure provides specific examples, various modifications and changes can be made without departing from the scope of the disclosure as set forth in the claims below. Accordingly, the specification and figures are to be regarded in an illustrative rather than a restrictive sense, and all such modifications are intended to be included within the scope of the present disclosure. Any benefits, advantages, or solutions to problems that are described herein with regard to a specific example are not intended to be construed as a critical, required, or essential feature or element of any or all the claims.

[00086] Furthermore, the terms "a" or "an," as used herein, are defined as one or more than one. Also, the use of introductory phrases such as "at least one" and "one or more" in the claims should not be construed to imply that the introduction of another claim element by the indefinite articles "a" or "an" limits any particular claim containing such introduced claim element to inventions containing only one such element, even when the same claim includes the introductory phrases "one or more" or "at least one" and indefinite articles such as "a" or "an." The same holds true for the use of definite articles.

[00087] Unless stated otherwise, terms such as "first" and "second" are used to arbitrarily distinguish between the elements such terms describe. Thus, these terms are not necessarily intended to indicate temporal or other prioritization of such elements.

CLAIMS

1. A method, implemented by at least one processor, for processing cloud incidents related information, including entity names and entity values, the method comprising:

   using the at least one processor, processing the cloud incidents related information to convert at least words and symbols corresponding to a cloud incident into machine learning formatted data; and

   using a machine learning pipeline, processing at least a subset of the machine learning formatted data to recognize entity names and entity values associated with the cloud incident.

2. The method of claim 1, further comprising using the machine learning pipeline, jointly processing at least a second subset of the machine learning formatted data with the at least the subset of the machine learning formatted data to recognize data types associated with the cloud incident.

3. The method of claim 1, further comprising using a multi-task learning layer, processing both the subset of the machine learning formatted data and the second subset of the machine learning formatted data to generate output data.

4. The method of claim 3, further comprising: (1) using a first time distributed dense layer, reshaping a first subset of the output data, wherein the first subset of the output data corresponds to entity names and entity values, to generate a first set of reshaped data and (2) using a second time distributed dense layer reshaping a second subset of the output data, wherein the second subset of the output data corresponds to data types, to generate a second set of reshaped data.

5. The method of claim 4, further comprising: (1) using a first attention layer, processing the first set of reshaped data, emphasizing a first set of tokens more likely to be entity names or entity types and (2) using a second attention layer, processing the second set of reshaped data, emphasizing a second set of tokens more likely to be data types.

6. The method of claim 5, further comprising: (1) using learned constraints associated with entity names and entity values, helping recognize the entity names and the entity values associated with the cloud incident, and (2) using learned constraints associated with data types, helping recognize the data types associated with the cloud incident.

7. The method of claim 1, further comprising generating a seed database of

tagged entity names and tagged entity values by unsupervised tagging of entity names and entity values based on patterns extracted from cloud incidents related information.

8. The method of claim 7, further comprising using unsupervised label propagation of the tagged entity names and the tagged entity values, to generate training data for training the machine learning pipeline.
9. A system, including at least one processor, for processing cloud incidents related information, including entity names and entity values associated with incidents having a potential to adversely impact products or services offered by a cloud service provider, the system configured to:

using the at least one processor, process the cloud incidents related information to convert at least words and symbols corresponding to a cloud incident into machine learning formatted data; and

using a machine learning pipeline, process at least a subset of the machine learning formatted data to recognize entity names and entity values associated with the cloud incident.

10. The system of claim 9, further configured to jointly process at least a second subset of the machine learning formatted data with the at least the subset of the machine learning formatted data to recognize data types associated with the cloud incident.
11. The system of claim 10, further configured to using a multi-task learning layer, process both the subset of the machine learning formatted data and the second subset of the machine learning formatted data to generate output data.
12. The system of claim 11, further configured to: (1) using a first time distributed dense layer, reshape a first subset of the output data, wherein the first subset of the output data corresponds to entity names and entity values, to generate a first set of reshaped data and (2) using a second time distributed dense layer reshape a second subset of the output data, wherein the second subset of the output data corresponds to data types, to generate a second set of reshaped data.
13. The system of claim 12, further configured to: (1) using a first attention layer, process the first set of reshaped data, emphasizing a first set of tokens more likely to be entity names or entity types and (2) using a second attention layer, process the second set of reshaped data, emphasizing a second set of tokens

more likely to be data types.

14. The system of claim 13, further configured to: (1) using learned constraints associated with entity names and entity values, help recognize the entity names and the entity values associated with the cloud incident, and (2) using learned constraints associated with data types, help recognize the data types associated with the cloud incident.

15. A method, implemented by at least one processor, for processing cloud incidents related information, including entity names, entity values, and data types, the method comprising:

using the at least one processor, processing the cloud incidents related information to convert at least words and symbols corresponding to a cloud incident into machine learning formatted data;

using a first machine learning pipeline, as part of a first prediction task, processing at least a first subset of the machine learning formatted data to recognize entity names and entity values associated with the cloud incident; and

using a second machine learning pipeline, as part of a second prediction task, processing at least a second subset of the machine learning formatted data to recognize data types associated with the cloud incident.

