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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
20 August 2020 (20.08.2020)

WIPO | PCT

(10) International Publication Number
WO 2020/167586 A1

(51) International Patent Classification:
G06F 40/221 (2020.01) *G06F 40/284* (2020.01)
G06F 40/237 (2020.01) *G06F 40/30* (2020.01)

(21) International Application Number:
PCT/US2020/017064

(22) International Filing Date:
06 February 2020 (06.02.2020)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
62/804,083 11 February 2019 (11.02.2019) US

(71) Applicant: **DB CYBERTECH, INC.** [US/US]; 15015 Avenue of Science, Suite 150, San Diego, California 92128 (US).

(72) Inventors: **ROSENBERG, David A.**; c/o DB Cybertech, Inc., 15015 Avenue of Science, Suite 150, San Diego, California 92128 (US). **ENG, Vincent**; c/o DB Cybertech, Inc., 15015 Avenue of Science, Suite 150, San Diego, California 92128 (US).

(74) Agent: **CHENG, Jonathan D.** et al.; c/o Procopio, Cory, Hargreaves & Savitch LLP, 525 B Street, Suite 2200, San Diego, California 92101 (US).

(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, 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).

(54) Title: AUTOMATED DATA DISCOVERY FOR CYBERSECURITY

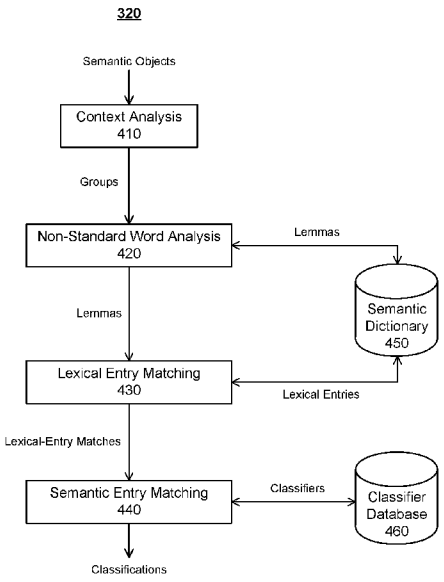


FIG. 4

(57) Abstract: Automated data discovery for cybersecurity. In an embodiment, a plurality of semantic objects, representing in-context textual components of statement(s) in a data-access language, captured from network traffic, are received. The textual components are parsed into group(s) of word(s). The group(s) are then matched to lemmas in a semantic dictionary to produce matched lemma(s), and the matched lemma(s) are matched to lexical entry(ies) in the semantic dictionary to produce matched lexical entry(ies). The matched lexical entry(ies) are matched to classifier(s) in a classifier database to produce classification(s). Each classification represents a possibility that the statement(s) access data within that classification. The classification(s) may be stored in association with their context for quantitative and/or qualitative analysis of the types and locations of data within a network.

WO 2020/167586 A1

[Continued on next page]

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))*

AUTOMATED DATA DISCOVERY FOR CYBERSECURITY

CROSS-REFERENCE TO RELATED APPLICATIONS

[1] This application claims priority to U.S. Provisional Patent App. No. 62/804,083, filed on February 11, 2019, and titled "Meta-data Classifier," which is hereby incorporated herein by reference as if set forth in full.

BACKGROUND

[2] Field of the Invention

[3] The embodiments described herein are generally directed to automated data discovery for cybersecurity, and, more particularly, to automated discovery and classification of data within a network to improve cybersecurity within the network.

[4] Description of the Related Art

[5] It is becoming increasingly valuable for entities (e.g., companies, government agencies, etc.) to understand not only what information they hold, but where they hold it. For example, in the event of a cyberattack or other data breach, it is important to know both what personally identifiable information (PII) the entity controls and where that PII is held.

[6] Data may exist in several forms and in many places within an entity's infrastructure. For example, structured data primarily resides in databases, and frequently in a relational database management system (RDBMS). Unstructured data, on the other hand, may exist across a number of files (e.g., documents, spreadsheets, etc.) spread across numerous systems. There is also a category of data called "semi-structured data," such as Not Only Structured Query Language (NoSQL), which eschews the rigid data consistency of RDBMS in favor of increased performance and scalability. Unless otherwise indicated, the term "structured data," when used herein, should be construed to include both structured and semi-structured data.

[7] Conventionally, an entity would ascertain the existence, classification, and location of structured data using a very labor-intensive and error-prone process of:

[8] (a) Manually identifying its databases via various registries and an exhaustive search;

[9] (b) Obtaining credentials for the identified databases; and

[10] (c) Probing, identifying, and classifying the information in each identified database, using the obtained credentials.

[11] It would be beneficial to have a more efficient means for discovering (e.g., identifying, locating, classifying, etc.) structured data (e.g., in relational and/or NoSQL databases), to aid in the prevention, detection, and/or mitigation of cyberattacks and other forms of data loss. Such means should enable entities to continuously monitor any access, modification, and movement of structured data, particularly when such activity falls outside of behavioral norms.

SUMMARY

[12] Accordingly, systems, methods, and non-transitory computer-readable media are disclosed to prevent, detect, and/or mitigate cyberattacks and/or other forms of data loss through a more efficient, automated means of discovering structured data.

[13] In an embodiment, a method is disclosed that comprises using at least one hardware processor to: receive a plurality of semantic objects, representing textual components of at least one or more statements, in a data-access language, captured from network traffic, within a context; parse the textual components into one or more groups of one or more words; matching the one or more the groups to lemmas in a semantic dictionary to produce one or more matched lemmas representing the one or more groups; matching the one or more matched lemmas to lexical entries in the semantic dictionary to produce one or more matched lexical entries; matching the one or more matched lexical entries to one or more classifiers in a classifier database to produce one or more classifications, wherein each of the one or more classifications represents a possibility that at least one of the one or more statements accesses data within that classification; and storing the one or more classifications in association with the context in an analytic database.

[14] The plurality of semantic objects may represent textual components of one or more statements in the data-access language and one or more comments in the data-access language. The textual components may comprise one or more identifiers utilized by one or more statements in the data-access language. Parsing the one or more identifiers into the one or more groups of one or more words may comprise expanding one non-standard identifier into two or more standard words. The textual components may be parsed into the one or more groups of one or more words according to the context. The context may comprise a database service. Parsing the textual components into one or more groups of one or more words may comprise, for one or more passes, semantically analyzing textual components within a window that moves or changes between passes.

[15] Matching the one or more matched lemmas to lexical entries in the semantic dictionary may comprise constraining the matching according to one or more of the following constraints: each of the one or more matched lemmas is matched, at most, once to the lexical entries in the semantic dictionary; a total number of characters of the one or more lemmas, matched to the lexical entries in the semantic dictionary, is maximized; and any of the one or more matched lemmas that are not matched to the lexical entries in the semantic dictionary are skipped. The total number of characters of the one or more lemmas may be determined based on a written representation associated with each of the one or more matched lexical entries in the semantic dictionary, wherein each written representation is in a single human language regardless of a human language of the groups which were matched to the one or more lemmas.

[16] Each of the one or more classifiers may comprise a containment hierarchy that comprises one or more categories contained within the classifier, one or more entry groups contained within each of the one or more categories, one or more entry sets contained within each of the one or more entry groups, and one or more lexical entries contained within each of the one or more entry sets. Any of the one or more entry groups that contains a plurality of lexical entries may be associated with a match window, such that the entry group is only matched if the one or more matched lexical entries comprise all of the contained plurality of lexical entries within the match window.

[17] The method may further comprise generating a quantitative score for each of the one or more classifications, wherein the quantitative score indicates a likelihood that the one or more statements implicate the respective classification within the context. Each of the one or more classifications may comprise a containment hierarchy that comprises one or more categories contained within the classification, one or more entry groups contained within each of the one or more categories, one or more entry sets contained within each of the one or more entry groups, and one or more lexical entries contained within each of the one or more entry sets, and generating the quantitative score for each of the one or more classifications may comprise: generating a local score for each of the one or more lexical entries; generating an entry-set score for each of the one or more entry sets based on the local scores for the one or more lexical entries contained within that entry set; generating an entry-group score for each of the one or more entry groups based on the entry-set scores for the one or more entry sets contained within that entry group; generating a category score for each of the one or more categories based on the entry-group scores for the one or more entry groups contained within that category; and generating a classification score for each of the one or more

classifications based on the category scores for the one or more categories contained within that classification. The local score for each of the one or more lexical entries may be calculated based on an actual number of times that the lexical entry occurs, an expected number of times that the lexical entry is expected to occur, and a weight. The entry-set score for each of the one or more entry sets may be calculated based on a maximum local score for the one or more lexical entries contained with that entry set, and a weight. The entry-group score for each of the one or more entry groups may be calculated based on: if the entry group contains only a single entry set, the entry-set score for the single entry set, and a weight; and, if the entry group contains a plurality of entry sets, an accumulated probability calculated based on the local scores of the one or more lexical entries contained in each of the plurality of entry sets, and a weight. The category score for each of the one or more categories may be calculated based on a maximum entry-group score for the one or more entry groups contained with that category, and a weight. The classification score for each of the one or more classifications may be calculated based on a maximum category score for the one or more categories contained with that classification. The method may further comprise filtering out at least one of the one or more categories contained within one or more classifications based on the category score calculated for the at least one category.

[18] The method may further comprise filtering out at least one of the one or more classifications by abbreviating at least one of the one or more classifiers to elide matches to the at least one classifier that produce the at least one classification.

[19] The method may be embodied in executable software modules of a processor-based system, such as a server, and/or in executable instructions stored in a non-transitory computer-readable medium.

BRIEF DESCRIPTION OF THE DRAWINGS

[20] The details of the present invention, both as to its structure and operation, may be gleaned in part by study of the accompanying drawings, in which like reference numerals refer to like parts, and in which:

[21] FIG. 1 illustrates an example infrastructure, in which one or more of the processes described herein, may be implemented, according to an embodiment;

[22] FIG. 2 illustrates an example processing system, by which one or more of the processed described herein, may be executed, according to an embodiment;

[23] FIG. 3 illustrates an example process for data discovery, according to an embodiment;

[24] FIG. 4 illustrates an example process for data classification, according to an embodiment;

[25] FIG. 5 illustrates an example classifier, according to an embodiment;

[26] FIG. 6 illustrates an example algorithm for quantitative scoring, according to an embodiment; and

[27] FIG. 7 illustrates an example algorithm for qualitative filtering, according to an embodiment.

DETAILED DESCRIPTION

[28] In an embodiment, systems, methods, and non-transitory computer-readable media are disclosed to prevent, detect, and/or mitigate cyberattacks and/or other forms of data loss through an improved means of discovering structured data. The disclosed embodiments may enable entities that control data to continuously monitor access, modification, and/or movement of that data, including when such activity falls outside of behavioral norms. After reading this description, it will become apparent to one skilled in the art how to implement the invention in various alternative embodiments and alternative applications. However, although various embodiments of the present invention will be described herein, it is understood that these embodiments are presented by way of example and illustration only, and not limitation. As such, this detailed description of various embodiments should not be construed to limit the scope or breadth of the present invention as set forth in the appended claims.

[29] 1. System Overview

[30] 1.1. Example Infrastructure

[31] FIG. 1 illustrates an example infrastructure for capturing and analyzing interactions between two or more network agents, according to an embodiment. The infrastructure comprises a platform 110 that comprises a capture/analysis system 112 and a detection system 114. Capture/analysis system 112 may be communicatively connected to one or more network taps 120 on network paths 140 between two or more network agents 130. The network comprising network agents 130 may comprise any type of network, including an intranet and/or the Internet, and network agents 130 may communicate using

any known communication protocols, such as Transmission Control Protocol (TCP), Internet Protocol (IP), Hypertext Transfer Protocol (HTTP), Secure HTTP (HTTPS), File Transfer Protocol (FTP), and/or the like. While only a single platform 110, a single capture/analysis system 112, a single detection system 114, a single network tap 120, and two network agents 130 are illustrated, the infrastructure may comprise any number and mixture of platforms 110, capture/analysis systems 112, detection systems 114, and network taps 120, and any plurality of network agents 130.

[32] Each network tap 120 is positioned within a network so as to capture data packets that are sent within network traffic between two or more network agents 130. For example, network tap 120 may be positioned on a network switch in network path 140 between network agents 130A and 130B. Networks taps may be implemented in software, hardware, or a combination of hardware and software. Network agents 130 may comprise any type and mixture of device(s) that are capable of network communication, including servers, desktop computers, mobile devices, Internet of Things (IoT) devices, electronic kiosks, and/or the like. The network switch is configured to relay raw data packets sent from network agent 130A to network agent 130B, and to relay raw data packets sent from network agent 130B to network agent 130A. The network tap 120 on the network switch is configured to copy all of these raw data packets, and transmit the copies of these raw data packets to capture/analysis system 112 via a communication path 150. While network tap 120 may occasionally miss data packets, network tap 120 will capture substantially all network traffic and transmit that network traffic as raw data packets to capture/analysis system 112.

[33] Capture/analysis system 112 receives the raw data packets, copied by network tap 120 from network traffic between a plurality of network agents 130, and assembles and analyzes the raw data packets to produce a semantic model of the network traffic. This capture-and-analysis process is described in detail in U.S. Patent Nos. 9,100,291, issued on August 4, 2016, 9,185,125, issued on November 10, 2015, and 9,525,642, issued on December 20, 2016, which are all hereby incorporated herein by reference as if set forth in full, and collectively referred to herein as "the prior patents." It should be understood that, in these prior patents, the capture-and-analysis device corresponds to capture/analysis system 112 described herein, and that the detector corresponds to detection system 114 described herein.

[34] Detection system 114 consumes the semantic model(s) of network traffic that are produced by capture/analysis system 112. Embodiments of this consumption by detection system 114 are the primary focus of the present disclosure. Thus, it should be understood

that processes disclosed herein may be implemented by detection system 114. Detection system 114 may be a software module that communicatively interfaces with capture/analysis system 112, via any known means (e.g., an application programming interface (API), shared memory, etc.), to receive the semantic models of network traffic. Detection system 114 may be executed by the same processing device as capture/analysis system 112 or a different processing device than capture/analysis system 112. In addition, while capture/analysis system 112 and detection system 114 are shown as being comprised within a single platform 110, capture/analysis system 112 and detection system 114 may instead be comprised in separate platforms and/or may be controlled by different operators. In other words, while detection system 114 utilizes the output of capture/analysis system 112, these two systems are not required to be under control of the same entity. However, for purposes of security, it may be beneficial for these two systems 112 and 114 to be under control of the same entity.

