

(19) **DANMARK**



Patent- og  
Varemærkestyrelsen

(12)

## Oversættelse af europæisk patentskrift

(10) **DK/EP 3729255 T3**

- 
- (51) Int.Cl.: **H 04 L 67/50 (2022.01)** **G 06 Q 30/0201 (2023.01)** **H 04 L 67/306 (2022.01)**  
**H 04 L 67/02 (2022.01)**
- (45) Oversættelsen bekendtgjort den: **2024-01-02**
- (80) Dato for Den Europæiske Patentmyndigheds bekendtgørelse om meddelelse af patentet: **2023-09-27**
- (86) Europæisk ansøgning nr.: **18890683.8**
- (86) Europæisk indleveringsdag: **2018-12-12**
- (87) Den europæiske ansøgnings publiceringsdag: **2020-10-28**
- (86) International ansøgning nr.: **US2018065155**
- (87) Internationalt publikationsnr.: **WO2019125858**
- (30) Prioritet: **2017-12-22 US 201762609753 P**
- (84) Designerede stater: **AL AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HR HU IE IS IT LI LT LU LV MC MK MT NL NO PL PT RO RS SE SI SK SM TR**
- (73) Patenthaver: **6Sense Insights, Inc., 450 Mission Street, Suite 201, San Francisco, California 94105, USA**
- (72) Opfinder: **BAJARIA, Viral, c/o 6Sense Insights, Inc. , 450 Mission Street, Suite 201, San Francisco, California 94105, USA**  
**SHAH, Premal, c/o 6Sense Insights, Inc. , 450 Mission Street, Suite 201, San Francisco, California 94105, USA**  
**VIJAYVARGIYA, Nitin, c/o 6Sense Insights, Inc. , 450 Mission Street, Suite 201, San Francisco, California 94105, USA**  
**CHANG, Dustin, c/o 6Sense Insights, Inc. , 450 Mission Street, Suite 201, San Francisco, California 94105, USA**
- (74) Fuldmægtig i Danmark: **Budde Schou A/S, Dronningens Tværgade 30, 1302 København K, Danmark**
- (54) Benævnelse: **MAPNING AF ENTITETER TIL KONTI**
- (56) Fremdragne publikationer:  
**WO-A1-2009/017942**  
**US-A1- 2014 012 976**  
**US-A1- 2016 142 272**  
**US-A1- 2016 239 868**  
**US-A1- 2017 243 028**  
**US-A1- 2017 364 931**  
**US-B1- 6 839 680**



# DESCRIPTION

## CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims priority to U.S. Provisional Patent App. No. 62/609,753, filed on December 22, 2017.

## BACKGROUND

### Field of the Invention

[0002] The invention relates to de-anonymizing anonymous Internet Protocol addresses.

### Description of the Related Art

[0003] One significant and well-known problem that arises in the context of Internet communications is the ability for Internet users to remain virtually anonymous. While this anonymity is arguably good for free speech, it is bad for business. For example, unless a visitor to a website chooses to accurately identify himself or herself, an operator of the website may know little to nothing about the identity of that visitor.

[0004] In most cases, all the operator of the website will know about the visitor is an Internet Protocol (IP) address used by the visitor's browsing device, a domain of the visitor (e.g., a domain with which the IP address is associated), and/or, if available, one or more "cookies" (data stored on the user's device by a website during a visit to that website). This is generally not enough information to initiate contact with the visitor, for example, to market a product or service to the visitor.

[0005] In contrast, in a brick-and-mortar store, a merchant may physically approach visitors to immediately initiate contact with that visitor, for example, to begin marketing a product or service. Thus, the Internet, while convenient for marketing and commerce, presents a problem for merchants whose visitors remain anonymous. This is a problem that simply does not exist in the brick-and-mortar context.

[0006] Accordingly, a solution is needed for piercing the unique shroud of anonymity afforded to visitors by the Internet. Preferably, such a solution would enable merchants to determine contact information for otherwise anonymous Internet visitors. Such a solution could be especially advantageous for predictive marketing methods, such as those described in U.S.

Patent Pub. No. 2013/0204663, filed on February 7, 2013.