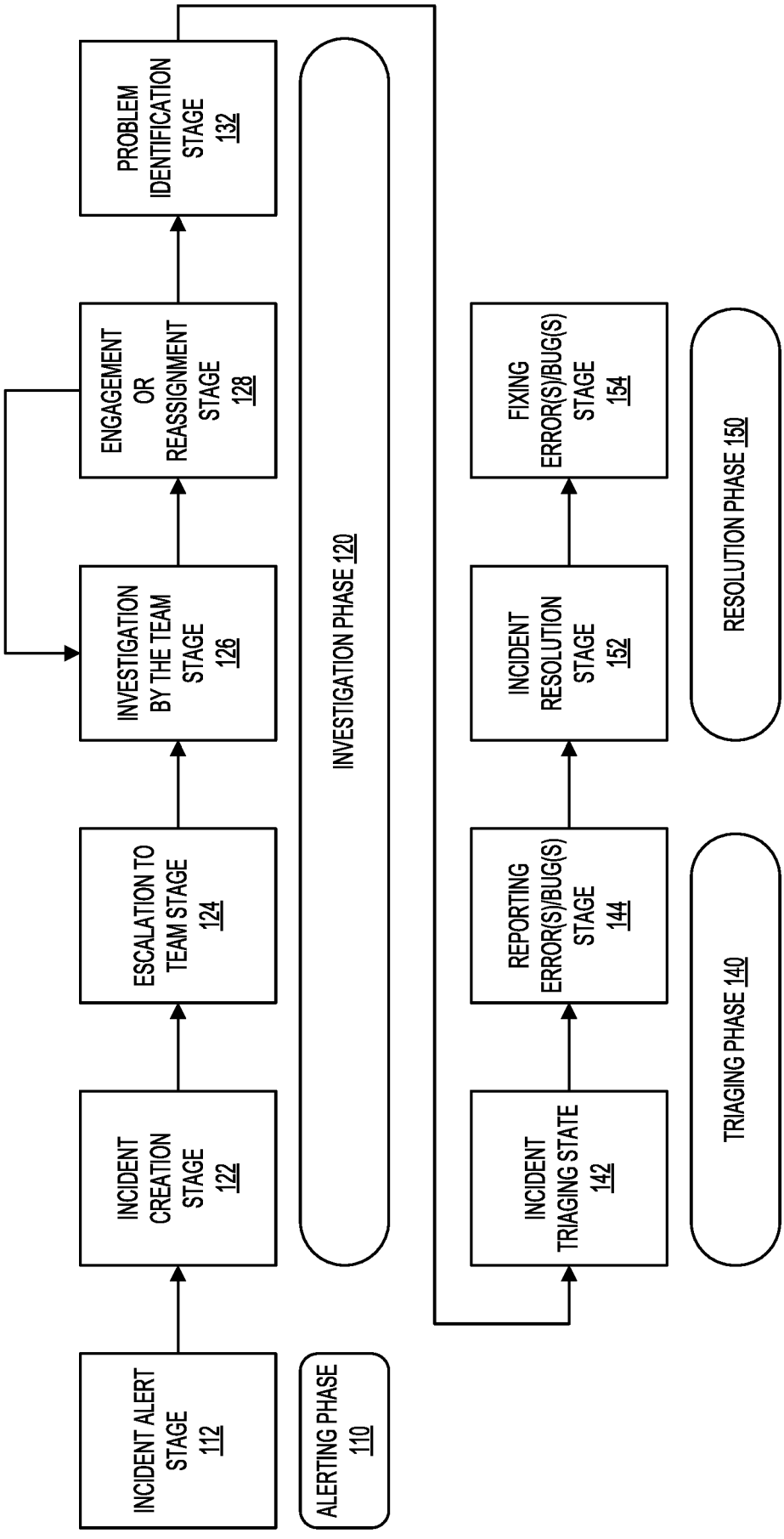

INCIDENT ALERT STAGE 112
ALERTING PHASE 110
INCIDENT CREATION STAGE 122
ESCALATION TO TEAM STAGE 124
INVESTIGATION BY THE TEAM STAGE 126
ENGAGEMENT OR REASSIGNMENT STAGE 128
PROBLEM IDENTIFICATION STAGE 132
INVESTIGATION PHASE 120
INCIDENT TRIAGING STATE 142
REPORTING ERROR(S)/BUG(S) STAGE 144
TRIAGING PHASE 140
INCIDENT RESOLUTION STAGE 152
FIXING ERROR(S)/BUG(S) STAGE 154
RESOLUTION PHASE 150
100

FIG. 1

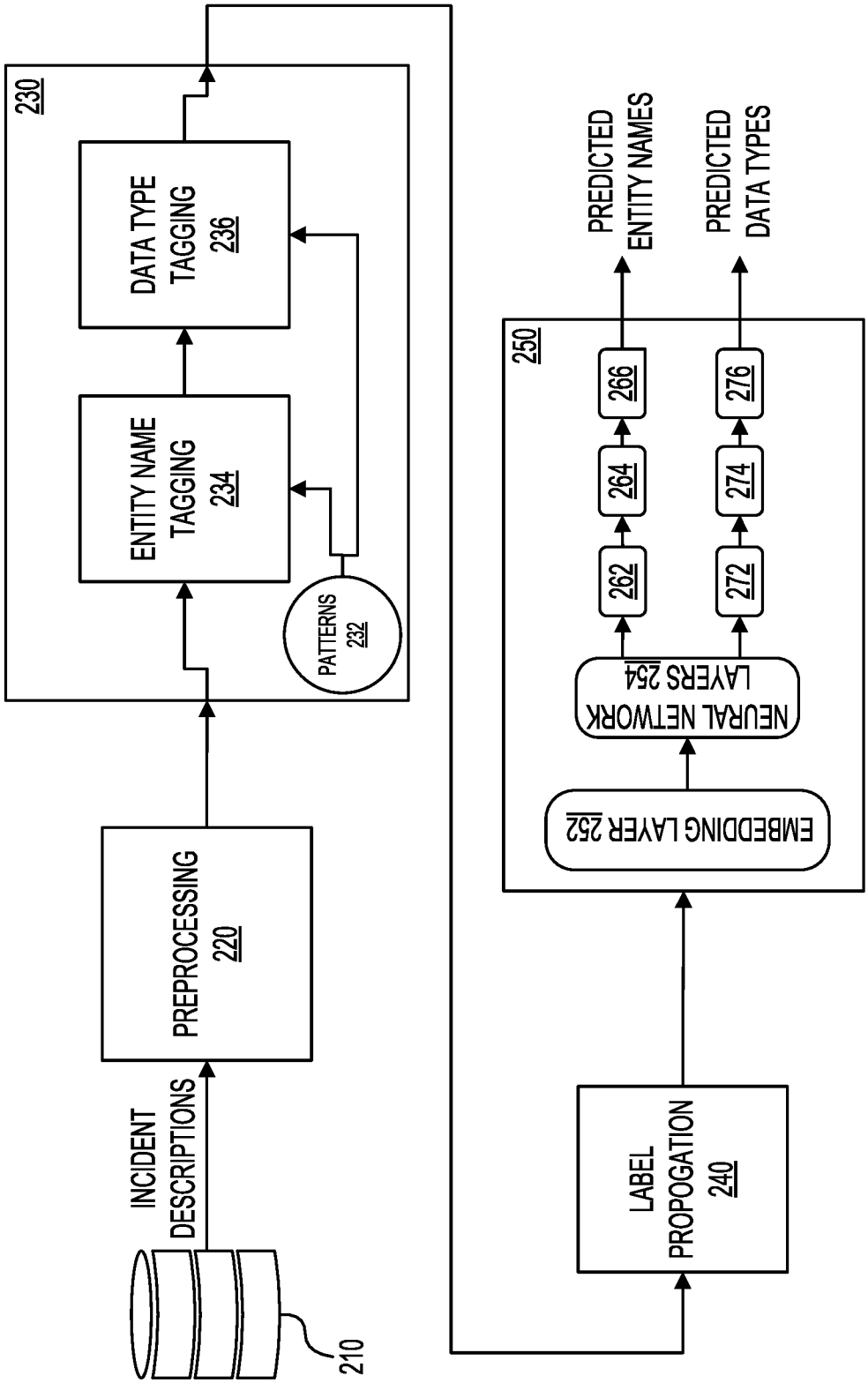

INCIDENT DESCRIPTIONS
210
PREPROCESSING 220
230
ENTITY NAME TAGGING 234
DATA TYPE TAGGING 236
PATTERNS 232
LABEL PROPOGATION 240
250
EMBEDDING LAYER 252
NEURAL NETWORK LAYERS 254
262
264
266
272
274
276
PREDICTED ENTITY NAMES
PREDICTED DATA TYPES
200