[35] 1.2. Example Processing Device

[36] FIG. 2 is a block diagram illustrating an example wired or wireless system 200 that may be used in connection with various embodiments described herein. For example, system 200 may be used as or in conjunction with one or more of the processes described herein (e.g., to store and/or execute software implementing the disclosed processes), and may represent components of platform 110, including capture/analysis system 112 and detection system 114, network tap 120, network agents 130, and/or any other processing devices described herein. In an embodiment in which capture/analysis system 112 and detection system 114 are implemented as software modules, one or more systems 200 may be used to store and execute these software modules. System 200 can be a server, a conventional personal computer, or any other processor-enabled device that is capable of wired or wireless data communication. Other computer systems and/or architectures may be also used, as will be clear to those skilled in the art.

[37] System 200 preferably includes one or more processors, such as processor 210. Additional processors may be provided, such as an auxiliary processor to manage input/output, an auxiliary processor to perform floating-point mathematical operations, a special-purpose microprocessor having an architecture suitable for fast execution of signal-processing algorithms (e.g., digital-signal processor), a slave processor subordinate to the main processing system (e.g., back-end processor), an additional microprocessor or controller for dual or multiple processor systems, and/or a coprocessor. Such auxiliary processors may be discrete processors or may be integrated with processor 210. Examples of processors

which may be used with system 200 include, without limitation, the Pentium® processor, Core i7® processor, and Xeon® processor, all of which are available from Intel Corporation of Santa Clara, California.

[38] Processor 210 is preferably connected to a communication bus 205. Communication bus 205 may include a data channel for facilitating information transfer between storage and other peripheral components of system 200. Furthermore, communication bus 205 may provide a set of signals used for communication with processor 210, including a data bus, address bus, and/or control bus (not shown). Communication bus 205 may comprise any standard or non-standard bus architecture such as, for example, bus architectures compliant with industry standard architecture (ISA), extended industry standard architecture (EISA), Micro Channel Architecture (MCA), peripheral component interconnect (PCI) local bus, standards promulgated by the Institute of Electrical and Electronics Engineers (IEEE) including IEEE 488 general-purpose interface bus (GPIB), IEEE 696/S-100, and/or the like.

[39] System 200 preferably includes a main memory 215 and may also include a secondary memory 220. Main memory 215 provides storage of instructions and data for programs executing on processor 210, such as one or more of the functions and/or modules discussed herein. It should be understood that programs stored in the memory and executed by processor 210 may be written and/or compiled according to any suitable language, including without limitation C/C++, Java, JavaScript, Perl, Visual Basic, .NET, and the like. Main memory 215 is typically semiconductor-based memory such as dynamic random access memory (DRAM) and/or static random access memory (SRAM). Other semiconductor-based memory types include, for example, synchronous dynamic random access memory (SDRAM), Rambus dynamic random access memory (RDRAM), ferroelectric random access memory (FRAM), and the like, including read only memory (ROM).

[40] Secondary memory 220 may optionally include an internal medium 225 and/or a removable medium 230. Removable medium 230 is read from and/or written to in any well-known manner. Removable storage medium 230 may be, for example, a magnetic tape drive, a compact disc (CD) drive, a digital versatile disc (DVD) drive, other optical drive, a flash memory drive, and/or the like.

[41] Secondary memory 220 is a non-transitory computer-readable medium having computer-executable code (e.g., disclosed software representing capture/analysis system 112 and/or detection system 114) and/or other data stored thereon. The computer software or data

stored on secondary memory 220 is read into main memory 215 for execution by processor 210.

[42] In alternative embodiments, secondary memory 220 may include other similar means for allowing computer programs or other data or instructions to be loaded into system 200. Such means may include, for example, a communication interface 240, which allows software and data to be transferred from external storage medium 245 to system 200. Examples of external storage medium 245 may include an external hard disk drive, an external optical drive, an external magneto-optical drive, and/or the like. Other examples of secondary memory 220 may include semiconductor-based memory, such as programmable read-only memory (PROM), erasable programmable read-only memory (EPROM), electrically erasable read-only memory (EEPROM), and flash memory (block-oriented memory similar to EEPROM).

[43] As mentioned above, system 200 may include a communication interface 240. Communication interface 240 allows software and data to be transferred between system 200 and external devices (e.g. printers), networks, or other information sources. For example, computer software or executable code may be transferred to system 200 from a network server (e.g., platform 110) via communication interface 240. Examples of communication interface 240 include a built-in network adapter, network interface card (NIC), Personal Computer Memory Card International Association (PCMCIA) network card, card bus network adapter, wireless network adapter, Universal Serial Bus (USB) network adapter, modem, a wireless data card, a communications port, an infrared interface, an IEEE 1394 fire-wire, and any other device capable of interfacing system 200 with a network (e.g., network(s) 120) or another computing device. Communication interface 240 preferably implements industry-promulgated protocol standards, such as Ethernet IEEE 802 standards, Fiber Channel, digital subscriber line (DSL), asynchronous digital subscriber line (ADSL), frame relay, asynchronous transfer mode (ATM), integrated digital services network (ISDN), personal communications services (PCS), transmission control protocol/Internet protocol (TCP/IP), serial line Internet protocol/point to point protocol (SLIP/PPP), and so on, but may also implement customized or non-standard interface protocols as well.

[44] Software and data transferred via communication interface 240 are generally in the form of electrical communication signals 255. These signals 255 may be provided to communication interface 240 via a communication channel 250. In an embodiment, communication channel 250 may be a wired or wireless network, or any variety of other communication links. Communication channel 250 carries signals 255 and can be

implemented using a variety of wired or wireless communication means including wire or cable, fiber optics, conventional phone line, cellular phone link, wireless data communication link, radio frequency ("RF") link, or infrared link, just to name a few.

[45] Computer-executable code (e.g., computer programs, such as implementations of the disclosed software processes) is stored in main memory 215 and/or secondary memory 220. Computer programs can also be received via communication interface 240 and stored in main memory 215 and/or secondary memory 220. Such computer programs, when executed, enable system 200 to perform the various functions of the disclosed embodiments as described elsewhere herein.

[46] In this description, the term "computer-readable medium" is used to refer to any non-transitory computer-readable storage media used to provide computer-executable code and/or other data to or within system 200. Examples of such media include main memory 215, secondary memory 220 (including internal memory 225, removable medium 230, and external storage medium 245), and any peripheral device communicatively coupled with communication interface 240 (including a network information server or other network device). These non-transitory computer-readable media are means for providing executable code, programming instructions, software, and/or other data to system 200.

[47] In an embodiment that is implemented using software, the software may be stored on a computer-readable medium and loaded into system 200 by way of removable medium 230, I/O interface 235, or communication interface 240. In such an embodiment, the software is loaded into system 200 in the form of electrical communication signals 255. The software, when executed by processor 210, preferably causes processor 210 to perform one or more of the processes described herein.

[48] In an embodiment, I/O interface 235 provides an interface between one or more components of system 200 and one or more input and/or output devices. Example input devices include, without limitation, sensors, keyboards, touch screens or other touch-sensitive devices, biometric sensing devices, computer mice, trackballs, pen-based pointing devices, and/or the like. Examples of output devices include, without limitation, other processing devices, cathode ray tubes (CRTs), plasma displays, light-emitting diode (LED) displays, liquid crystal displays (LCDs), printers, vacuum fluorescent displays (VFDs), surface-conduction electron-emitter displays (SEDs), field emission displays (FEDs), and/or the like. In some cases, an input and output device may be combined, such as in the case of a touch panel display (e.g., in a smartphone, tablet, or other mobile device).

[49] System 200 may also include optional wireless communication components that facilitate wireless communication over a voice network and/or a data network (e.g., in the case of user system 130). The wireless communication components comprise an antenna system 270, a radio system 265, and a baseband system 260. In system 200, radio frequency (RF) signals are transmitted and received over the air by antenna system 270 under the management of radio system 265.

[50] In an embodiment, antenna system 270 may comprise one or more antennae and one or more multiplexors (not shown) that perform a switching function to provide antenna system 270 with transmit and receive signal paths. In the receive path, received RF signals can be coupled from a multiplexor to a low noise amplifier (not shown) that amplifies the received RF signal and sends the amplified signal to radio system 265.

[51] In an alternative embodiment, radio system 265 may comprise one or more radios that are configured to communicate over various frequencies. In an embodiment, radio system 265 may combine a demodulator (not shown) and modulator (not shown) in one integrated circuit (IC). The demodulator and modulator can also be separate components. In the incoming path, the demodulator strips away the RF carrier signal leaving a baseband receive audio signal, which is sent from radio system 265 to baseband system 260.

[52] If the received signal contains audio information, then baseband system 260 decodes the signal and converts it to an analog signal. Then the signal is amplified and sent to a speaker. Baseband system 260 also receives analog audio signals from a microphone. These analog audio signals are converted to digital signals and encoded by baseband system 260. Baseband system 260 also encodes the digital signals for transmission and generates a baseband transmit audio signal that is routed to the modulator portion of radio system 265. The modulator mixes the baseband transmit audio signal with an RF carrier signal, generating an RF transmit signal that is routed to antenna system 270 and may pass through a power amplifier (not shown). The power amplifier amplifies the RF transmit signal and routes it to antenna system 270, where the signal is switched to the antenna port for transmission.

[53] Baseband system 260 is also communicatively coupled with processor 210, which may be a central processing unit (CPU). Processor 210 has access to data storage areas 215 and 220. Processor 210 is preferably configured to execute instructions (i.e., computer programs, such as the disclosed software) that can be stored in main memory 215 or secondary memory 220. Computer programs can also be received from baseband processor 260 and stored in main memory 210 or in secondary memory 220, or executed upon receipt.

Such computer programs, when executed, enable system 200 to perform the various processes described herein.

[54] 2. Process Overview

[55] Embodiments of processes for automated data discovery will now be described in detail. It should be understood that the described processes may be embodied in one or more software modules that are executed by one or more hardware processors (e.g., processor 210), for example, as capture/analysis system 112 and/or detection system 114. The described processes may be implemented as instructions represented in source code, object code, and/or machine code. These instructions may be executed directly by the hardware processor(s), or alternatively, may be executed by a virtual machine operating between the object code and the hardware processors.

[56] Alternatively, the described processes may be implemented as a hardware component (e.g., general-purpose processor, integrated circuit (IC), application-specific integrated circuit (ASIC), digital signal processor (DSP), field-programmable gate array (FPGA) or other programmable logic device, discrete gate or transistor logic, etc.), combination of hardware components, or combination of hardware and software components. To clearly illustrate the interchangeability of hardware and software, various illustrative components, blocks, modules, circuits, and steps are described herein generally in terms of their functionality. Whether such functionality is implemented as hardware or software depends upon the particular application and design constraints imposed on the overall system. Skilled persons can implement the described functionality in varying ways for each particular application, but such implementation decisions should not be interpreted as causing a departure from the scope of the invention. In addition, the grouping of functions within a component, block, module, circuit, or step is for ease of description. Specific functions or steps can be moved from one component, block, module, circuit, or step to another without departing from the invention.

[57] Furthermore, while the processes, described herein, are illustrated with a certain arrangement and ordering of steps, each process may be implemented with fewer, more, or different steps and a different arrangement and/or ordering of steps. In addition, it should be understood that any step, which does not depend on the completion of another step, may be executed before, after, or in parallel with that other independent step, even if the steps are described or illustrated in a particular order.

[58] 2.1. Data Discovery

[59] FIG. 3 illustrates the data flow in a process 300 for automated data discovery, according to an embodiment. Process 300 may be executed by at least one hardware processor (e.g., processor 210) of detection system 114.

[60] In sub-process 310, in-context statements are received and analyzed to produce semantic objects. Sub-process 310 is described in greater detail in the prior patents. As discussed in the prior patents, network traffic (e.g., TCP/IP traffic representing client-server database information) is collected (e.g., extracted "from the wire" in physical and/or virtual environments using network tap(s) 120), and decoded (e.g., at both the level of the communication protocol used and the level of the data-access language used) to produce a semantic model of the collected network traffic. This semantic model may comprise or be used to produce a plurality of semantic objects within context.

[61] In an embodiment, the inputs to sub-process 310 are statements within their contexts. The statements may comprise any statement in any programming language for accessing data (e.g., within a relational database). Examples of data-access languages include, without limitation, Oracle™, MySQL™, SQL Server™, MS Access™, IBM Db2™, dBase™, FoxPro™, and the like. However, for ease and consistency of understanding, the data-access language will primarily be illustrated herein as a form of Structured Query Language (SQL). The outputs from sub-process 310 may be a set of semantic objects representing one or more statements that have been extracted from network traffic (e.g., between network agents 130). Each semantic object may represent one or more textual components (e.g., an identifier used in a SQL statement) of the statement, from which it was derived via sub-process 310, as well as an indication of the semantic context in which each of these textual component(s) appeared. For example, the semantic objects may be logically arranged in a hierarchical parse tree, such that the semantic context of each textual component is indicated or implied by its position within a structured descent into the hierarchical parse tree (e.g., its ancestral and/or descendant nodes).

[62] In sub-process 320, the semantic objects, output by sub-process 310, are received and analyzed by a data classifier to produce classifications, which may also be referred to as "semantic matches." The classifications may be represented and stored within an analytic database 330, which is accessible to sub-processes 340 and/or 350. Sub-processes 340 and 350 represent two beneficial uses of the classifications in analytic database 330. Any detection system 114 may implement one or both of sub-processes 340 and 350, may utilize the classifications in some other way, and/or may simply provide access to the classifications

in analytic database 330 (e.g., for use by another system). In general, it should be understood that detection system 114 may consist of the entirety of process 300 or any subset of the sub-processes in process 300 (e.g., only sub-process 320, only sub-process 340, only sub-process 350, a combination of sub-processes 320, 340, and 350, a combination of sub-processes 320 and 340, a combination of sub-process 320 and 350, a combination of sub-processes 340 and 350, etc.), including subsets that comprise only portions of one or more of the sub-processes in process 300.