**[0007]** US 2016/142272 A1 discloses techniques for transforming information about unique visits to a website or interactions with a mobile application by a single individual on different devices into unified analytical data representing these visits. One or more visitor stitching models can be used to quantify unique visits as belonging to the same person with a certain level of confidence. In addition to maintaining separate records for each unique visit, a unified record is generated, which represents a combination of the unique visits by the same person. Various analytics can then be performed on the unique and unified records. The unified records are favored over the separate data records wherever the stitching model corresponding to the unified record provides a certainty that meets or exceeds a user-specified confidence level. The visitor stitching model used for the analytics is considered non-committal because different models can be employed or none at all, depending on user preferences..

**[0008]** US 2016/239868 A1 discloses systems and methods for associating a plurality of Internet-enabled devices with a common user profile for targeting Internet content or advertising. One method includes: receiving, from a plurality of Internet-enabled devices, a plurality of requests for electronic content or advertising; extracting, from each of the plurality of requests, a source IP address and a unique identifier associated with the respective Internet-enabled device; identifying each possible pair of devices from which requests were received; calculating for each possible pair of devices a probability that the pair of devices are owned or operated by a common user; and prompting a user to either confirm a characteristic of a prior browsing session or to log-in to an account associated with the common user based on a comparison of the calculated probability to one or more thresholds..

**[0009]** In US 2017/364931 A1 a distributed model generation system includes a master node that estimates parameter sets for a topic classification (TC) model. The estimated parameter sets are loaded into a queue. Multiple training nodes download the estimated parameter sets from the queue for training associated TC models. The training nodes generate model performance values for the trained TC models and send the model performance values back to the master node. The master node uses the model performance values and the associated parameter sets to estimate additional TC model parameter sets. The master node estimates new parameter sets until a desired model performance value is obtained. The master node may use a Bayesian optimization to more efficiently estimate the parameter sets and may distribute the high processing demands of model training and testing operations to the training nodes.

## SUMMARY

**[0010]** The invention comprises a method for de-anonymizing anonymous Internet Protocol addresses according to claim 1, a system for de-anonymizing anonymous Internet Protocol addresses according to claim 18 and non-transitory computer-readable medium according to claim 19.

**[0011]** Advantageous embodiments are set out in the dependent claims. The scope of the invention is defined by the appended claims.

## **BRIEF DESCRIPTION OF THE DRAWINGS**

**[0012]** 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:

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

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

FIG. 3 illustrates a process for mapping anonymous entities to known accounts, according to an embodiment;

FIGS. 4A and 4B illustrate a process for determining a source confidence value for a plurality of sources;

FIG. 5 illustrates the interactions between various data structures in one example implementation of the process illustrated in FIG. 3, according to an embodiment; and

FIG. 6 illustrates the interactions between various data structures in one example implementation of a web service, according to an embodiment.

## **DETAILED DESCRIPTION**

**[0013]** In an embodiment, systems, methods, and non-transitory computer-readable media are disclosed for mapping anonymous entities (e.g., IP address, domain, cookie, and/or device) to known accounts (e.g., companies). 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.

### **1. System Overview**

## 1.1. Infrastructure

**[0014]** FIG. 1 illustrates an example infrastructure in which a system that maps anonymous entities (e.g., IP address, domain, cookie, and/or device) to known accounts (e.g., companies) may operate, according to an embodiment. The infrastructure may comprise a platform 110 (e.g., one or more server devices) which hosts and/or executes one or more of the various functions, processes, methods, and/or software modules described herein. Platform 110 may comprise dedicated servers, or may instead comprise cloud instances, which utilize shared resources of one or more servers. These servers or cloud instances may be collocated and/or geographically distributed. Platform 110 may also comprise or be communicatively connected to a server application 112 and/or one or more databases 114. In addition, platform 110 may be communicatively connected to one or more user systems 130 via one or more networks 120. Platform 110 may also be communicatively connected to one or more external systems 140 (e.g., data sources, websites, web services, other platforms, etc.) via one or more networks 120.