FIG. 2

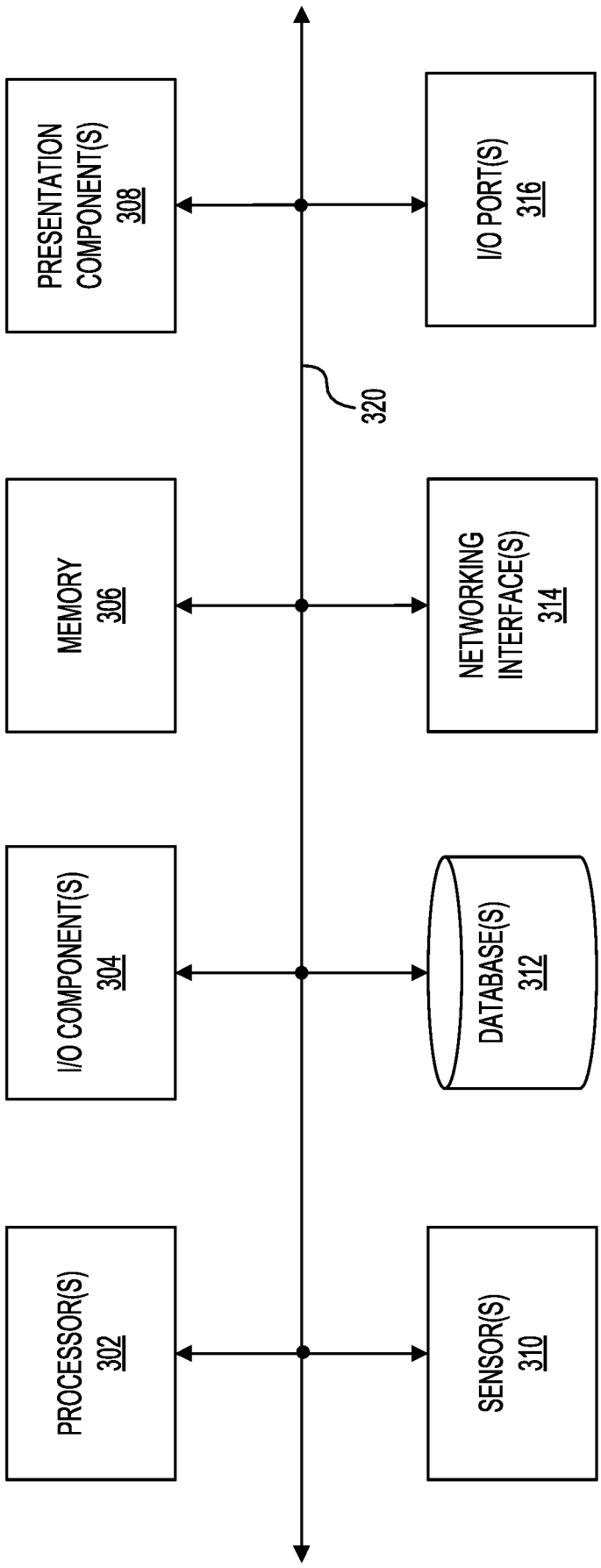

PRESENTATION COMPONENT(S) 308
I/O PORT(S) 316
320
MEMORY 306
NETWORKING INTERFACE(S) 314
I/O COMPONENT(S) 304
DATABASE(S) 312
PROCESSOR(S) 302
SENSOR(S) 310
300