[63] In sub-process 340, explicit and/or implicit information, defined by the classifications output by sub-process 320, may be analyzed to generate real numbers for each classification. In an embodiment, each real number represents a classification score that indicates the relative importance of a given classification, matched in sub-process 320 and stored in analytic database 330. These classifications may be filtered (e.g., by their respective classification scores, for example, using one or more score thresholds) to narrow the universe of classifications, available in analytic database 330, to only those classifications that are associated with certain statements and/or contexts that, for example, are associated with activity that falls outside of behavioral norms, and therefore, may represent a cyberattack or other form of data loss.

[64] In sub-process 350, the classifications output by sub-process 320 may be processed using relatively general classification taxonomies, specialized to local environments, to yield selective classifications. This qualitative approach is on par with the quantitative approach in sub-process 340, but with no implicit filtering required. Thus, this qualitative approach in sub-process 350 can be more suitable for compliance-oriented environments than the quantitative approach in sub-process 340.

[65] 2.1.1. Data Classification

[66] FIG. 4 illustrates the data flow in data-classification sub-process 320 within process 300, according to an embodiment. Sub-process 320 may be executed by at least one hardware processor (e.g., processor 210) of detection system 114.

[67] In sub-process 410, the semantic objects, output by sub-process 310 of process 300, are analyzed to produce groups of one or more words for further analysis. Each semantic object may represent an identifier or comment in a data-access language, within a data-access language statement, executing within the context of client-server communication discovered within network traffic. Each semantic object may be analyzed in sub-process 410 to determine one or more word sequences, within the represented statement or comment, that

are likely to convey the intent of the statement or comment, which will be useful for classification. Specifically, in the case of a statement in a data-access language, an identifier (e.g., the name or alias of a database, table, column, etc.) within the statement may be broken down into a word sequence. In the case of a human-language comment, the entire comment may be output as a word sequence consisting of the raw text of the comment. Each word sequence may be referred to herein as a "group." However, it should be understood that each word sequence or group may comprise only a single word or any plurality of words.

[68] As a simple example of sub-process 410, it will be assumed that the statement "SELECT Soc_Sec_No FROM User_Data" is analyzed by sub-process 310 to produce one or more semantic objects. In this case, sub-process 410 may select the identifiers "Soc_Sec_No" (i.e., a column identifier) and "User_Data" (i.e., a relation identifier) for subsequent processing.

[69] In sub-process 420, a non-standard word analysis is applied to each group, produced by sub-process 410. The non-standard word analysis for a group determines the best match between the group (i.e., sequence of one or more words) and a sequence of one or more lemmas in semantic dictionary 450. As an example, semantic dictionary 450 may comprise WordNet™, produced by Princeton University, for the English Language. A lemma is a standard word that is a known key to a set of one or more entries within semantic dictionary 450. In other words, sub-process 420 receives groups of words and standardizes them into sequences of lemmas that can be found in semantic dictionary 450.

[70] Continuing the simple example from above, in sub-process 420, sub-process 420 may expand the identifiers "Soc_Sec_No" and "User_Data" into the following groups: "Soc"; "Sec"; "No"; "User"; and "Data". Then, the group "Soc" may be matched to the lemma "social", the group "Sec" may be matched to the lemma "security", and the group "No" may be matched to the lemma "number". In addition, the group "User" may be matched to the lemma "user", and the group "Data" may be matched to the lemma "data". It should be understood that these are all examples of single-word groups. In other examples, a single group may comprise two or more words. For example, the SQL comment "/* social security number */" would form the group consisting of "social", "security", and "number".

[71] In sub-process 430, sequences of one or more of the lemmas, produced by sub-process 420, are matched to lexical entries in semantic dictionary 450 to produce lexical-entry matches, which could also be referred to as "lexemes." Each lexical-entry match represents a unit of meaning represented by a set of one or more of the lemmas output by sub-process 420. In other words, sub-process 430 maps sequences of lemma(s) to their potential

meanings. Continuing the simple example from above, sub-process 430 may match the sequence of lemmas "Soc", "Sec", and "No" to a lexical entry for "social security number".

[72] In an embodiment, each lexical-entry match may comprise a sequence of one or more tuples. For example, each tuple may comprise a two-tuple with at least the following components: (1) a subsequence of one or more lemmas, produced by sub-process 420, that matched one or more lexical entries in semantic dictionary 450; and (2) a set of one or more references to the one or more lexical entries in semantic dictionary 450 that were matched.

[73] In addition, lexical-entry matching in sub-process 430 may be limited by one or more logical constraints. For example, the constraints may include one or more, including potentially all, of the following:

[74] (1) Each lexical-entry match comprises a subsequence of the input lemmas, with each input lemma matched, at most, once across all matched subsequences;

[75] (2) For the English language, the total number of characters, computed by summing the string sizes of each lemma in a matched subsequence of input lemmas, is maximized considering the written representation of the matched lexical entries from semantic dictionary 450;

[76] (3) For non-English languages, the total number of characters, computed by summing the string sizes of each lemma in a matched subsequence of input lemmas, is maximized considering the translation strings for the matched lexical entries from semantic dictionary 450 (i.e., instead of the written representation as in constraint (2)); and/or

[77] (4) Input lemmas which have no match in semantic dictionary 450 are skipped as semantic white-space (i.e., treated as if they are not present).

[78] One effect of these constraints is that sub-process 430 favors lexical-entry matches (e.g., to either written representations for English-language lemmas or translations for foreign-language lemmas) that contain multiple lemmas and represent the most specific possible lexical entry in semantic dictionary 450. In addition, sub-process 430 skips any non-matches as unimportant. For foreign-language inputs (e.g., statements in SQL or other data-access language written in any non-English language), constraint (3) leverages existing English translations of lexical entries in semantic dictionary 450 (e.g., WordNet™), so that the subsequent semantic entry matching in sub-process 440 can apply the same classifiers regardless of the human language of the groups input to sub-process 410.

[79] In sub-process 440, the lexical-entry matches, output by sub-process 430, are matched against specific classifiers to produce classifications. The classifiers may be defined as records within an extensible classifier database 460. As mentioned above, the resulting classifications from sub-process 440 may be stored in analytic database 330.

[80] In an embodiment, the input to sub-process 440 is a sequence of one or more of the tuples (e.g., two-tuples) that were described above as the output of sub-process 430. The classifications, output by sub-process 440, may comprise a sequence of one or more references to lexical entries in semantic dictionary 450, within a data structure representing each classification.

[81] In an embodiment, each classifier in classifier database 460 may be defined as a containment hierarchy – in which each child in the hierarchy is "contained" by, and hence implicitly associated with, exactly one parent in the hierarchy – in a top-down fashion. FIG. 5 illustrates example classifiers, according to an embodiment. As illustrated, classifier database 460 may comprise one or more classifiers 510. Each classifier 510 represents one entire classification taxonomy, and may have one or more categories 520 as children. Each category 520 may be recursively defined to represent a specific classification asserted by its associated classifier 510, and may have one or more other categories 520 and/or one or more entry groups 530 as children. Each entry group 530 represents a specific way that a category 520 can be inferred from lexical-entry matches, and may have one or more entry sets 540 as children. In addition, an entry group 530 that is able to match to a plurality of lexical entries may have an associated match window (e.g., defined by a specified window size). In this case, if a plurality of lexical entries 550 within the entry group 530 are separated from each other (e.g., by one or more other words, groups, or lemmas), but the separation amount is within the match window, the plurality of lexical entries 550 will collectively match the entry group 530. On the other hand, if the plurality of lexical entries 550 are separated from each other and the separation amount is not within the match window, the plurality of lexical entries 550 will not match the entry group 530. Finally, each entry set 540 may specify one or more lexical entries 550 that are associated with one aspect of the entry group 530 that is the parent of the entry set 540. While the illustrated hierarchy will be used throughout the present description for consistency and ease of understanding, the hierarchy may comprise fewer or more levels than illustrated and described.

[82] In an embodiment, the classification process in sub-process 440 may be limited by one or more logical constraints. For example, the constraints may include one or more,

including potentially all, of the following, for each statement or comment that is input into process 300:

[83] (1) All entry groups 530 of each of the classification(s), output by sub-process 440, are the same;

[84] (2) The arity of the lexical entries 550, output by sub-process 440, is the same as the number of entry sets 540 associated with the single entry group 530;

[85] (3) There is a one-to-one correspondence between the lexical entries 550, output by sub-process 440, and the entry sets 540 associated with the single entry group 530, with each of the entry sets 540 satisfied by a different one of these lexical entries 550;

[86] (4) Each of the lexical entries 550, output by sub-process 440, is associated with exactly one lexical-entry match, input by sub-process 430, such that an ordinal can be determined that represents the position of each lexical entry 550 within the sequence of lexical-entry matches; and/or

[87] (5) The size of the window, formed by the minimum and maximum of the ordinals, is not greater than the size of the match window associated with the single entry group 530.

[88] One effect of these constraints is to identify every semantic match in a classifier that can be satisfied by a lexical-entry match.

[89] 2.1.1.1. Context Analysis

[90] Embodiments of sub-process 410 for context analysis will now be described in further detail. The purpose of context analysis is to determine which textual components, within a statement from a data-access language, found in network traffic, should be considered. Specifically, the context analysis determines which groups of words (i.e., word sequences) should be analyzed as a whole in sub-processes 420-440.

[91] In an embodiment, sub-process 410 analyzes the semantic model output by the processes described in the prior patents. The semantic model may comprise elements of statements, found in network traffic (e.g., in a data access language), arranged as semantic objects in parse trees within their contexts. When forming the groups, sub-process 410 may consider one or more, including potentially all, of the following context indications, if present in or otherwise known for the given statement:

[92] • server: the server being accessed by the statement (e.g., as defined by an optional first part of a four-part relation identifier);

[93] • database: the database being accessed by the statement (e.g., as defined by an optional second part of the four-part relation identifier);

[94] • schema: the schema being used by the statement (e.g., as defined by an optional third part of the four-part relation identifier);

[95] • relation: the type of relationship (e.g., as defined by a required fourth part of the four-part relation identifier);

[96] • user: the name of a database user identified in the statement.

[97] • role: the name of a database role identified in the statement.

[98] • alias: an identifier optionally assigned as an alias (e.g., of the result column of a query) in the statement.

[99] • column: a reference to a column of a relation.

[100] • comment: the full text of an inline or multi-line natural-language comment.

[101] Within such context, the textual components of a statement in a data-access language are grouped by sub-process 410 for further analysis by subsequent sub-processes (e.g., 420-440). In an embodiment, sub-process 410 may comprise a plurality of discrete passes on one or more parse trees of semantic objects in the semantic model. Each pass may comprise a semantic analysis of one or more textual components within a window that moves or changes between passes. The textual components will generally comprise individual identifiers (e.g., user-defined identifiers of tables, columns, etc., within statements) or natural-language word sequences (e.g., comments found with statements).

[102] 2.1.1.2. Non-Standard Word Analysis

[103] Embodiments of sub-process 420 for non-standard word analysis will now be described in further detail. Word sequences, in statements formed according to whichever data-access language is used, are frequently encoded in a manner that renders simple matching to standard word dictionaries ineffective. Examples of such issues include compounds words (e.g., identifiers of columns, databases, etc.), syntax driven by the rules of the particular data-access language that was used, abbreviations (e.g., employed within identifiers and/or natural-language comments), and/or the like. In short, human programmers often employ non-standard words (i.e., words not found in standard dictionaries) for reasons including syntax rules of the chosen data-access language, brevity, and/or the like. Thus, one goal of non-standard word analysis is to expand these non-standard words to standard words that can be matched to standard dictionaries.

[104] The state of the art in non-standard word analysis is represented by "A Text Normalisation System for Non-Standard English Words," by Flint et al., Dept. of Theoretical and Applied Linguistics, Computer Laboratory, Univ. of Cambridge, Cambridge, U.K. ("Flint et al."), which is hereby incorporated herein by reference as if set forth in full. In an embodiment, the non-standard word analysis in Flint et al. is modified to tailor the analysis to the particular data-classification requirements unique to data-access languages. For instance, the non-standard word analysis may be tailored in one or more, including potentially all, of the following manners:

[105] • Context and/or scope. Non-standard word analysis is highly dependent upon the groups that are input. In an embodiment, the groups are defined by context analysis in sub-process 410. This affects various heuristics employed by the non-standard word analysis in sub-process 420, and materially changes the lemma sequences output by sub-process 420.

[106] • Semantic dictionary. In an embodiment, semantic dictionary 450 is used in place of standard word dictionaries. Thus, lemmas are derived by sub-process 420 from semantic dictionary 450, instead of a standard word dictionary, thereby altering the results of the non-standard word analysis.

[107] • Multi-Language Support. Non-standard word analysis is generally language-dependent. In other words, the process of decoding text sequences to standard words involves language-specific rules. However, in an embodiment, sub-process 420 receives the human language as a parameter and applies rules, specific to the human language specified by that parameter, to produce the lemmas. In this way, sub-process 420 recognizes textual references to non-English words that correspond to the meaning of the lemmas in semantic dictionary 450.

[108] 2.1.1.3. Lexical Entry Matching

[109] Embodiments of sub-process 440 for lexical entry matching will now be described in further detail. In particular, after the non-standard word analysis of sub-process 420 has produced a set of lemmas, these lemmas are matched to lexical entries in semantic dictionary 450, in order to efficiently represent the relationship between statements found in network traffic.