**[0015]** Network(s) 120 may comprise the Internet, and platform 110 may communicate with user system(s) 130 and external system(s) 140 through the Internet using standard transmission protocols, such as HyperText Transfer Protocol (HTTP), Secure HTTP (HTTPS), File Transfer Protocol (FTP), FTP Secure (FTPS), SSH FTP (SFTP), and the like, as well as proprietary protocols. While platform 110 is illustrated as being connected to various systems through a single set of network(s) 120, it should be understood that platform 110 may be connected to the various systems via different sets of one or more networks. For example, platform 110 may be connected to a subset of user systems 130 and/or external systems 140 via the Internet, but may be connected to one or more other user systems 130 and/or external systems 140 via an intranet. Furthermore, while only a few user systems 130 and external systems 140, one server application 112, and one set of database(s) 114 are illustrated, it should be understood that the infrastructure may comprise any number of user systems, external systems, server applications, and databases.

**[0016]** User system(s) 130 may comprise any type or types of computing devices capable of wired and/or wireless communication, including without limitation, desktop computers, laptop computers, tablet computers, smart phones or other mobile phones, servers, game consoles, televisions, set-top boxes, electronic kiosks, point-of-sale terminals, Automated Teller Machines, and the like.

**[0017]** Platform 110 may comprise web servers which host one or more websites and/or web services. In embodiments in which a website is provided, the website may comprise one or more user interfaces, including, for example, webpages generated in HyperText Markup Language (HTML) or other language. Platform 110 transmits or serves these user interfaces in response to requests from user system(s) 130. In some embodiments, these user interfaces

may be served in the form of a wizard, in which case two or more user interfaces may be served in a sequential manner, and one or more of the sequential user interfaces may depend on an interaction of the user or user system with one or more preceding user interfaces. The requests to platform 110 and the responses from platform 110, including the user interfaces, may both be communicated through network(s) 120, which may include the Internet, using standard communication protocols (e.g., HTTP, HTTPS, etc.). These user interfaces or web pages may comprise a combination of content and elements, such as text, images, videos, animations, references (e.g., hyperlinks), frames, inputs (e.g., textboxes, text areas, checkboxes, radio buttons, drop-down menus, buttons, forms, etc.), scripts (e.g., JavaScript), and/or the like, including elements comprising or derived from data stored in one or more databases (e.g., database(s) 114) that are locally and/or remotely accessible to platform 110. Platform 110 may also respond to other requests from user system(s) 130.

**[0018]** Platform 110 may further comprise, be communicatively coupled with, or otherwise have access to one or more database(s) 114. For example, platform 110 may comprise one or more database servers which manage one or more databases 114. A user system 130 or server application 112 executing on platform 110 may submit data (e.g., user data, form data, etc.) to be stored in database(s) 114, and/or request access to data stored in database(s) 114. Any suitable database may be utilized, including without limitation MySQL™, Oracle™, IBM™, Microsoft SQL™, Sybase™, Access™, and the like, including cloud-based database instances and proprietary databases. Data may be sent to platform 110, for instance, using the well-known POST request supported by HTTP, via FTP, etc. This data, as well as other requests, may be handled, for example, by server-side web technology, such as a servlet or other software module (e.g., server application 112), executed by platform 110.

**[0019]** In embodiments in which a web service is provided, platform 110 may receive requests from external system(s) 140, and provide responses in Javascript Object Notation (JSON), eXtensible Markup Language (XML), and/or any other suitable or desired format. In such embodiments, platform 110 may provide an application programming interface (API) which defines the manner in which user system(s) 130 and/or external system(s) 140 may interact with the web service. Thus, user system(s) 130 and/or external system(s) 140 (which may themselves be servers), can define their own user interfaces, and rely on the web service to implement or otherwise provide the backend processes, methods, functionality, storage, and/or the like, described herein. For example, in such an embodiment, a client application 132 executing on one or more user system(s) 130 may interact with server application 112 executing on platform 110 to execute one or more or a portion of one or more of the various functions, processes, methods, and/or software modules described herein. Client application 132 may be "thin," in which case processing is primarily carried out server-side by server application 112 on platform 110. A basic example of a thin client application is a browser application, which simply requests, receives, and renders webpages at user system(s) 130, while server application on platform 110 is responsible for generating the webpages and managing database functions. Alternatively, client application 132 may be "thick," in which case processing is primarily carried out client-side by user system(s) 130. It should be understood that client application 132 may perform an amount of processing, relative to server application