FIG. 3

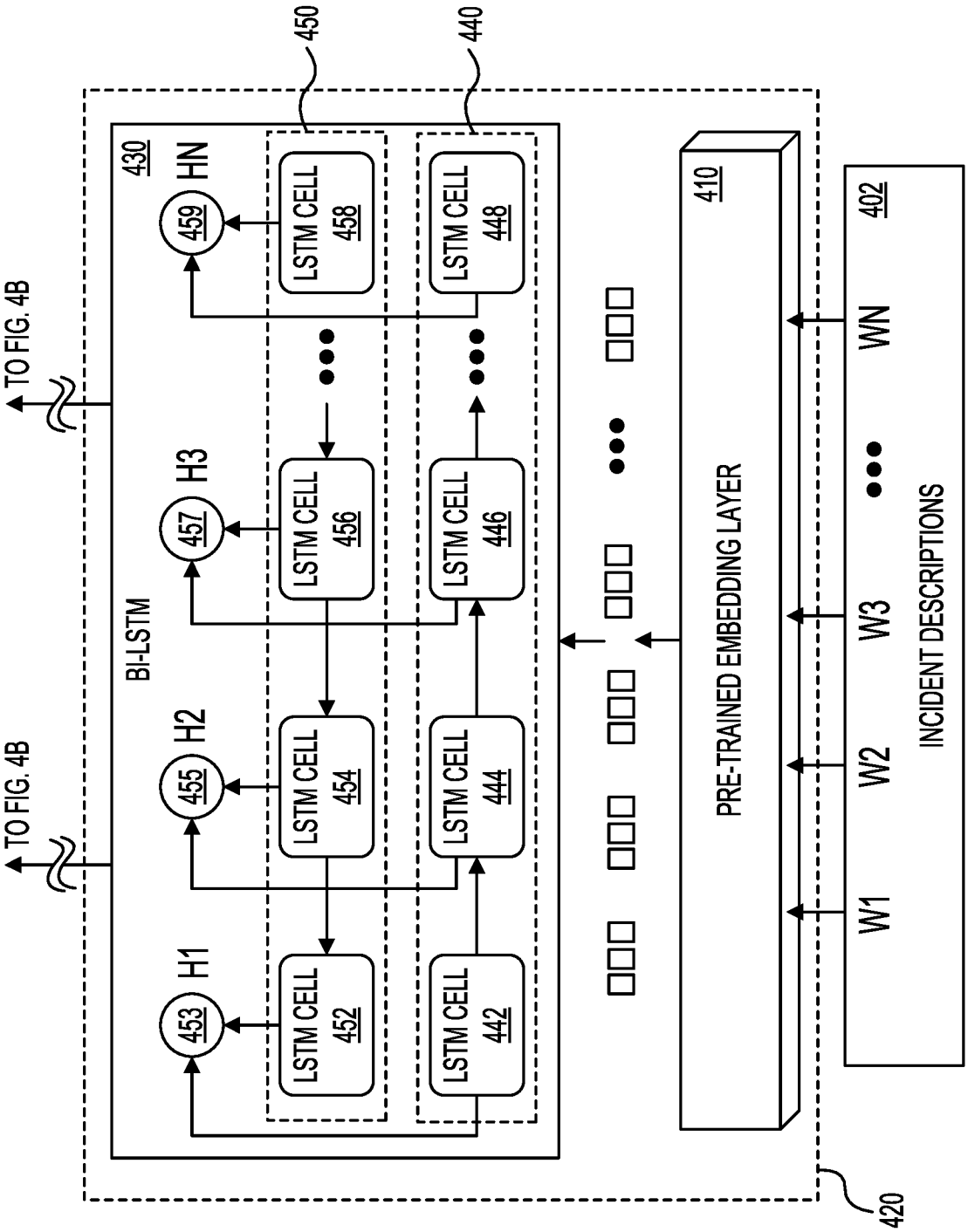

TO FIG. 4B
TO FIG. 4B
BI-LSTM
430
450
440
H1
H2
H3
HN
453
455
457
459
LSTM CELL 452
LSTM CELL 454
LSTM CELL 456
LSTM CELL 458
LSTM CELL 442
LSTM CELL 444
LSTM CELL 446
LSTM CELL 448
PRE-TRAINED EMBEDDING LAYER
410
W1
W2
W3
WN
INCIDENT DESCRIPTIONS
402
400
420

FIG. 4A

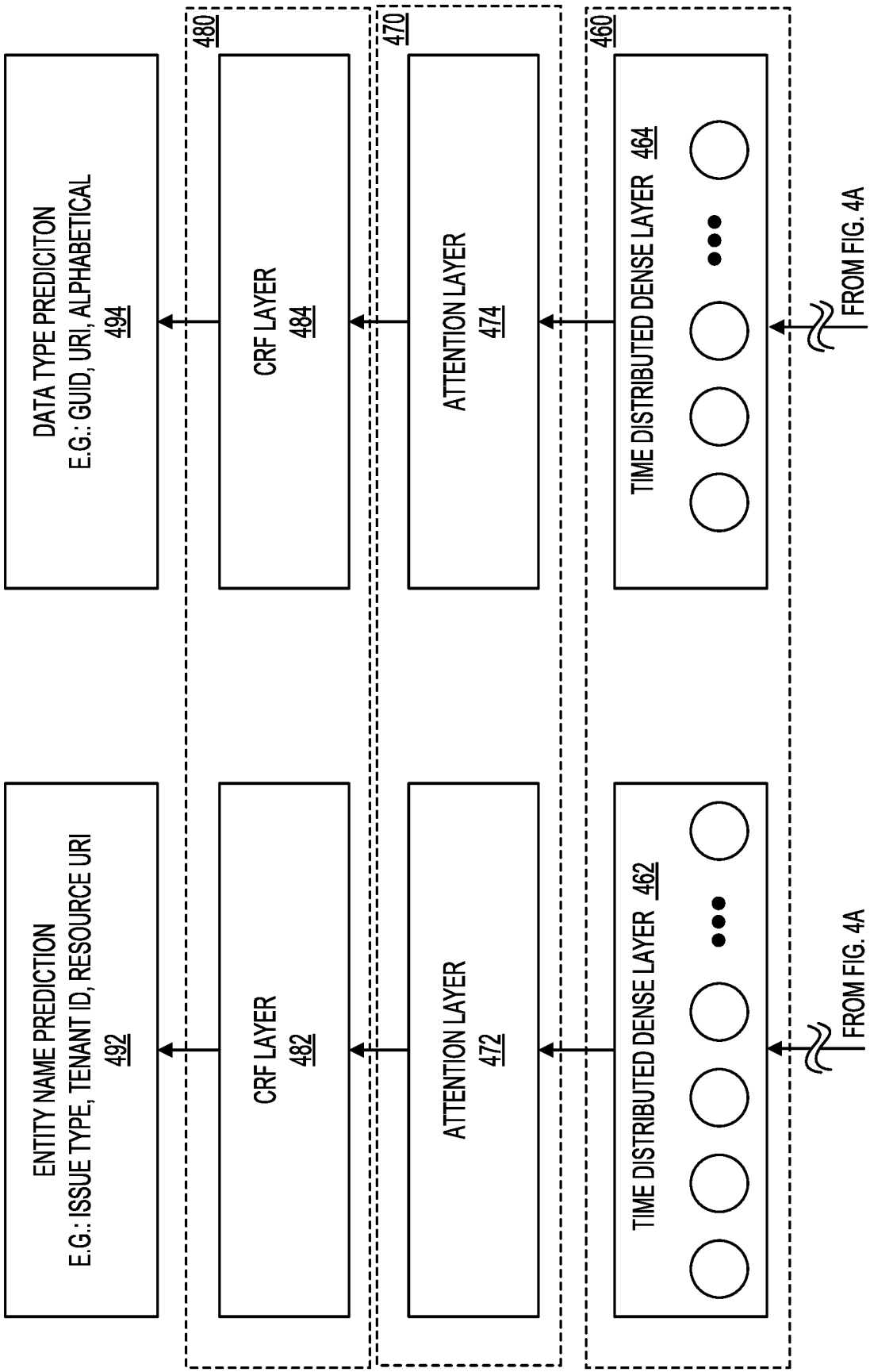
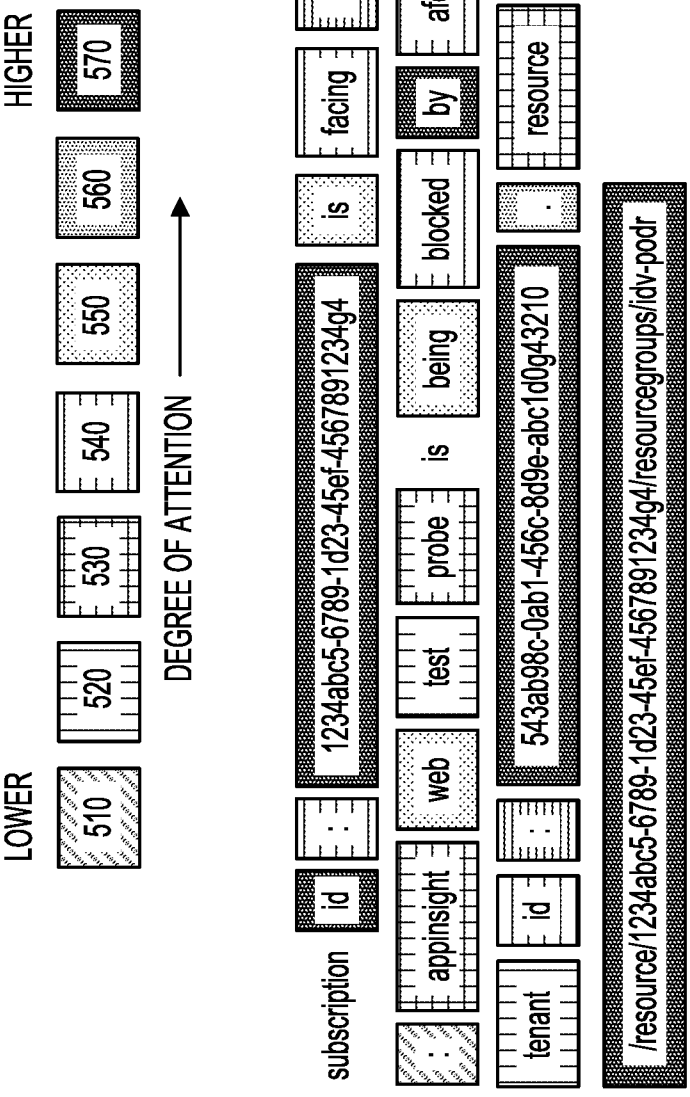