[110] In an embodiment, the context of lexical-entry matches may be retained via representation in a database (e.g., in analytic database 330, which may itself be a relational

database). For instance, the lexical-entry matches, output by sub-process 430, may be represented using two tables (e.g., in PostgreSQL):

Table 1: Relation: stmt_lex_entry		
Column	Type	Meaning
id	Integer	primary key
sql_id	Integer	foreign key to statement in data-access language
context	entry_context	context of reference
ordinal	Integer	zero-based ordinal of context of mappings
src_ident	String	text (e.g., of identifier or comment)
comment_offset	Integer	offset of comment within statement text
mappings	Array of ident_mapping	lemma match(es) within this context
certainty	Real	representation of match quality (0.0 to 1.0)
earliest	Integer	earliest time of statement
language	language_t	source language of matched text

Table 2: Type: ident_mapping		
Column	Type	Meaning
offset	Integer	token offset within context
lemmas	Array of String	matched lemma(s)
wn_ids	Integer	references to lemma(s) in semantic dictionary

[111] The above tables are a very efficient representation, in terms of both time and memory, of all matched lemmas, within individual statements and their contexts. The stmt_lex_entry table represents the result of lexical-entry matching in sub-process 430, and each row in the stmt_lex_entry table represents the result of a lexical analysis on a group output by sub-process 410. The tables may also encode a direct reference to the matched lexical entry in semantic dictionary 450 (e.g., via the wn_ids column in the ident_mapping table). The lexical-entry matches represent all matches, within semantic dictionary 450, to a statement, independent of the details of the user-extensible classifiers discussed elsewhere herein. Advantageously, the lexical-entry matches can be very quickly and efficiently analyzed for many purposes within the semantic-entry matching of sub-process 440, without recourse to the statement text or lemmas.

[112] In an embodiment, each lexical-entry match records the human source language of the matched text, for example, via the language column in the stmt_lex_entry table. The

value of this language column may be set to a parameterized language value from the non-standard word analysis in sub-process 420 and/or semantic dictionary 450. The corresponding ident_mapping(s) in the mappings column of the stmt_lex_entry table may include references to both language-dependent and language-independent identifiers in semantic dictionary 450.

[113] 2.1.1.4. Classifier Definition

[114] In an embodiment, one objective of the data classifier is to allow for the separate representation of completely extensible classification taxonomies, which can be easily defined and extended as needed. There are two design features which can help achieve this goal: (1) semantic dictionary 450; and (2) the classifier definition.

[115] In an embodiment, semantic dictionary 450 comprises at least a portion (e.g., a large subset of English-language words) of Princeton University's WordNet™, as represented in the Resource Definition Framework (RDF). Then, the classifier definitions may reference not just words, but the words' meanings within a large universe of terms, leveraging many years of linguistic research.

[116] In addition, in such an embodiment, foreign-language representations can be mapped to their corresponding lexical entries in semantic dictionary 450, thereby facilitating multi-language support. For example, the WordNet™ RDF ontology includes the translation of senses. Furthermore, many people have contributed to creating translations linked to WordNet™. By leveraging the ability to map English senses to non-English translations of senses in this manner, semantic dictionary 450 enables different source languages to be matched to the same classifications in sub-process 440. For instance, the "ident_mapping" entries, in the lexical-entry matches output by sub-process 430, may reference language-independent WordNet™ identifiers to be used in the semantic-entry matching of sub-process 440.

[117] In an embodiment, classifiers 510 in classifier database 460 may define arbitrary classification taxonomies in terms of sets of lexical entries in WordNet™, defined via the Web Ontology Language (OWL) of the semantic web. This allows for a completely extensible, generally language-independent classification mechanism, defined transparently and purely declaratively via a simple declarative specification. Advantages to such a definition may include one or more of the following:

[118] • The classification taxonomies are completely arbitrary trees, which define semantic categories 520 ultimately in terms of arbitrary references to lexical entries in semantic dictionary 450.

[119] • Multi-language mappings are implied by the same mapping.

[120] • Various time-saving mechanisms are provided to allow for implicit references to lexical entries via WordNet™ synsets, which define particular meanings (e.g., complete with glosses and examples) via implication over a plurality of lexical entries. These mechanisms include, without limitation, discrete lexical entries, complete synsets, hyponym (more specific) synsets that are transitively closed, meronym (part-whole hierarchies) synsets that are transitively closed, and/or arbitrary SPARQL Protocol and RDF Query Language (SPARQL) queries.

[121] • An entry group 530 can specify an optional match window with a set of entry sets 540, thereby requiring that each entry set 540 – perhaps providing only weak evidence for category membership – be separately satisfied within this match window within a particular match context. In practice, this makes it easier to express category membership with fairly general phrases, without resort to sequential notions or complex syntax.

[122] • Allows for user-defined weighting to indicate the strength of implication of particular elements (e.g., children) within a category 520, entry group 530, and/or entry set 540.

[123] In an embodiment, since RDF/OWL is completely declarative, the implementation may be defined directly as follows. Notably, functional hyperlinks have been removed from the following definitions (by removing the "http://" prefix) to comply with the Manual of Patent Examining Procedure (MPEP) for the U.S. Patent and Trademark Office. The following definitions set constants to be used throughout the definition, according to an embodiment:

```
@prefix rdf: <www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix : <dbnetworks.com/rdf/classifier#> .
@prefix rdfs: <www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <www.w3.org/2002/07/owl#> .
@prefix xsd: <www.w3.org/2001/XMLSchema#> .
@prefix lemon: <lemon-model.net/lemon#> .
@prefix wno: <wordnet-rdf.princeton.edu/ontology#>
```

[124] The following defines classifiers 510, as well as name and description properties to be used, according to an embodiment:

```
<dbnetworks.com/rdf/classifier>
a owl:Ontology;
rdfs:comment "Ontology describing class taxonomies & relation to Wordnet";

:classifier
a rdfs:Class, owl:Class;
rdfs:comment "Represents a complete classifier"@en.

:name
a rdf:Property, owl:DatatypeProperty, owl:FunctionalProperty;
rdfs:comment "Name of element";
rdfs:domain [
   a rdfs:Class;
   owl:unionOf
      (:classifier :category :entryGroupClass :entrySetClass :entryClass
      :attributeClass)
];
rdfs:range xsd:string.

:desc
a rdf:Property, owl:DatatypeProperty, owl:FunctionalProperty;
rdfs:comment "Description of element";
rdfs:domain [
   a rdfs:Class;
   owl:unionOf
      (:classifier :category :entryGroupClass :entrySetClass :entryClass
      :attributeClass)
];
rdfs:range xsd:string.
```

[125] The following defines categories 520, as well as the weight property to be used for categories 520, entry groups 530, or entry sets 540, and the parent property, according to an embodiment:

```
:category
a rdfs:Class, owl:Class;
rdfs:comment "Represents a category within a classification taxonomy"@en;
rdfs:subClassOf [
   a owl:Restriction;
   owl:cardinality "1"^^xsd:nonNegativeInteger;
   owl:onProperty :parent
].

:weight
a rdfs:Property, owl:DatatypeProperty, owl:FunctionalProperty;
rdfs:comment
"""Weight to apply in significance computation for assoc. domain object""";
rdfs:domain [
   a rdfs:Class;
   owl:unionOf (:category :entryGroupClass :entryClass)
];
```

```
rdfs:range xsd:decimal.

:parent
a rdfs:Property, owl:ObjectProperty, owl:FunctionalProperty;
rdfs:comment "The classifier or category a category is part of"@eng;
rdfs:domain :category;
rdfs:range [
   a rdfs:Class;
   owl:unionOf (:classifier :category)
].
```

[126] The following defines entry groups 530, according to an embodiment:

```
:entryGroupClass
a rdfs:Class, owl:Class;
rdfs:comment "Represents a group of conditional entry sets"@eng;
rdfs:subClassOf [
   a owl:Restriction;
   owl:minQualifiedCardinality "1"^^xsd:nonNegativeInteger;
   owl:onProperty :entrySet
], [
   a owl:Restriction;
   owl:cardinality "1"^^xsd:nonNegativeInteger;
   owl:onProperty :window
].
```

[127] The following defines entry sets 540 and lexical entries 550, according to an embodiment:

```
:entrySet
a rdfs:Property, owl:ObjectProperty;
rdfs:comment "Associates a set of entries with a group"@eng;
rdfs:domain :entryGroupClass;
rdfs:range :entrySetClass.
:entrySetClass
a rdfs:Class, owl:Class;
rdfs:comment "Represents a set of conditional entries"@eng.

:entry
a rdfs:Property, owl:ObjectProperty;
rdfs:comment "Specifies a lexical entry"@eng;
rdfs:domain [
   a rdfs:Class;
   owl:unionOf (:category :entrySetClass)
];
rdfs:range :entryClass.

:entryGroup
a rdfs:Property, owl:ObjectProperty;
rdfs:comment "Specifies an entry group class for a category";
rdfs:domain :category;
rdfs:range :entryGroupClass.
:entryClass
a rdfs:Class;
rdfs:subClassOf [
```

```
   a owl:Restriction;
   owl:cardinality "1"^^xsd:nonNegativeInteger;
   owl:onProperty :cEntry
].
:cEntry owl:equivalentClass [
rdf:type owl:Class;
owl:unionOf (:lexicalEntry :synset :synsetHyponyms :synsetMeronyms :query)
].

:baseEntry
a rdfs:Property, owl:DatatypeProperty;
rdfs:comment "Super-property of all concrete entries"@en;
rdfs:domain [
   a rdfs:Class;
   owl:unionOf (:category :entryClass :entrySetClass)
].

:lexicalEntry
a rdfs:Property, owl:DatatypeProperty;
rdfs:subPropertyOf :baseEntry;
rdfs:range lemon:LexicalEntry;
rdfs:comment "Represents single wordnet lexical entry".

:synset
a rdfs:Property, owl:DatatypeProperty;
rdfs:subPropertyOf :baseEntry;
rdfs:range wno:Synset;
rdfs:comment "Closure of direct lexical entries of a synset identified".

:synsetHyponyms
a rdfs:Property, owl:DatatypeProperty;
rdfs:subPropertyOf :baseEntry;
rdfs:range wno:Synset;
rdfs:comment "Closure of lexical entries over (improper) hypnyms".

:synsetMeronyms
a rdfs:Property, owl:DatatypeProperty;
rdfs:subPropertyOf :baseEntry;
rdfs:range wno:Synset;
rdfs:comment "Closure of lexical entries over (improper) merynyms".

:query
a rdfs:Property, owl:DatatypeProperty;
rdfs:subPropertyOf :baseEntry;
rdfs:range xsd:string;
rdfs:comment "Represents the lexical entries returned by the sparql query
represented by t".
```

[128] The following defines the window property that specifies the size of the match window, discussed elsewhere herein, as well as heritable and attribute properties, according to an embodiment:

```
:window
a rdfs:Property, owl:DatatypeProperty, owl:FunctionalProperty;
rdfs:comment "Size of match window";
rdfs:range :entryGroupClass;
rdfs:domain xsd:nonNegativeInteger.
```

```
:attributeClass
a rdfs:Class, owl:Class;
rdfs:comment "Abstract, potentially heritable attribute of a category";
rdfs:subClassOf [
   a owl:Restriction;
   owl:cardinality "1"^^xsd:nonNegativeInteger;
   owl:onProperty :name
], [
   a owl:Restriction;
   owl:cardinality "1"^^xsd:nonNegativeInteger;
   owl:onProperty :heritable
].

:heritable
a rdfs:Property, owl:DatatypeProperty, owl:FunctionalProperty;
rdfs:comment "Boolean indicating heritability of associated property";
rdfs:domain :attributeClass;
rdfs:range xsd:boolean.

:attribute
a rdfs:Property, owl:ObjectProperty;
rdfs:comment "Assigns an attribute to a category";
rdfs:domain [
   a rdfs:Class;
   owl:unionOf (:category :entryGroupClass :entrySetClass :entryClass)
];
rdfs:range :attributeClass.
```

[129] For purposes of illustration, one example of a partial classifier 510 for the General Data Protection Regulation (GDPR) for the European Union (EU) is defined below. For purposes of practicality, only a single category 520, and two entry groups 530 are illustrated. In addition, functional hyperlinks have been removed (by removing the “http://” prefix) to comply with the MPEP for the U.S. Patent and Trademark Office.

```
:gdpr
  a cls:classifier;
  cls:name "GDPR classifier";
  cls:desc "Default data-classifier for the GDPR".

:identifying
  a cls:category;
  cls:parent :gdpr;
  cls:desc "Information which might identify an individual";
  cls:entryGroup [
    cls:desc "Name variants";
    cls:window 2;
    cls:entrySet [
      cls:lexicalEntry <wordnet-rdf.princeton.edu/lemma/family>;
      cls:lexicalEntry <wordnet-rdf.princeton.edu/lemma/first>;
      cls:lexicalEntry <wordnet-rdf.princeton.edu/lemma/full>;
      cls:lexicalEntry <wordnet-rdf.princeton.edu/lemma/last>;
      cls:lexicalEntry <wordnet-rdf.princeton.edu/lemma/middle>;
      cls:lexicalEntry <wordnet-rdf.princeton.edu/lemma/screen>;
      cls:lexicalEntry <wordnet-rdf.princeton.edu/lemma/user>;
```

```
    ];
    cls:entrySet [
      cls:lexicalEntry <wordnet-rdf.princeton.edu/lemma/name>;
    ]
  ];
  cls:entryGroup [
    cls:desc "tax/economy/personal & id & number variants";
    cls:window 3;
    cls:entrySet [
      cls:lexicalEntry <wordnet-rdf.princeton.edu/wn31/economy-n>;
      cls:lexicalEntry <wordnet-rdf.princeton.edu/lemma/personal==>;
      cls:lexicalEntry <wordnet-rdf.princeton.edu/lemma/tax>;
      cls:lexicalEntry <wordnet-rdf.princeton.edu/lemma/taxpayer==>;
    ];
    cls:entrySet [
      cls:lexicalEntry <wordnet-rdf.princeton.edu/lemma/identity==>;
      cls:lexicalEntry <wordnet-rdf.princeton.edu/lemma/identification==>;
      cls:lexicalEntry <wordnet-rdf.princeton.edu/lemma/id>;
    ];
    cls:entrySet [
      cls:lexicalEntry <wordnet-rdf.princeton.edu/lemma/number>;
    ]
  ];
```

[130] 2.1.1.5. Semantic Entry Matching

[131] Embodiments of sub-process 440 for semantic entry matching will now be described in further detail. Given the lexical-entry matches and classifier definitions described above, classifier matching in sub-process 440 can be done at high speed by incrementally determining all matches to categories 520, constrained by window requirements, requirements for multiple entry sets 540 to be collectively satisfied within an entry group 530 for a match to the entry group 530 (when applicable), and/or the like.