112 on platform 110, at any point along this spectrum between "thin" and "thick," depending on the design goals of the particular implementation. In any case, the application described herein, which may wholly reside on either platform 110 (e.g., in which case application 112 performs all processing) or user system(s) 130 (e.g., in which case application 132 performs all processing) or be distributed between platform 110 and user system(s) 130 (e.g., in which case server application 112 and client application 132 both perform processing), can comprise one or more executable software modules that implement one or more of the processes, methods, or functions of the application(s) described herein.

## **1.2. Example Processing Device**

**[0020]** 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 mechanisms, processes, methods, or functions (e.g., to store and/or execute the application or one or more software modules of the application) described herein, and may represent components of platform 110, user system(s) 130, external system(s) 140, and/or other processing devices described herein. System 200 can be a server or any 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.

**[0021]** 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, or a coprocessor. Such auxiliary processors may be discrete processors or may be integrated with the 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.

**[0022]** 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 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, or standards promulgated by the Institute of Electrical and Electronics Engineers (IEEE) including

IEEE 488 general-purpose interface bus (GPIB), IEEE 696/S-100, and the like.

**[0023]** 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 of the application 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).

**[0024]** Secondary memory 220 may optionally include an internal memory 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 the like.

**[0025]** Removable storage medium 230 is a non-transitory computer-readable medium having stored thereon computer-executable code (e.g., disclosed software modules) and/or data. The computer software or data stored on removable storage medium 230 is read into system 200 for execution by processor 210.

**[0026]** 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, an external storage medium 245 and 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 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), or flash memory (block-oriented memory similar to EEPROM).

**[0027]** 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 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 network interface card (NIC),

a wireless data card, a communications port, an infrared interface, an IEEE 1394 fire-wire, or any other device capable of interfacing system 200 with a network 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.

**[0028]** 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.

**[0029]** Computer-executable code (i.e., computer programs, such as the disclosed application, or software modules) is stored in main memory 215 and/or the 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.

**[0030]** 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 (e.g., software and computer programs) to 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 mediums are means for providing executable code, programming instructions, and software to system 200.

**[0031]** 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 the features and functions described elsewhere herein.

**[0032]** 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, keyboards, touch screens or other touch-sensitive devices, biometric sensing devices, computer mice, trackballs, pen-based pointing devices, and the like. Examples of output devices include, without limitation, 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 the like.

**[0033]** System 200 may also include optional wireless communication components that facilitate wireless communication over a voice network and/or a data network. The wireless communication components may comprise an antenna system 270, a radio system 265, and a baseband system 260. In system 200, RF signals are transmitted and received over the air by antenna system 270 under the management of radio system 265.

**[0034]** In one 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.

**[0035]** 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.

**[0036]** 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 codes 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.

**[0037]** 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 application or software modules) 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 functions of the disclosed embodiments. For example, data storage areas 215 or 220 may include various software modules.

### **1.3. Example Data Structures**

**[0038]** The term "entity" may be used herein to refer to data that is associated with a visitor or contact throughout an activity. For example, an entity may include, without limitation, an IP address used by a visitor's or contact's device (e.g., a user system 130) for online communications, a domain name (e.g., for a website and/or email domain) associated with a visitor or contact during online communications, and/or a cookie that has been stored on a visitor's device during browsing. Such data may be permanently, semi-permanently, or temporarily associated with the visitor or contact, and typically will not uniquely identify the visitor or contact. For example, an IP address may be a static IP address that is always associated with a particular visitor's device, or a dynamic IP address that may be temporarily assigned to one device and then subsequently reassigned by an Internet service provider (ISP) to a different device. In addition, a particular visitor may utilize multiple IP addresses (e.g., one IP address while the visitor is at work and another IP address while the visitor is at home). Similarly, one domain name may be associated with multiple visitors or contacts, and one visitor or contact may be associated with multiple domain names. As another example, a visitor may be associated with multiple cookies at once and over time, and each cookie may exist for only a limited time on a visitor's device before being deleted (e.g., manually by a user of the device, automatically after an expiration period or upon termination of a session, etc.).