480
470
460
DATA TYPE PREDICITON
E.G.: GUID, URI, ALPHABETICAL
494
CRF LAYER
484
ATTENTION LAYER
474
TIME DISTRIBUTED DENSE LAYER 464
FROM FIG. 4A
ENTITY NAME PREDICTION
E.G.: ISSUE TYPE, TENANT ID, RESOURCE URI
492
CRF LAYER
482
ATTENTION LAYER
472
TIME DISTRIBUTED DENSE LAYER 462
FROM FIG. 4A

FIG. 4B


LOWER
HIGHER
510
520
530
540
550
560
570
DEGREE OF ATTENTION
subscription
id
:
1234abc5-6789-1d23-45ef-456789123g4
is
facing
issue
:
appinsight
web
test
probe
is
being
blocked
by
afd
waf
rule.
tenant
id
:
543ab98c-0ab1-456c-8d9e-abc1d0g43210
.
resource
uri
/resource/1234abc5-6789-1d23-45ef-456789123g4/resourcegroups/idv-podr

500

FIG. 5

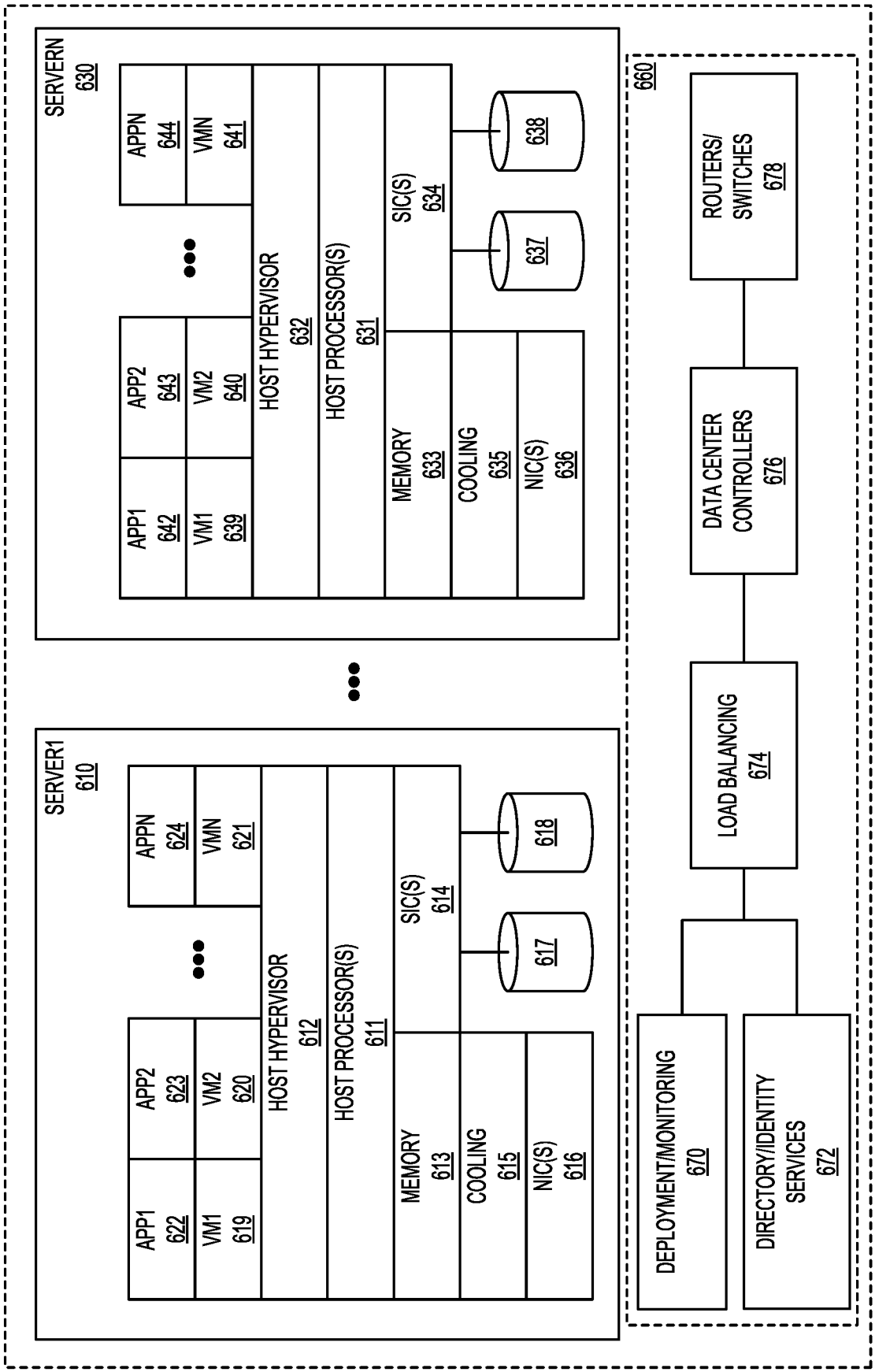
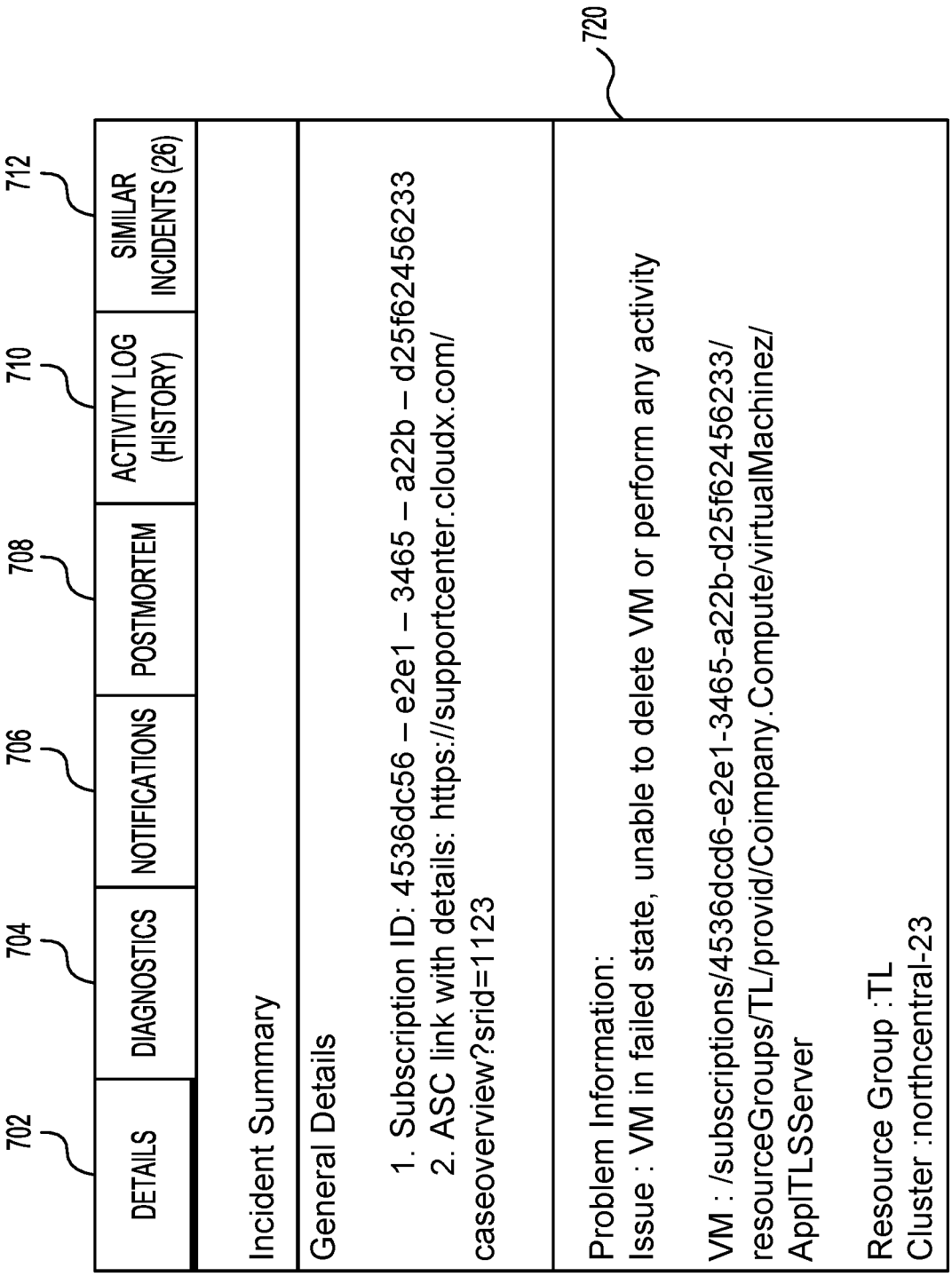
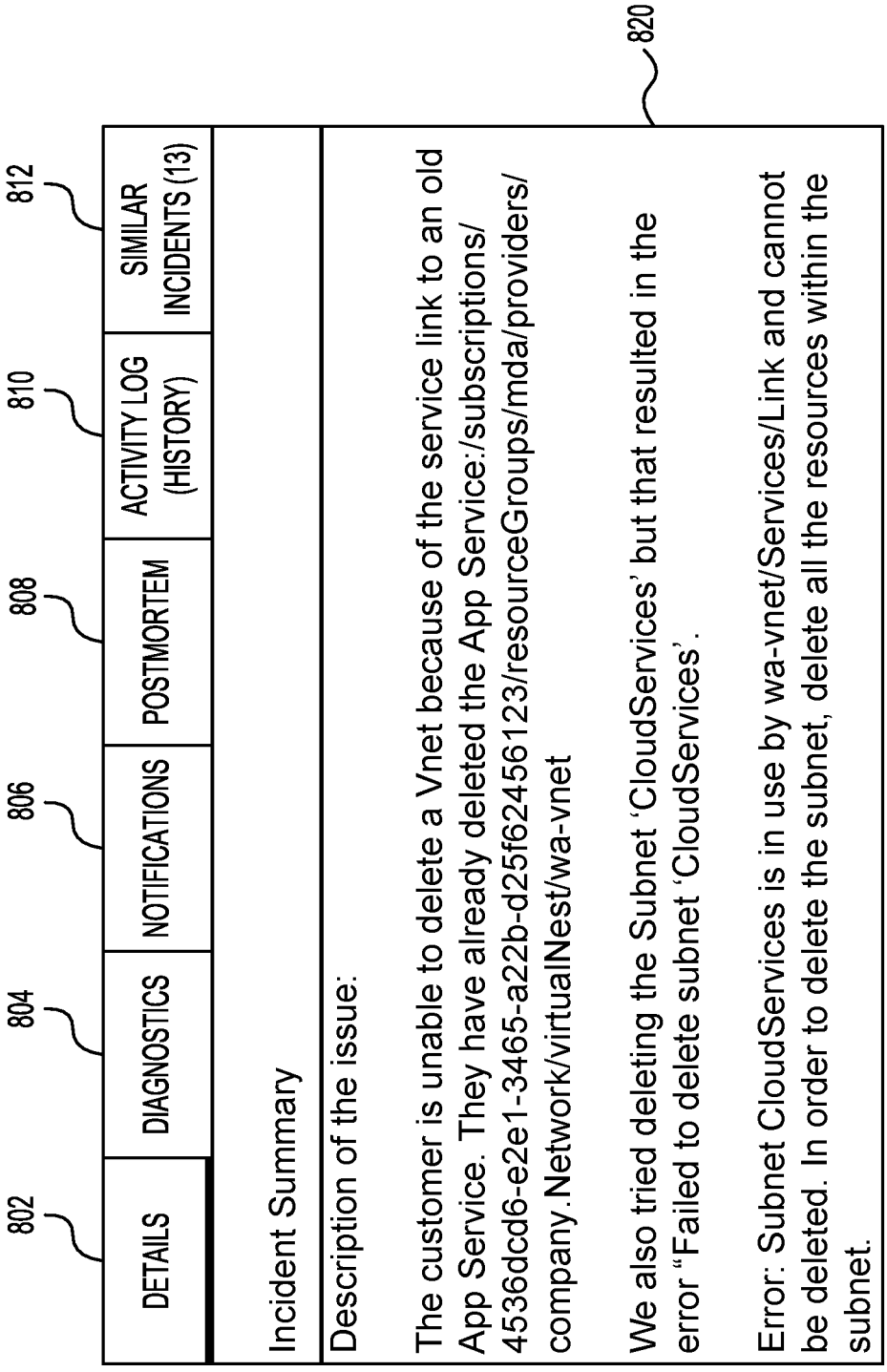