[132] In an embodiment, each row in the stmt_lex_entry table, described above, is analyzed by sub-process 440. Each row represents a lexical-entry match, and may give rise, in sub-process 440, to zero, one, or a plurality of classifications. With respect to statements, each classification is evidence that the invocation of the statement (e.g., an SQL statement) accesses data within a category 520 of a classifier 510.

[133] In an embodiment, classifications, output by sub-process 400, may be represented as follows for fast and efficient scoring and reporting in sub-processes 340 and/or 350:

Table 3: Relation: stmt_entry_match		
Column	Type	Meaning
sle_id	Integer	reference to row in stmt_lex_entry table
entry_sets	stmt_entry_set_match_t[]	list of classification match(es)

Table 4: Type: stmt_entry_set_match_t		
Column	Type	Meaning
offset	Integer	offset within mappings column of stmt_lex_entry row
ce_id	Integer	reference to classifier that was matched

[134] Notable concepts reflected in this representation of the classifications include, without limitation:

[135] • each row in the stmt_entry_match table represents the satisfaction of a discrete category 520 defined by a classifier 510;

[136] • classifiers 510 are expanded from RDF/OWL definitions into a concrete (e.g., PostgreSQL) schema referenced by the matches (e.g., via the ce_id column of the stmt_entry_set_match_t table);

[137] • the specific lexical entries satisfying a single-entry-set or multi-entry-set entry group 530 are also directly represented; and/or

[138] • this representation allows fast and flexible scoring and reporting (e.g., in sub-processes 340 and/or 350) of all classifier implications (e.g., represented as classifications in analytic database 330), even in very large environments with many thousands of databases and clients, without loss of detail.

[139] 2.1.2. Quantitative Scoring

[140] Embodiments of sub-process 340 for quantitative scoring will now be described. Quantitative scoring is appealing in security-oriented environments. The quantitative scoring of sub-process 340 may account for both the selectivity of classifications, which compare local match frequencies to broad contexts, and various weightings (e.g., user-defined weightings). Quantitative scoring appeals primarily to analytic applications or users, which are typically interested in data security. One goal of quantitative scoring is to generate a score that indicates the relative likelihood that statements, within a particular context that defines a subset of all observed statements that have been classified in sub-process 320, implicate a particular category 520 within a taxonomy of one or more classifiers 510. While a particular algorithm for generating a quantitative category score will be described below, it should be understood that other algorithms may be used. In addition, while the steps will be

described below in a certain order, the steps may be executed in a different order than described and/or one or more steps may be executed in parallel with another step.

[141] In an embodiment, the quantitative scoring of sub-process 340 may utilize the following data types and relations (e.g., defined in PostgreSQL):

Table 5: Type: stmt_entry_match_score_t		
Column	Type	Meaning
sle_id	Int	Reference to id column in stmt_lex_entry table
offset_	Int	Offset of match within lexical group
sql_id	Int	Reference to SQL statement
sle_context	Parser.access_t	Access type, 'identifier' or 'comment'
sle_context_aux	Context_aux_t	Auxiliary context
sle_certainty	Float	Certainty value for the row in stmt_lex_entry table
ent_id	Int	Reference to classifier lexical entry
ent_weight	Float	User-defined weighting of classifier lexical entry
wn_id	Int	Reference to lexical entry in semantic dictionary
wn_prob	Float	Probability of lexical entry in semantic dictionary
ces_id	Int	Reference to classifier entry set
g_id	Int	Reference to classifier entry group
window_	Int	Window size of classifier entry group
g_weight	Float	User defined weighting of classifier entry group
cat_id	Int	Reference to classifier category
c_id	Int	Reference to classifier

Table 6: Type: stmt_lex_entry_stats_t		
Column	Type	Meaning
lex_entries	Int	Total number of lexical entries considered
lemmas	Int	Total number of lemmas associated with lexical entries
wn_entries	Int	Total number of lexical entries for lemmas in semantic dictionary
score_vf	Float	A weighting based on lemmas considered

Table 7: Type: score_state_t		
Column	Type	Meaning
stats	sle_stats_t	Summary statistics from stmt_lex_entry_stats_t
cat_id	Int	Reference to current classifier category
grp_id	Int	Reference to current classifier entry group
ces_id	Int	Reference to current classifier entry set
ent_id	Int	Reference to current classifier lexical entry
sql_id	Int	Reference to current SQL statement

context	entry_context_t	Current context of lexical entry
context_aux	entry_context_aux_t	Current auxiliary context of lexical entry
classifiers	classifier_t*	Head of current classifier list

Table 8: Type: match_score_t		
Column	Type	Meaning
stmt_stats	stmt_lex_entry_stats_t	Summary associated with the aggregation
scores	classifier_score_t[]	Scores of the classified results

Table 9: Type: classifier_score_t		
Column	Type	Meaning
c_id	Int	Reference to classifier
score	Float	Floating point score

[142] FIG. 6 illustrates an example algorithm 600 for quantitative scoring, according to an embodiment. For simplicity and generality, algorithm 600 may be defined as a PostgreSQL user-defined aggregate. This scoring aggregate may be applied to any set of one or more rows in the stmt_entry_match table, to produce a scalar score for that set of row(s). In an embodiment, the scoring aggregate scans over records of the stmt_entry_match_score_t and stmt_lex_entry_stats_t data types. The stmt_entry_match_score_t records may be associated in a one-to-one relation with rows in the stmt_entry_match table. Functions of the scoring aggregate may be implemented as three user-defined functions: state initialization (step 605), transition (step 610), and finalization (steps 615). These functions may be programmed in C or any other suitable programming language, and operate on records of the score_state_t data type.

[143] In an embodiment, values in the wn_prob column of the stmt_entry_match_score_t records are computed via a procedure derived from Zipf's Law. Zipf's Law is a heuristic that relates word sequences, sorted by frequency of occurrence, to a probability of occurrence, without undue dependence on a specific application corpus.

[144] Each stmt_lex_entry_stats_t record may be created by a subsidiary function of the scoring aggregate, which sums values across all of the rows in the stmt_lex_entry table that are associated with the given context of interest (e.g., all of the rows in the stmt_lex_entry table that are associated with a particular database service). In an embodiment, the values in the score_vf column of the stmt_lex_entry_stats_t records represent the scale of the given context of interest, compared to the scale of all lexical entries in semantic dictionary 450.

[145] Step 605 represents the initialization function of the scoring aggregate of algorithm 600, in which data structures are initialized. For example, in step 605, algorithm 600 initializes a record of the match_score_t data type and creates a score_state_t record (e.g., implemented as a C-language struct type). For example, all values may be initialized to zero, except for the stmt_stats column, which may be initialized to the record of the stmt_lex_entry_stats_t data type, representing the summary of statistics, which may be used and remain constant across all iterations of the subsequent steps.

[146] In an embodiment, a classifiers linked list refers to records associated with classifiers 510. Each classifier record contains the associated classifier state, derived from a stmt_entry_match_score_t record, which includes a categories linked list of records associated with categories 520 in that classifier 510. In turn, each record in the categories linked list refers to an entry-groups linked list of records associated with entry groups 530 in that category 520. In turn, each record in the entry-groups linked list refers to an entry-sets linked list of records associated with entry sets 540 in that entry group 530. In turn, each record in the entry-sets linked list refers to a lexical-entries linked list of records associated with lexical entries 550 in that entry set 540. In turn, each record in the lexical-entries linked list refers to a matches linked list of records associated with the matches between that lexical entry 550 and each distinct statement (e.g., SQL statement), entry_context_aux_t record, and entry_context_t record. In turn, each record in the matches linked list refers to another match record (e.g., C-language struct type) that represents the next statement, entry_context_aux_t record, or entry_context record. Notably, starting with the classifiers field of the score_state_t record, passed (e.g., by PostgreSQL) to each application of the transition function in step 610, a "current" classifier 510, category 520, entry group 530, entry set 540, lexical entry 550, and match is defined by transitive closure over the linked lists embedded in these records (e.g., C-language struct type records). Each of the records in the various nested linked lists may be implemented as a C-language struct type, and may also comprise a score record, which represents the number of matches accumulated during the aggregation.

[147] Step 610 represents the transition function of the scoring aggregate of algorithm 600, which a classification record is created. For example, in an embodiment of step 610, algorithm 600 sorts all stmt_entry_match_score_t records of interest by their c_id, cat_id, g_id, ces_id, ent_id, sql_id, sle_context_aux, and sle_context columns, in that order, and then applies the transition function to each of these stmt_entry_match_score_t records of interest. In one implementation, the transition function may comprise the following steps to aggregate data in the score_state_t record that was created in step 605:

[148] (1) If the c_id represents the first row seen for the identified classifier 510 (e.g., the first iteration of the transition function, or a newly seen identifier indicating the start of the next classifier 510, etc.), create a new record for this current classifier 510, and add this new classifier record to the classifiers linked list;

[149] (2) Increment the number of matches associated with the current classifier 510;

[150] (3) If the cat_id represents the first row seen for the identified category 520 for the current classifier 510 (e.g., the categories linked list for the current classifier 510 is empty, or the cat_id is a newly seen identifier indicating the start of the next category 520, etc.), create a new record for this current category 520, and add this new category record to the categories linked list for the current classifier record;

[151] (4) Increment the number of matches associated with the current category 520;

[152] (5) If the g_id represents the first row seen for the identified entry group 530 for the current category 520 of the current classifier 510 (e.g., the entry-groups linked list for the current category 520 is empty, or the g_id is a newly seen identifier indicating the start of the next entry group 530, etc.), create a new record for this current entry group 530, and add this new entry-group record to the entry-groups linked list for the current category record;

[153] (6) Increment the number of matches associated with the current entry group 530;

[154] (7) If the ces_id represents the first row seen for the identified entry set 540 for the current entry group 530 of the current category 520 of the current classifier 510 (e.g., the entry-sets linked list for the current entry group 530 is empty, or the ces_id is a newly seen identifier indicating the start of the next entry set 540, etc.), create a new record for this current entry set 540, and add this new entry-set record to the entry-sets linked list for the current entry-group record;

[155] (8) Increment the number of matches associated with the current entry set 540;

[156] (9) If the ent_id represents the first row seen for the identified lexical entry 550 for the current entry set 540 of the current entry group 530 of the current category 520 of the current classifier 510 (e.g., the lexical-entries

linked list for the current entry set 540 is empty, or the ent_id is a newly seen identifier indicating the start of the next lexical entry 550, etc.), create a new record for this current lexical entry 550, and add this new lexical-entry record to the lexical-entries linked list for the current entry-set record;

[157] (10) If the sql_id represents the first row seen for the identified statement (e.g., the statements associated with the current matches linked list for the current lexical entry 550 is empty, or the sql_id is a newly seen identifier indicating the start of the next statement, etc.), create a new match record for the current statement, and add this new match record to the matches linked list for the current lexical-entry record;

[158] (11) If the sle_context_aux represents the first row seen for the current sle_context_aux value (e.g., the sle_context_aux value represents the next auxiliary context in the aggregation sequence), create a new match record for this sle_context_aux value, and add this new match record to the matches linked list for the current lexical-entry record.

[159] (12) If the sle_context represents the first row seen for the current sle_context value (e.g., the sle_context value represents the next context in the aggregation sequence), create a new match record for this sle_context value, and add this new match record to the matches linked list for the current lexical-entry record.

[160] (13) Increment the number of matches associated with the current match record.

[161] Step 615 represents the finalization function of the scoring aggregate of algorithm 600. In an embodiment of step 615, algorithm 600 converts the score_state_t record, which has accumulated information over the aggregation of all of the stmt_entry_match_score_t records of interest, into a final score represented by a classifiers_score_t record. Specifically, the finalization function of algorithm 600 performs a depth-first walk of the tree (e.g., created in step 610), rooted in the classifiers linked list, and descends into the categories linked list, entry-groups linked list, entry-sets linked list, and lexical-entries linked list, in turn, to produce a classifiers_score_t record for each classifier 510. For example, step 615 may comprise calculating a local score for each matched lexical entry 550, calculating an entry-set score for each matched entry set 540 based on the local scores of the matched lexical entries 550 contained within the matched entry set 540, calculating an entry-group score for each

matched entry group 530 based on the entry-set scores of the matched entry sets 540 contained within the matched entry group 530, calculating a category score for each matched category 520 based on the entry-group scores of the matched entry groups 530 contained within the matched category 520, and calculating a classifier score for each matched classifier 510 based on the category scores of the matched categories 520 contained with the matched classifier 510.

[162] In an embodiment, the local score for each matched lexical entry 550 is computed by first determining the number of lexical entries 550 expected from a randomly distributed population of references to the lexical entry, in semantic dictionary 450, that is associated with the classifier lexical entry 550. This number may be computed by multiplying the Zipf's Law estimated probability of the lexical entry in semantic dictionary 450 by the total number of lexical entries (e.g., wn_entries) from the associated stmt_lex_entry_stats_t record. Then, the actual number of occurrences is determined by multiplying the number of matched lexical entries 550 (i.e., associated with the lexical entry in semantic dictionary 450) by the certainty associated with the matched lexical entry. The certainty represents the strength of a non-standard word match and, by default, may be set to 1.0 (e.g., if no specific certainty is available or specified). Next, an intermediate score for a given matched lexical entry 550 is calculated by subtracting the expected number of matches from the actual number of matches, and dividing the difference by the expected number of matches. A raw score can then be calculated by multiplying this intermediate score by a user-defined weighting for the matched lexical entry 550, to produce the local score for the matched lexical entry 550.

[163] In an embodiment, the entry-set score for each matched entry set 540 is calculated by multiplying the maximum local score for the matched lexical entries 550 contained by the given entry set 540 by the weighting value in the score_vf column of the stmt_lex_entry_stats_t record.

[164] In an embodiment, the entry-group score for each matched entry group 530 is computed based on the entry-set scores of the entry sets 540 contained by the given entry group 530. If the given entry group 530 contains a single entry set 540, the entry-group score for that given entry group 530 is calculated by multiplying the entry-set score for the single entry set 540 by a weight (e.g., a user-defined weight for the entry group 530). On the other hand, if the given entry group 530 contains a plurality of entry sets 540 (i.e., the given entry group 530 is satisfied via the satisfaction of a plurality of entry sets 540), multiple lexical entries in semantic dictionary 450 have been satisfied, thereby implying additionally selectivity for which algorithm 600 should preferably account.