**[0039]** The term "event" may be used herein to refer to any activity from any source. Activities may include, without limitation, accessing a webpage of a website, filling out an online form, purchasing a product via an ecommerce site, contacting a merchant or salesperson (e.g., via an online form, electronic mail (email) message, instant-messaging chat, Short Message Service (SMS) text, Multimedia Messaging Service (MMS) message, etc.), and/or any other online interaction. Activities could also include offline activities, such as attending a trade show or seminar, calling a customer-service call center, making an offline purchase, and/or the like. Sources may include, without limitation, a website, a marketing automation platform (MAP), a cookie-tracking source, vendor data, and/or the like. As an example, a visitor to a website may fill out an online form within a webpage of the website, and this activity may be communicated to the application, along with an entity representing the visitor (e.g., the IP address used by the visitor's device to communicate with the website, a domain name associated with the visitor's ISP or enterprise, a string representing the partial or entire contents of a tracking cookie stored on the visitor's device, etc.), as an event.

**[0040]** An event may be implemented as a data structure (e.g., vector, linked list, array, delimited string, etc.) which comprises a representation of the entity (e.g., IP address, domain, and/or cookie) that represents the visitor and a timestamp representing the time at which the activity occurred. It should be understood that this data structure could also comprise any additional data, which may aid in mapping the visitor to an account, identifying or categorizing

the activity, and/or the like. For example, each event data structure could also comprise a representation of the activity, source information indicating the source of the event and/or activity, and/or the like. In addition, the event may comprise visitor-specific information, if, for example, the visitor submitted a form or logged into his or her account during a visit. In such a case, the source can incorporate user-specific information, entered in the form or acquired from a user profile associated with the logged-in account, in the event information recorded for the event.

**[0041]** The term "mapping" may be used herein to refer to each association between an entity and an account (e.g., company name or other company identifier). Each mapping indicates that there is at least some reason to believe that the entity has been used by the associated account in the past, and therefore, that current or future activity by that entity may represent online activity by that account. A mapping may be implemented as a data structure which comprises a representation of the entity (e.g., IP address, domain, and/or cookie) and an account identifier. It should be understood that this data structure could also comprise any additional data, including a span, delay, spread, strength, source weight, overlap, and/or confidence.

**[0042]** In an embodiment, each mapping data structure comprises or is associated with a span value. The span value represents a time period over which the respective mapping was active. For example, if the mapping was derived from three events that occurred on January 1, 2017, March 3, 2017, and November 4, 2017, the span value may be calculated as the number of days between the first and last events (i.e., January 1, 2017 through November 4, 2017, or three-hundred-seven days). It should be understood that the span value may be calculated and represented according to any level of granularity (e.g., number of days, hours, minutes, seconds, milliseconds, etc.), but, for simplicity, will be measured in days in the present disclosure.

**[0043]** In an embodiment, each mapping data structure comprises or is associated with a delay value. The delay value represents a time period since the respective mapping was last active. For example, if on January 1, 2018, no event has occurred for a mapping since November 4, 2017, the delay value may be calculated as the number of days between November 4, 2017 and January 1, 2018 (i.e., fifty-eight days). It should be understood that the delay value may be calculated and represented according to any level of granularity (e.g., number of days, hours, minutes, seconds, milliseconds, etc.), but, for simplicity, will be measured in days in the present disclosure.

**[0044]** In an embodiment, each mapping data structure comprises or is associated with a spread value. The spread value represents a time period that indicates the frequency with which the mapping was active over its span value. For example, if the mapping was derived from three or more events that occurred on January 1, 2017, March 3, 2017, and November 4, 2017, the spread value may be three days. It should be understood that the spread value may be calculated and represented according to any level of granularity (e.g., number of days over which the events occurred, or total duration of the events in days, hours, minutes, seconds,

milliseconds, etc.), but, for simplicity, will be measured in days in the present disclosure.