600
SERVER1
610
APP1
622
VM1
619
APP2
623
VM2
620
APPN
624
VMN
621
HOST HYPERVISOR
612
HOST PROCESSOR(S)
611
MEMORY
613
COOLING
615
NIC(S)
616
SIC(S)
614
617
618
SERVERN
630
APP1
642
VM1
639
APP2
643
VM2
640
APPN
644
VMN
641
HOST HYPERVISOR
632
HOST PROCESSOR(S)
631
MEMORY
633
COOLING
635
NIC(S)
636
SIC(S)
634
637
638
660
DEPLOYMENT/MONITORING
670
DIRECTORY/IDENTITY
SERVICES
672
LOAD BALANCING
674
DATA CENTER
CONTROLLERS
676
ROUTERS/
SWITCHES
678

FIG. 6


DETAILS
DIAGNOSTICS
NOTIFICATIONS
POSTMORTEM
ACTIVITY LOG (HISTORY)
SIMILAR INCIDENTS (26)
702
704
706
708
710
712
720
Incident Summary
General Details
1. Subscription ID: 4536dc56 – e2e1 – 3465 – a22b – d25f62456233
2. ASC link with details: https://supportcenter.cloudx.com/caseoverview?srid=1123
Problem Information:
Issue : VM in failed state, unable to delete VM or perform any activity
VM : /subscriptions/4536dcd6-e2e1-3465-a22b-d25f62456233/resourceGroups/TL/provid/Coimpany.Compute/virtualMachinez/ApplTLSServer
Resource Group :TL
Cluster :northcentral-23

700

FIG. 7


802
804
806
808
810
812
820
DETAILS
DIAGNOSTICS
NOTIFICATIONS
POSTMORTEM
ACTIVITY LOG (HISTORY)
SIMILAR INCIDENTS (13)
Incident Summary
Description of the issue:
The customer is unable to delete a Vnet because of the service link to an old App Service. They have already deleted the App Service:/subscriptions/4536dcd6-e2e1-3465-a22b-d25f62456123/resourceGroups/mda/providers/company.Network/virtualNest/wa-vnet
We also tried deleting the Subnet 'CloudServices' but that resulted in the error "Failed to delete subnet 'CloudServices'.
Error: Subnet CloudServices is in use by wa-vnet/Services/Link and cannot be deleted. In order to delete the subnet, delete all the resources within the subnet.

800

FIG. 8

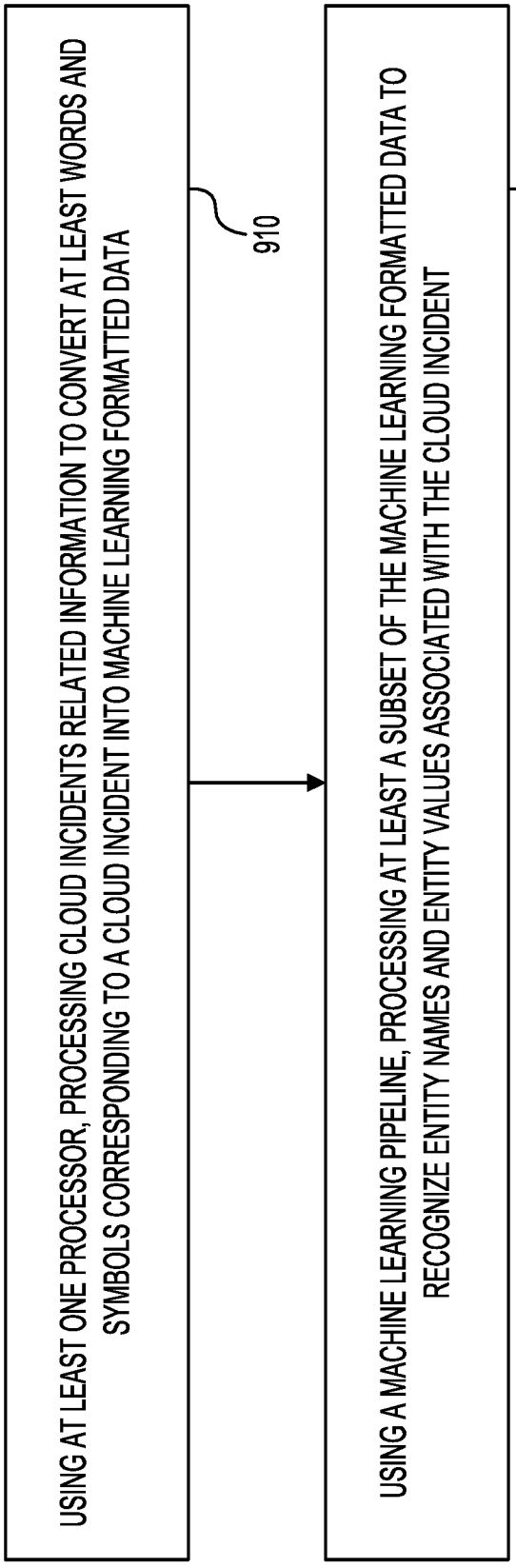

USING AT LEAST ONE PROCESSOR, PROCESSING CLOUD INCIDENTS RELATED INFORMATION TO CONVERT AT LEAST WORDS AND SYMBOLS CORRESPONDING TO A CLOUD INCIDENT INTO MACHINE LEARNING FORMATTED DATA
910
USING A MACHINE LEARNING PIPELINE, PROCESSING AT LEAST A SUBSET OF THE MACHINE LEARNING FORMATTED DATA TO RECOGNIZE ENTITY NAMES AND ENTITY VALUES ASSOCIATED WITH THE CLOUD INCIDENT
920
900