[165] Thus, in an embodiment, in the event that the given entry group 530 contains a plurality of entry sets 540, the entry-group score is computed by first computing a probability estimate that is accumulated by a series of computations that are based on the maximum local scores of the lexical entries 550 that are associated with each of the entry sets 540 in the given entry group 530. This accumulated probability estimate may be initiated by, for the first entry set 540 in the given entry group 530, computing the probability of the lexical entry in semantic dictionary 450 that is associated with the matched lexical entry 550 having the maximum local score from all matched lexical entries 550 contained within the first entry set 540. This probability may be calculated by, first, subtracting the probability associated with that lexical entry in semantic dictionary 450 from 1.0, and then raising the difference by the power given by the total number of lexical entries in semantic dictionary 450 (e.g., as represented by the value in the wn_entries column in the stmt_lex_entry_stats_t record), to produce an intermediate quantity. A probability may then be calculated by subtracting the intermediate quantity from 1.0, and dividing the difference by the weight (e.g., user-defined weight for the lexical entry 550) associated with the matched lexical entry 550 having the maximum local score.

[166] For each subsequent entry set 540, contained within the given entry group 530, a similar computation is performed. However, for each subsequent entry set 540, the intermediate quantity is calculated by subtracting the probability associated with the lexical entry in semantic dictionary 450, corresponding to matched lexical entry 500 within the given entry set 540 having the maximum local score, from 1.0, and then raising the difference by the power given by the window size of the given entry group 530. The probability is calculated in the same manner as for the first entry set 540 by subtracting the intermediate quantity from 1.0, and dividing the difference by the weight (e.g., user-defined weight) associated with the matched lexical entry 550 having the maximum local score.

[167] The probability for the given entry group 530 is accumulated by multiplying the probability calculated for each subsequent entry set 540 by the accumulated probability calculated in the previous step (i.e., after accounting for the preceding entry set 540). After the accumulated probability has taken into account every entry set 540, contained within the given entry group 530, the entry-group score for the given entry group 530 may then be calculated by dividing 2.0 by this final accumulated probability, and then multiplying the quotient by the weight associated with the given entry group 530 (e.g., a user defined weight represented by the value in the score_vf column of the associated stmt_lex_entry_stats_t record).

[168] In an embodiment, the category score for each matched category 520 is computed from the entry-group scores of the entry groups 530 contained within that category 520. For example, the category score for a given category 520 may be calculated by multiplying the maximum entry-group score from all of the entry-group scores of the entry groups 530, contained within the given category 520, by a weight (e.g., a user-defined weight) associated with the given category 520.

[169] In an embodiment, the classifier score for each matched classifier 510 is computed from the category scores of the categories 520 contained with that classifier 510. For example, the classifier score for a given classifier 510 may be the maximum category score from all of the category scores of the categories 520 contained within the given classifier 510. Empirically, the ranges of classifier scores can be stratified as follows:

Table 10: Ranges of Raw Scores	
Range	Significance
< 0.0	None
0.0 – 1.0	Negligible
1.0 – 3.0	Modest
> 3.0	Powerful (and increasingly powerful as score rises)

[170] In an embodiment, sub-process 340 may also comprise quantitative filtering in step 620. However, in an alternative embodiment, step 620 may be omitted. For example, when depicting the categories 520 of classifications implicated by a given context (e.g., in response to a user or system query), sub-process 340 may execute a meta-classification algorithm in step 620 that heuristically identifies the "knee" in the curve of the category scores for the given context. This heuristic essentially estimates the slope of the graph of category scores, and filters out category scores beyond the point where this slopes reaches some minimum (e.g., thereby representing insignificant category scores), subject to one or more parameters that prevent the algorithm from ignoring significant category scores that fall above a first threshold value and/or that enable the algorithm to ignore insignificant category scores that fall below a second threshold value. The algorithm may employ a coordinate transform that is intended to reflect the scale of all data as square. Advantageously, the qualitative filtering of step 620 minimizes false positives, thereby making it easier for operators to pinpoint exactly what and where their classified data are, across very large enterprises.

[171] 2.1.3. Qualitative Filtering

[172] Embodiments of sub-process 350 for qualitative filtering will now be described. Qualitative filtering enables explicit filtering of the least useful category matches, and may automatically find optimal match clusters which identify the weakest matches implicating the largest number of contexts (e.g., typically discrete database services). Qualitative filtering allows operators to explicitly reduce generic classifiers to their local requirements and still minimize false positives. Qualitative filtering appeals primarily to applications or operators that wish to see "all of the data," regardless of the quantitative significance. Such appeal is common in compliance-oriented environments, in which all potential risks need to be explicitly considered.

[173] In an embodiment, the approach to implementing sub-process 350 is to selectively negate specific classifier matches implied by observed statements, to form an abbreviated classifier with dramatically reduced noise. Many matches may be implied by a classifier definition, and there may be enormous variance in the data found in any particular environment. Thus, sub-process 350 may comprise an algorithm that makes recommendations for highly tailored negations, designed to reduce the largest number (i.e., benefit) of implicated contexts (e.g., typically discrete database services) with the lowest classifier category group significance (i.e., risk). The algorithm may exhaustively discover discrete sets of category satisfactions at specific levels of significance, and present these sets in order of increasing risk. A user may specify one or more elements (e.g., via a graphical user interface), which precisely define a modified or abbreviated classifier. The semantic entry matching in sub-process 440 may be extended to elide matches negated by the abbreviated classifier. The net effect of sub-process 350 is to allow the depiction of all classifications found in any context, while using local and explicit decision-making to suppress noise.

[174] FIG. 7 illustrates an example algorithm 700 for qualitative filtering, according to an embodiment. Algorithm 700 may be implemented as a single PostgreSQL stored procedure).

[175] In step 705, algorithm 700 computes and stores a first relation between each entry in the stmt_lex_entry table and the associated context (e.g., database service identifier).

[176] In step 710, algorithm 700 projects a second relation between each context (e.g., database service identifier) and category scores (e.g., the category scores produced by algorithm 600) for all associated classifications. This projection may be staged through a number of intermediate relations.

[177] In step 715, algorithm 700 projects a third relation giving the maximum category score by distinct entry sets 540 that satisfy one or more classifications. This relation may be given a primary key.

[178] In step 720, algorithm 700 projects a fourth relation between each context (e.g., database service identifier) and the primary key of the third relation projected in step 715.

[179] In step 725, algorithm 700 perform a loop for as long as the number of distinct contexts (e.g., database service identifiers) within the fourth relation from step 720 is greater than zero. Specifically, if the number of distinct contexts within the fourth relation is greater than zero (i.e., "Yes" in step 725), algorithm 700 proceeds to step 730. Otherwise, if the number of distinct contexts within the fourth relation is zero (i.e., "No" in step 725), algorithm 700 ends.

[180] In step 730, algorithm 700 determines the minimum category score in the fourth relation and determines a context (e.g., database service identifier) associated with this minimum score. It should be understood that the determined context may be one of a plurality of contexts associated with the minimum score.

[181] In step 735, algorithm 700 computes the set of entry sets 540 required to negate all of the classifications associated with this context (e.g., database service identifier).

[182] In step 740, algorithm 700 deletes all of the rows from the fourth relation that match any of the entry sets 540 within the set computed in step 735. This deletion of rows represents the effect of negating all of the contexts (e.g., services) implicated by just these entry sets 540.

[183] In step 745, algorithm 700 computes the new number of distinct contexts (e.g., database service identifiers) in the fourth relation following the deletion in step 740.

[184] In step 750, algorithm 700 returns a row comprising the set of entry sets 540 deleted in step 740, the maximum quantitative score associated with the proposed negation (e.g., representing the associated risk), and the number of released contexts (e.g., database service identifiers), which represents the delta between the number of distinct contexts computed in step 745 and the entry criteria to the loop represented by step 725. This result row serves several useful purposes. First it recommends a specific set of classifier entry sets that, if negated, would reduce the number of implicated contexts (e.g., service identifiers), with a strictly maximum score/minimum risk. Second, it characterizes this level of risk. Third, it gives the number of contexts (e.g., service identifiers) that will no longer be implicated if the given set of entry sets 540 is negated. In an embodiment, if the end-user

selects such a row, sub-process 350 will effectively negate the associated set of entry sets 540, thereby pruning the associated classifier 510.

[185] Following step 750, algorithm 700 continues the loop, represented by step 725, with this new number of distinct contexts, unless this new number of distinct contexts is zero. Returning multiple "frontier" rows, via a plurality of iterations of step 750 within these loops, allows the end-user to make multiple, inherently conditional decisions at one time.

[186] It should be understood that a user may view the recommendations of classifier entry sets 540 (e.g., via a graphical user interface), produced by iterations of steps 725-750 of algorithm 700 (e.g., in increasing level of significance and decreasing level of triviality). The user may select one or more of the recommendations to thereby abbreviate the classifier 510 by negating the recommended entry sets 540. Thereafter, classifications which rely on the negated entry sets 540 may be filtered out from classifications shown to the user, so that the user can concentrate on only those classifications which are important to the user.

[187] 2.2. Applications

[188] As discussed throughout the present description, process 300 automates data discovery and monitoring in a manner that was not previously possible. Specifically, the semantic analysis of sub-process 310 automatically captures and produces a semantic analysis of statements and comments in network traffic between network agents 130. Then, the data classification of sub-process 320 classifies the captured statements and comments in their contexts. In essence, sub-processes 310 and 320 infer what and where data is stored based on the data-access-language statements used to access that data. The result is a representation, stored in analytic database 330, of the type of data (i.e., classifications of data) being stored in the network and the locations (i.e., contexts) of each type of data. Because this representation of types and locations of data is produced automatically, entities no longer have to go through the costly and time-consuming process of manually and explicitly identifying, accessing, and probing their structured data.

[189] Furthermore, as discussed herein, the representations of types and locations of data in the network, stored in analytic database 330, can be quickly scored and filtered via sub-processes 340 and 350. This enables the operator of the network to quickly identify relevant types of data and their locations. Notably, because English and non-English language statements and comments are mapped to the same lexical entries 550 in semantic dictionary 450, as described elsewhere herein, classifications can be reported to end user's in

any chosen language (e.g., by leveraging the translations of lexical entries present in semantic dictionary 450).

[190] Accordingly, operators can more efficiently manage their data to mitigate cyberattacks and other data leaks (e.g., by securing or changing the locations where sensitive data is stored), better handle the aftermath of cyberattacks or other data leaks (e.g., by identifying what data was stored in the affected locations), more easily comply with regulations (e.g., GDPR) governing the handling of sensitive data (e.g., by identifying where such data is stored to determine how that data is secured), and/or the like. Other advantages of automated discovery of the types and locations of data, stored within an entity's network, will be readily apparent to those skilled in the art.

[191] The above description of the disclosed embodiments is provided to enable any person skilled in the art to make or use the invention. Various modifications to these embodiments will be readily apparent to those skilled in the art, and the general principles described herein can be applied to other embodiments without departing from the spirit or scope of the invention. Thus, it is to be understood that the description and drawings presented herein represent a presently preferred embodiment of the invention and are therefore representative of the subject matter which is broadly contemplated by the present invention. It is further understood that the scope of the present invention fully encompasses other embodiments that may become obvious to those skilled in the art and that the scope of the present invention is accordingly not limited.

[192] Combinations, described herein, such as "at least one of A, B, or C," "one or more of A, B, or C," "at least one of A, B, and C," "one or more of A, B, and C," and "A, B, C, or any combination thereof" include any combination of A, B, and/or C, and may include multiples of A, multiples of B, or multiples of C. Specifically, combinations such as "at least one of A, B, or C," "one or more of A, B, or C," "at least one of A, B, and C," "one or more of A, B, and C," and "A, B, C, or any combination thereof" may be A only, B only, C only, A and B, A and C, B and C, or A and B and C, and any such combination may contain one or more members of its constituents A, B, and/or C. For example, a combination of A and B may comprise one A and multiple B's, multiple A's and one B, or multiple A's and multiple B's.

CLAIMS

What is claimed is:

1. A method comprising using at least one hardware processor to:

receive a plurality of semantic objects, representing textual components of at least one or more statements, in a data-access language, captured from network traffic, within a context;

parse the textual components into one or more groups of one or more words;

matching the one or more the groups to lemmas in a semantic dictionary to produce one or more matched lemmas representing the one or more groups;

matching the one or more matched lemmas to lexical entries in the semantic dictionary to produce one or more matched lexical entries;

matching the one or more matched lexical entries to one or more classifiers in a classifier database to produce one or more classifications, wherein each of the one or more classifications represents a possibility that at least one of the one or more statements accesses data within that classification; and

storing the one or more classifications in association with the context in an analytic database.

2. The method of Claim 1, wherein the plurality of semantic objects represent textual components of one or more statements in the data-access language and one or more comments in the data-access language.

3 The method of Claim 1, wherein the textual components comprise one or more identifiers utilized by one or more statements in the data-access language.

4. The method of Claim 3, wherein parsing the one or more identifiers into the one or more groups of one or more words comprises expanding one non-standard identifier into two or more standard words.

5. The method of Claim 1, wherein the textual components are parsed into the one or more groups of one or more words according to the context.

6. The method of Claim 1, wherein the context comprises a database service.

7. The method of Claim 1, wherein parsing the textual components into one or more groups of one or more words comprises, for one or more passes, semantically analyzing textual components within a window that moves or changes between passes.

8. The method of Claim 1, wherein matching the one or more matched lemmas to lexical entries in the semantic dictionary comprises constraining the matching according to one or more of the following constraints:

each of the one or more matched lemmas is matched, at most, once to the lexical entries in the semantic dictionary;

a total number of characters of the one or more lemmas, matched to the lexical entries in the semantic dictionary, is maximized; and

any of the one or more matched lemmas that are not matched to the lexical entries in the semantic dictionary are skipped.

9. The method of Claim 8, wherein the total number of characters of the one or more lemmas is determined based on a written representation associated with each of the one or more matched lexical entries in the semantic dictionary, wherein each written representation is in a single human language regardless of a human language of the groups which were matched to the one or more lemmas.