**[0045]** In an embodiment, each mapping data structure comprises or is associated with a strength value. The strength value represents the frequency with which the mapping was active over its span value. For example, if the mapping was derived from one event that occurred on January 1, 2017, two events that occurred on March 3, 2017, and three events that occurred on November 4, 2017, the strength value may be six, since there were six events. Notably, the strength value is different than the spread value since it indicates the number of events, rather than the number of days (e.g., multiple events may occur in one day) or other amount of time. It should be understood that, in the event that the spread value is a number of days, the strength value may equal or exceed the spread value, but should never be less than the spread value.

**[0046]** In an embodiment, each mapping data structure comprises or is associated with a source weight. The source weight represents a relative accuracy of the source or source(s) of the events, from which the respective mapping was derived. The calculation of the source weight is described elsewhere herein.

**[0047]** In an embodiment, each mapping data structure comprises or is associated with an overlap value. The overlap value represents the number of sources that corroborate the respective mapping. For example, if the mapping was derived from events that were received from three different sources or the mapping corresponds to other mappings derived from two other different sources, the overlap value would be three total sources (or two corroborating sources).

**[0048]** In an embodiment, each mapping data structure comprises or is associated with a confidence value which has been computed based on one or more factors (e.g., the span, delay, spread, strength, source weight, and/or overlap statistics), as described elsewhere herein. In this context, the confidence value for a particular mapping represents a measure of confidence that a visitor or contact, who has used, is using, or will use the respective entity, is associated with the respective account. The confidence value will primarily be described herein as a decimal value between zero and one. However, it should be understood that the range of possible confidence values can be easily converted to any desired range of values, with corresponding modifications to any disclosed calculations which rely upon the confidence value.

**[0049]** The term "event pool" is used herein to refer to a data structure in which all of the raw event data structures are stored. For example, whenever an event is received from a source, a representation of the event may be stored in the event pool as the described event data structure. Each event data structure may comprise, for example, a timestamp (e.g., indicating the date and/or time on which the activity, underlying the event, occurred), the entity (e.g., IP address, website and/or email domain, and/or cookie), the source from which the event was received and/or at which the event occurred, a hash of an email address associated with the event (if any), a company name (if any), a geographical location (based on a user entry during

the underlying activity or a user profile if the user was logged in to an account for the activity), and/or the geographical-IP location if the entity includes an IP address (i.e., each IP address can be associated with a country, since different IP blocks are allocated to different countries). Each event data structure could also comprise additional information (e.g., relevant information from a user entry or user profile), such as city, state, Zip code, Global Positioning System (GPS) coordinates (e.g., latitude, longitude, and/or elevation), and/or the like. Prior to entry in the event pool, one or more fields of each event data structure may be normalized, as described elsewhere herein.

**[0050]** Since every single event is individually stored in the event pool and the number of events being collected will generally be extremely large, the event pool will be correspondingly large. Thus, in order to process the events in a more condensed state, the application may utilize a "mapping pool." Specifically, the term "mapping pool" is used herein to refer to a data structure which aggregates events, stored in the event pool, to form mappings.

## **2. Process Overview**

**[0051]** Embodiments of processes for mapping anonymous entities to known accounts 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, for example, as the application discussed herein (e.g., server application 112, client application 132, and/or a distributed application comprising both server application 112 and client application 132), which may be executed wholly by processor(s) of platform 110, wholly by processor(s) of user system(s) 130, or may be distributed across platform 110 and user system(s) 130 such that some portions or modules of the application are executed by platform 110 and other portions or modules of the application are executed by user system(s) 130. The described process 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. In addition, the disclosed application may be built upon or interfaced with one or more existing systems.

**[0052]** 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 as

defined in the appended claims. 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 as defined in the appended claims.

**[0053]** FIG. 3 illustrates a process 300 for mapping anonymous entities to known accounts, according to an embodiment. While process 300 is illustrated with a certain arrangement and ordering of steps, process 300 may be implemented with fewer, more, or different steps and a different arrangement and/or ordering of steps.

## **2.1. Collection into Event Pool**

**[0054]** In step 310, the application receives events from one or more sources (e.g., external systems 140 and/or internal sources on platform 110) and collects them into an event pool. As described elsewhere herein, each event may be received as a data structure comprising at least an entity and a timestamp. Each event may be stored in a data structure within the event pool, and may comprise the entity (e.g., IP address, domain, and/or cookie), the timestamp, an identifier of the source of the event, and, optionally, additional information (e.g., a geographical-IP location associated with an IP address in the entity, a company or user name, etc.).