FIG. 9

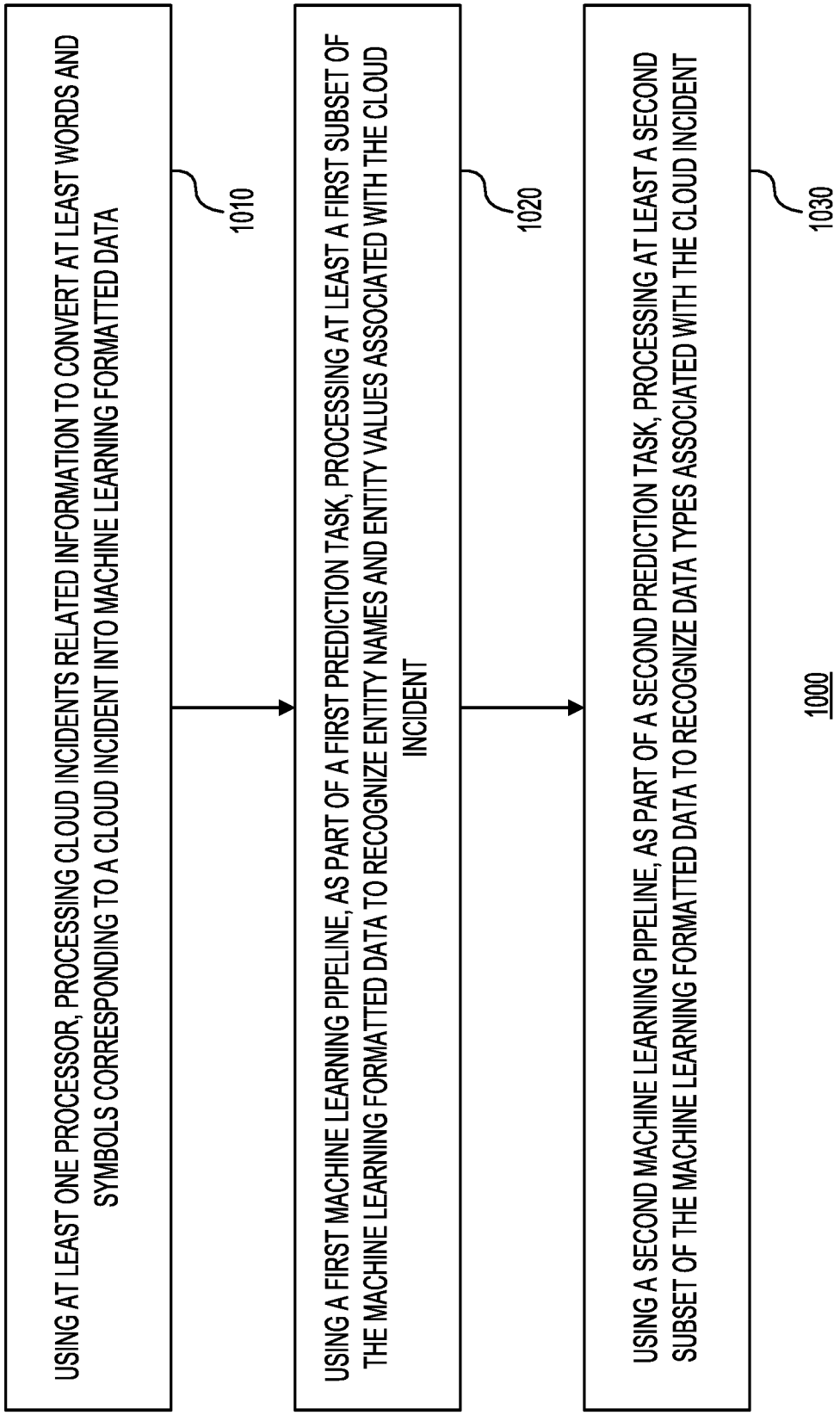

USING AT LEAST ONE PROCESSOR, PROCESSING CLOUD INCIDENTS RELATED INFORMATION TO CONVERT AT LEAST WORDS AND SYMBOLS CORRESPONDING TO A CLOUD INCIDENT INTO MACHINE LEARNING FORMATTED DATA
1010
USING A FIRST MACHINE LEARNING PIPELINE, AS PART OF A FIRST PREDICTION TASK, PROCESSING AT LEAST A FIRST SUBSET OF THE MACHINE LEARNING FORMATTED DATA TO RECOGNIZE ENTITY NAMES AND ENTITY VALUES ASSOCIATED WITH THE CLOUD INCIDENT
1020
USING A SECOND MACHINE LEARNING PIPELINE, AS PART OF A SECOND PREDICTION TASK, PROCESSING AT LEAST A SECOND SUBSET OF THE MACHINE LEARNING FORMATTED DATA TO RECOGNIZE DATA TYPES ASSOCIATED WITH THE CLOUD INCIDENT
1030
1000

FIG. 10

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2021/032130

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06Q10/10 G06F40/295
ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06Q G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	WANG XI ET AL: "Multitask learning for biomedical named entity recognition with cross-sharing structure", BMC BIOINFORMATICS, vol. 20, no. 1, 16 August 2019 (2019-08-16), pages 1-13, XP055832181, DOI: 10.1186/s12859-019-3000-5 Retrieved from the Internet: URL:https://bmcbioinformatics.biomedcentral.com/track/pdf/10.1186/s12859-019-3000-5.pdf> page 1 - page 4 ----- -/--	1-15

[X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
25 August 2021	01/09/2021

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Krafft, Gerald

2

Form PCT/ISA/210 (second sheet) (April 2005)

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	YANG JIANLIANG ET AL: "Information Extraction from Electronic Medical Records Using Multitask Recurrent Neural Network with Contextual Word Embedding", APPLIED SCIENCES, vol. 9, no. 18, 4 September 2019 (2019-09-04), pages 1-16, XP055832198, DOI: 10.3390/app9183658 Retrieved from the Internet: URL:https://www.researchgate.net/publication/335623420_Information_Extraction_from_Electronic_Medical_Records_Using_Multitask_Recurrent_Neural_Network_with_Contextual_Word_Embedding/fulltext/5d7103cea6fdcc9961aff82e/Information-Extraction-from-Electronic-Medical-Records-Using-Multitask-Recurrent-Neural-> page 4 - page 9	1-15
X	SINGLA KARAN ET AL: "Multitask Learning for Darpa Lorelei's Situation Frame Extraction Task", ICASSP 2020 - 2020 IEEE INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH AND SIGNAL PROCESSING (ICASSP), IEEE, 4 May 2020 (2020-05-04), pages 8149-8153, XP033794362, DOI: 10.1109/ICASSP40776.2020.9054710 [retrieved on 2020-04-01] page 8150 - page 8151	1-15
X	Sharma Nikita: "How to build deep neural network for custom NER with Keras", INTERNET ARTICLE, 29 December 2019 (2019-12-29), XP055832564, Retrieved from the Internet: URL:https://confusedcoders.com/data-science/deep-learning/how-to-build-deep-neural-network-for-custom-ner-with-keras [retrieved on 2021-08-17] page 1 - page 6	1-15

2