10. The method of Claim 1, wherein each of the one or more classifiers comprises a containment hierarchy that comprises one or more categories contained within the classifier, one or more entry groups contained within each of the one or more categories, one or more entry sets contained within each of the one or more entry groups, and one or more lexical entries contained within each of the one or more entry sets.

11. The method of Claim 10, wherein any of the one or more entry groups that contains a plurality of lexical entries is associated with a match window, such that the entry group is only matched if the one or more matched lexical entries comprise all of the contained plurality of lexical entries within the match window.

12. The method of Claim 1, further comprising generating a quantitative score for each of the one or more classifications, wherein the quantitative score indicates a likelihood that the one or more statements implicate the respective classification within the context.

13. The method of Claim 12, wherein each of the one or more classifications comprises a containment hierarchy that comprises one or more categories contained within the classification, one or more entry groups contained within each of the one or more categories, one or more entry sets contained within each of the one or more entry groups, and one or more lexical entries contained within each of the one or more entry sets, and wherein generating the quantitative score for each of the one or more classifications comprises:

generating a local score for each of the one or more lexical entries;

generating an entry-set score for each of the one or more entry sets based on the local scores for the one or more lexical entries contained within that entry set;

generating an entry-group score for each of the one or more entry groups based on the entry-set scores for the one or more entry sets contained within that entry group;

generating a category score for each of the one or more categories based on the entry-group scores for the one or more entry groups contained within that category; and

generating a classification score for each of the one or more classifications based on the category scores for the one or more categories contained within that classification.

14. The method of Claim 13, wherein the local score for each of the one or more lexical entries is calculated based on an actual number of times that the lexical entry occurs, an expected number of times that the lexical entry is expected to occur, and a weight.

15. The method of Claim 13, wherein the entry-set score for each of the one or more entry sets is calculated based on a maximum local score for the one or more lexical entries contained with that entry set, and a weight.

16. The method of Claim 13, wherein the entry-group score for each of the one or more entry groups is calculated based on:

if the entry group contains only a single entry set, the entry-set score for the single entry set, and a weight; and,

if the entry group contains a plurality of entry sets, an accumulated probability calculated based on the local scores of the one or more lexical entries contained in each of the plurality of entry sets, and a weight.

17. The method of Claim 13, wherein the category score for each of the one or more categories is calculated based on a maximum entry-group score for the one or more entry groups contained with that category, and a weight.

18. The method of Claim 13, wherein the classification score for each of the one or more classifications is calculated based on a maximum category score for the one or more categories contained with that classification.

19. The method of Claim 13, further comprising filtering out at least one of the one or more categories contained within one or more classifications based on the category score calculated for the at least one category.

20. The method of Claim 1, further comprising filtering out at least one of the one or more classifications by abbreviating at least one of the one or more classifiers to elide matches to the at least one classifier that produce the at least one classification.

21. A system comprising:

at least one hardware processor; and

one or more software modules that, when executed by the at least one hardware processor, perform the method of any one of Claims 1-20.

22. A non-transitory computer-readable medium having instructions stored therein, wherein the instructions, when executed by a processor, cause the processor to perform the method of any one of Claims 1-20.

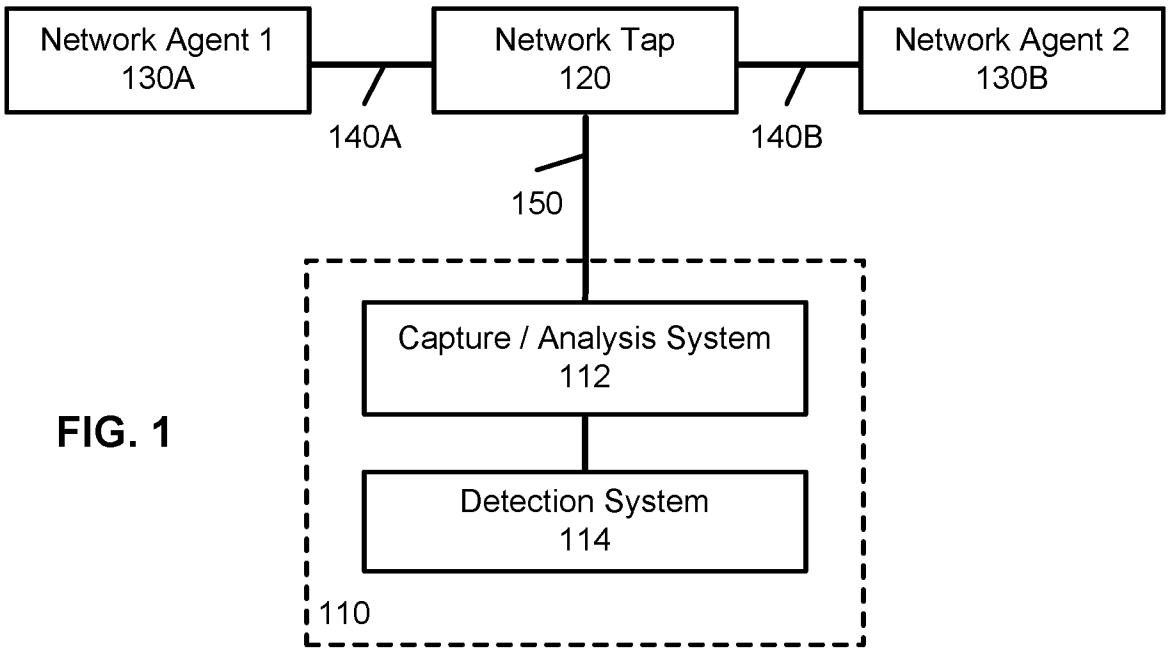

Network Agent 1
130A
Network Tap
120
Network Agent 2
130B
140A
140B
150
Capture / Analysis System
112
Detection System
114
110

FIG. 1

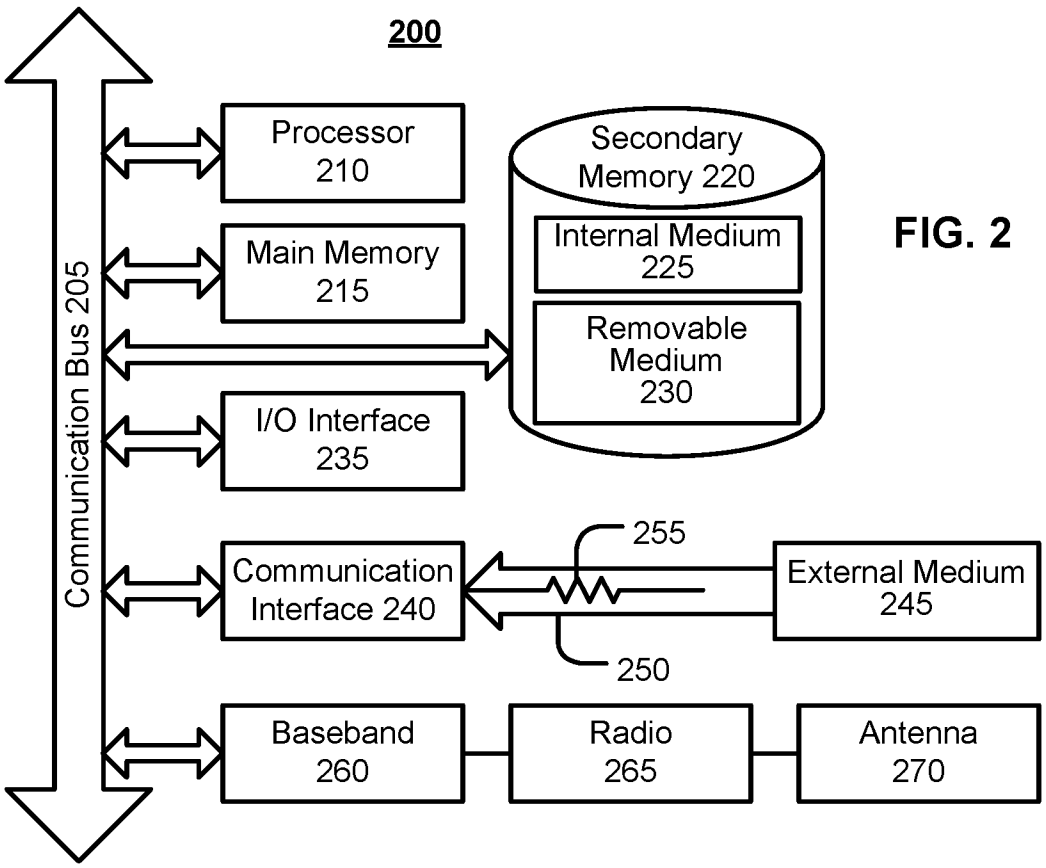

200
Communication Bus 205
Processor
210
Main Memory
215
I/O Interface
235
Secondary
Memory 220
Internal Medium
225
Removable
Medium
230
Communication
Interface 240
255
250
External Medium
245
Baseband
260
Radio
265
Antenna
270

FIG. 2

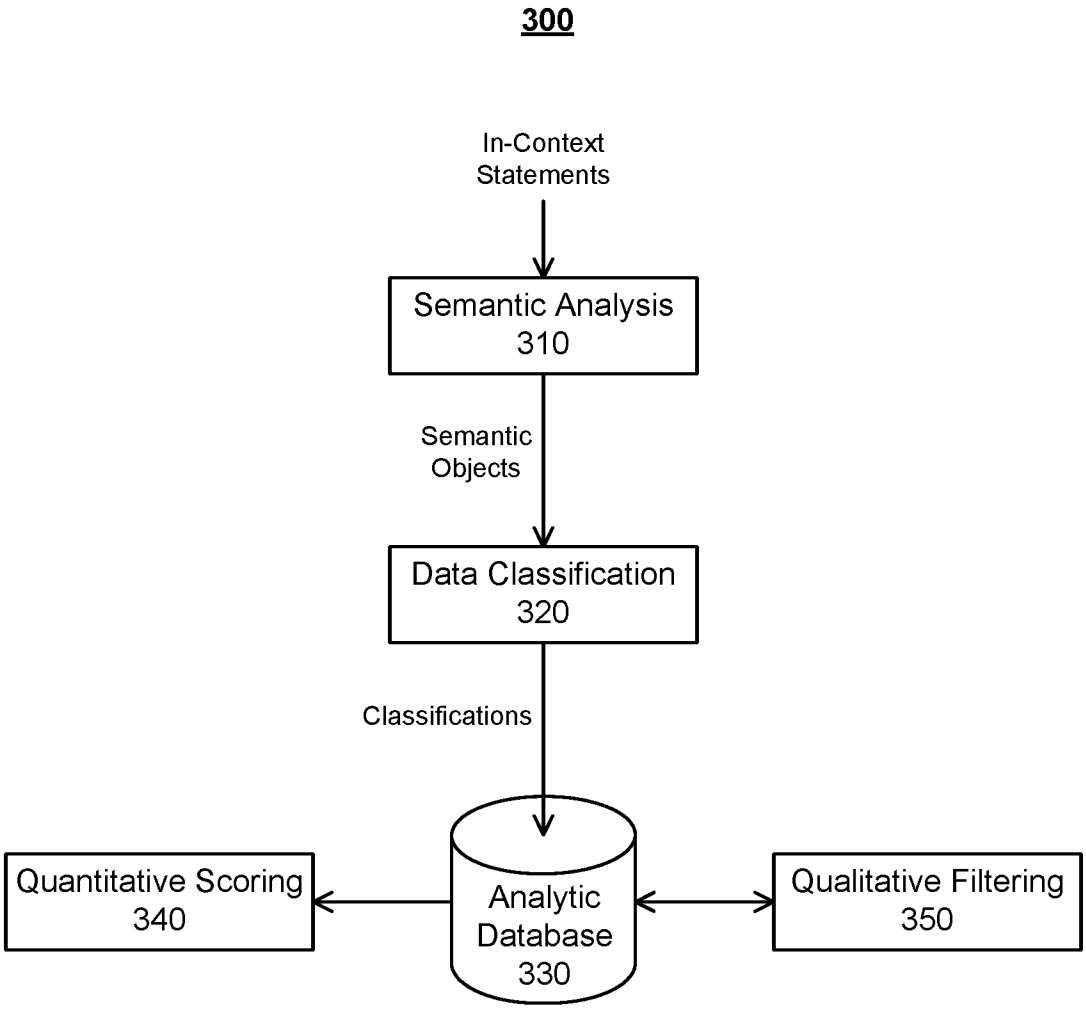

300
In-Context Statements
Semantic Analysis 310
Semantic Objects
Data Classification 320
Classifications
Analytic Database 330
Quantitative Scoring 340
Qualitative Filtering 350

FIG. 3

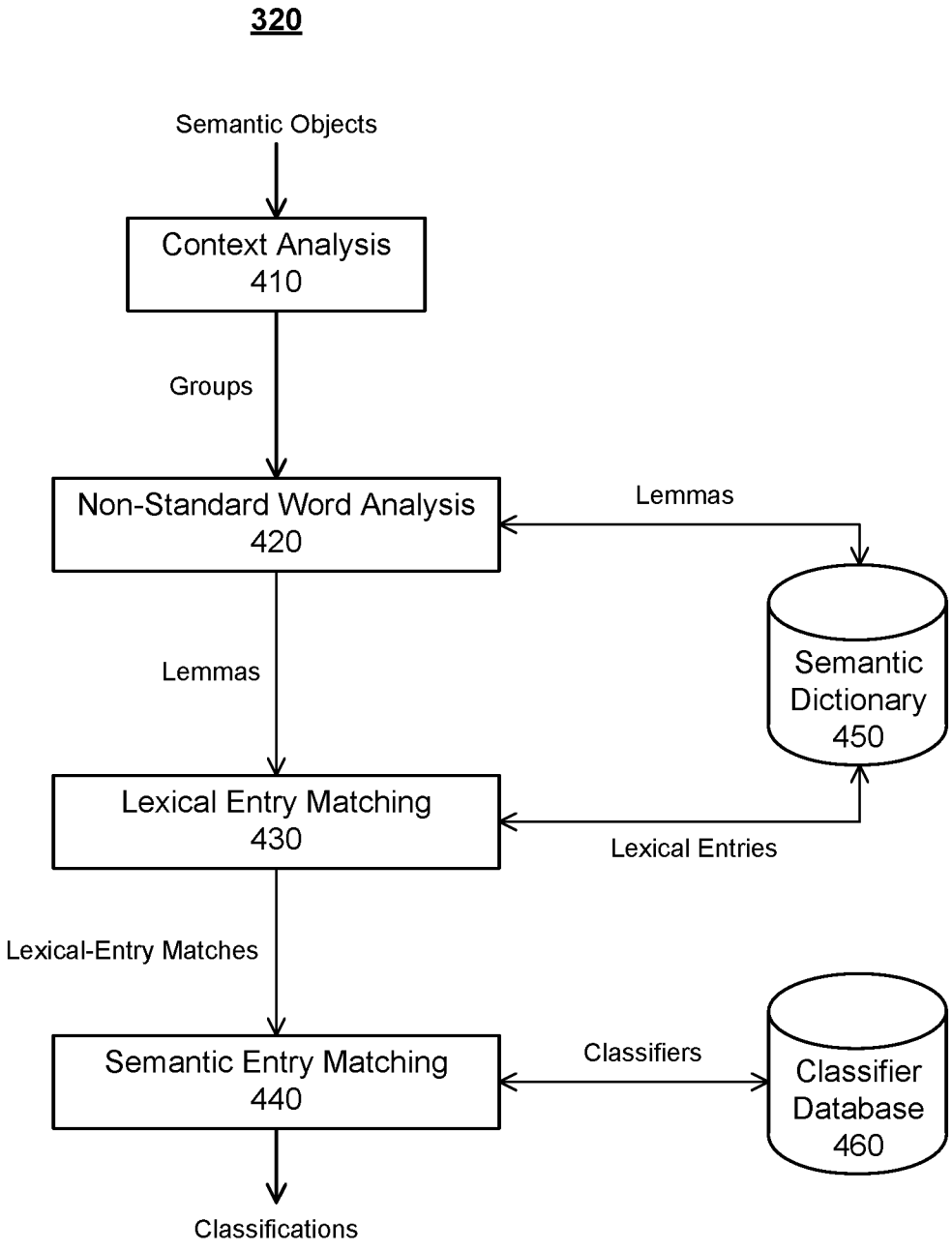

320
Semantic Objects
Context Analysis
410
Groups
Non-Standard Word Analysis
420
Lemmas
Semantic
Dictionary
450
Lemmas
Lexical Entry Matching
430
Lexical Entries
Lexical-Entry Matches
Semantic Entry Matching
440
Classifiers
Classifier
Database
460
Classifications

FIG. 4

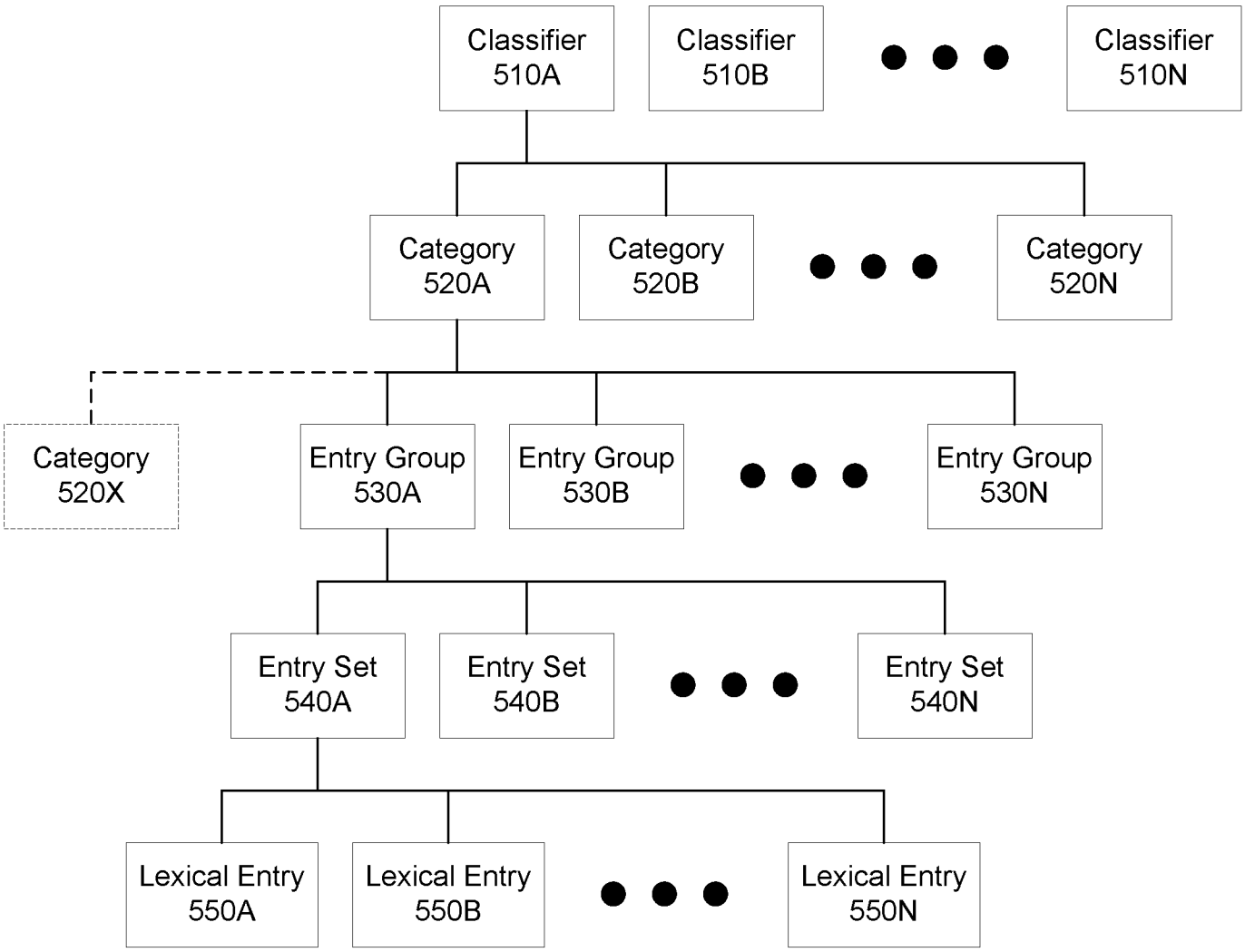

Classifier 510A
Classifier 510B
Classifier 510N
Category 520A
Category 520B
Category 520N
Category 520X
Entry Group 530A
Entry Group 530B
Entry Group 530N
Entry Set 540A
Entry Set 540B
Entry Set 540N
Lexical Entry 550A
Lexical Entry 550B
Lexical Entry 550N

FIG. 5

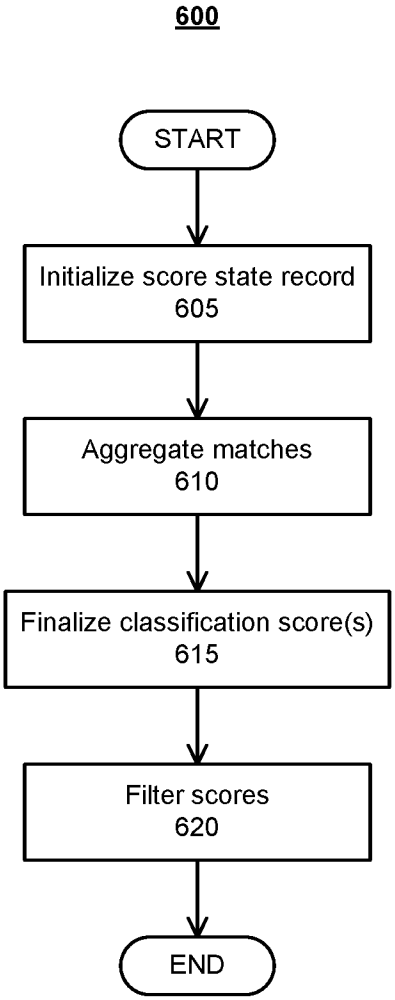
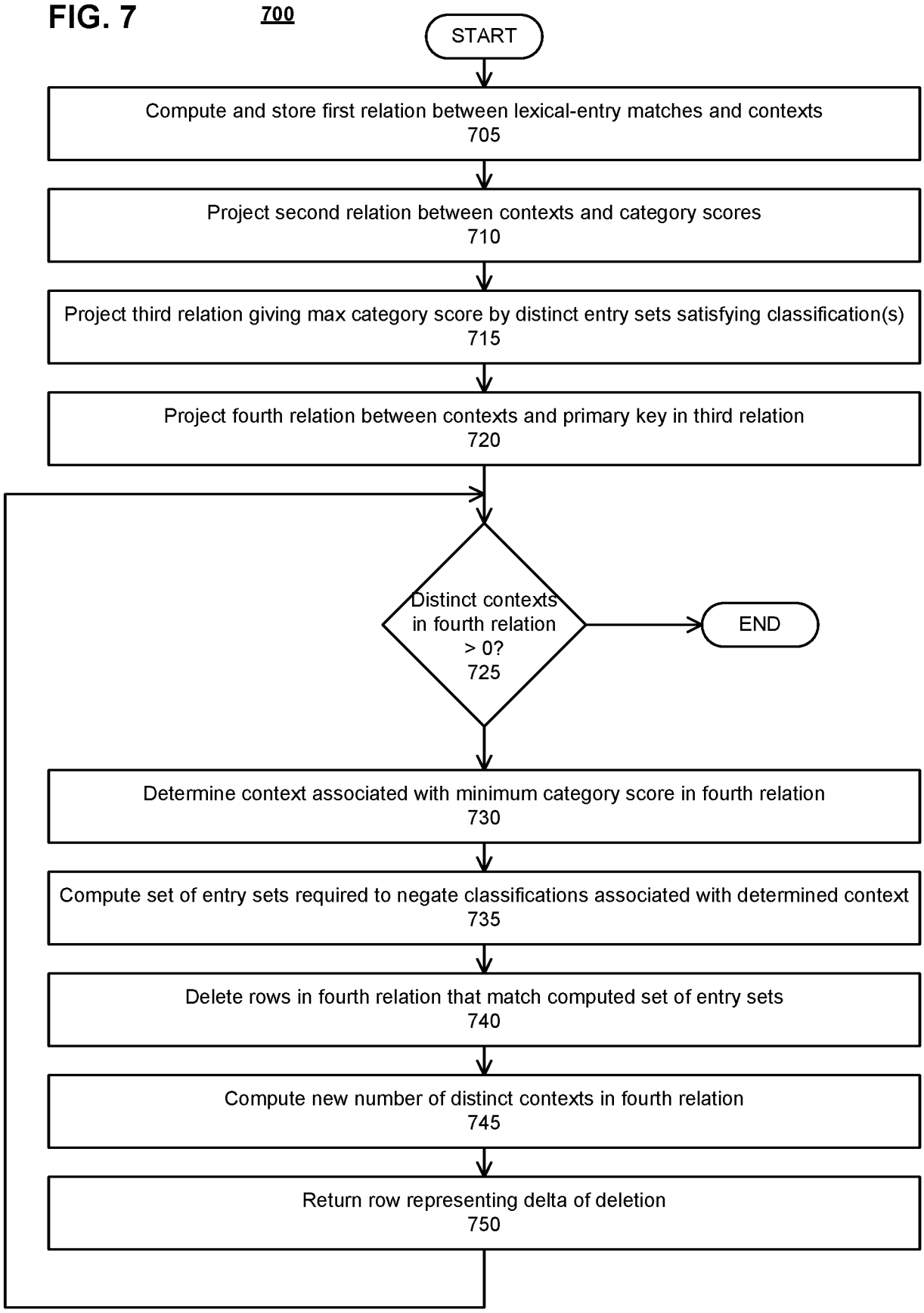

600
START
Initialize score state record
605
Aggregate matches
610
Finalize classification score(s)
615
Filter scores
620
END

FIG. 6

FIG. 7


700
START
Compute and store first relation between lexical-entry matches and contexts
705
Project second relation between contexts and category scores
710
Project third relation giving max category score by distinct entry sets satisfying classification(s)
715
Project fourth relation between contexts and primary key in third relation
720
Distinct contexts in fourth relation > 0?
725
END
Determine context associated with minimum category score in fourth relation
730
Compute set of entry sets required to negate classifications associated with determined context
735
Delete rows in fourth relation that match computed set of entry sets
740
Compute new number of distinct contexts in fourth relation
745
Return row representing delta of deletion
750

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2020/017064

A. CLASSIFICATION OF SUBJECT MATTER

IPC(8) - G06F 40/221; G06F 40/237; G06F 40/284; G06F 40/30 (2020.01)

CPC - G06F 40/211; G06F 21/50; G06F 40/237; G06F 40/30; H04L 63/1408 (2020.02)

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)

See Search History document

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

USPC - 704/9; 704/257; 704/270; 707/755; 726/1 (keyword delimited)

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

See Search History document

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2016/0132492 A1 (ALIBABA GROUP HOLDING LIMITED) 12 May 2016 (12.05.2016) entire document	1-22
A	US 8,214,199 B2 (ANISMOVICH et al) 03 July 2012 (03.07.2012) entire document	1-22
A	US 2014/0201838 A1 (DB NETWORKS, INC.) 17 July 2014 (17.07.2014) entire document	1-22
A	WO 2017/127850 A1 (HASAN SYED KAMRAN) 27 July 2017 (27.07.2017) entire document	1-22
A	US 2016/0239480 A1 (VIRTUOZ SA) 18 August 2016 (18.08.2016) entire document	1-22
A	US 2017/0010829 A1 (WISTRON CORPORATION) 12 January 2017 (12.01.2017) entire document	1-22
A	US 2009/0024604 A1 (ZHAO et al) 22 January 2009 (22.01.2009) entire document	1-22

☐ 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
16 April 2020	28 APR 2020
Name and mailing address of the ISA/US Mail Stop PCT, Attn: ISA/US, Commissioner for Patents P.O. Box 1450, Alexandria, VA 22313-1450 Facsimile No. 571-273-8300	Authorized officer Blaine R. Copenheaver PCT Helpdesk: 571-272-4300 PCT OSP: 571-272-7774