**[0055]** Notably, the raw event data may be noisy. Thus, in an embodiment, the event data is normalized as it is entered into the event pool or after it has been entered into the event pool. For example, normalization may be performed on the geographical-IP locations (e.g., converting all variations of the same country name into a single, canonical country name using nomenclature and/or grammar techniques), the company or account names (e.g., converting all variations of the same company or account name into a single, canonical name using nomenclature and/or grammar techniques), domain names (e.g., discarding any portion of the domain name preceding the second-level domain name), and/or other fields. Fuzzy matching may be used to identify variations of these field values. The normalization can be performed across fields in all of the entries in the event pool to maximize the subsequent aggregation process.

**[0056]** As an example of normalization, one or more aliases of a master domain (e.g., abc.com is an alias of abc.go.com) may be detected in incoming event data and normalized to the master domain prior to entry into the event pool. Advantageously, this avoids the same domain being analyzed as different domains. In an embodiment, an alias set can be constructed using one or more (including all) of the following techniques (e.g., implemented by the application):

1. (1) Send a request to a first domain (e.g., abc.com) to see if it redirects to a second domain (e.g., abc.go.com). If the request to the first domain redirects to the second domain, the first domain is an alias of the second domain (e.g., abc.com is an alias of abc.go.com).

2. (2) Retrieve the Domain Name System (DNS) IP address for a domain (e.g., using a DNS lookup), and construct a set of domains associated to the retrieved IP address (e.g., using a reverse IP lookup, such that DNS IP => set of domains). Once the set of domains has been constructed, the application can send a request to each domain in the set of domains, and check the similarity between the returned webpages (e.g., based on style). For example, if the set of domains for the IP address 1.1.1.1 is [a.com, b.com, c.com], the webpages at a.com, b.com, and c.com can be retrieved, and the styles of those retrieved webpages can be compared. If the styles are identical or similar, the application can determine that the set of domains [a.com, b.com, c.com] is a set of aliases, and select one of the domains from the set of domains to act as the master domain. For instance, the application may randomly select a.com as the master domain, and normalize all future event data to the master domain by modifying any event data associated with alias domains b.com and c.com to be associated with the master domain a.com (any event data associated with a.com will remain associated with a.com).
3. (3) Construct a set of domains (e.g., seen in the event data) based on their string distances from each other, send a request to each domain in the set of domains, and check the similarity between the returned webpages (e.g., based on style). The similarity matching may be the same as or different than the similarity matching in technique (2) above. For example, the application may identify JonBakery.com, JonBakes.com, and JonBaker.com as a set of domains, representing potential aliases of each other, based on the string distance between the domain names. The webpages at JonBakery.com, JonBakes.com, and JonBaker.com can be retrieved, and the styles of those retrieved webpages can be compared. If the styles are identical or similar, the application can determine that the set of domains [JonBakery.com, JonBakes.com, and JonBaker.com] is a set of aliases, and select one of the domains from the set of domains to act as the master domain. For instance, the application may randomly select JonBaker.com as the master domain, and normalize all future event data to the master domain by modifying any event data associated with alias domains JonBakery.com and JonBakes.com to be associated with the master domain JonBaker.com (any event data associated with JonBaker.com will remain associated with JonBaker.com).

**[0057]** The sets of domains, determined to be aliases using techniques (1), (2), and/or (3) above, can be merged into a single set of alias domains. Then, in an embodiment, merged sets of alias domains with unreasonably high numbers of domains in them (e.g., sets comprising a number of domains exceeding a predetermined threshold value) are filtered out from the collected event data. Specifically, event data associated with any domains in any filtered-out sets of domains can be excluded from the event pool. This is because large sets of alias domains are indicative of redirection by domain sellers, hosting services, social media platforms, and/or the like, rather than association with a valid account.

**[0058]** In addition, event data may be filtered out based on one or more other criteria. For

example, event data associated with an IP address within ranges of IP addresses associated with web service providers (e.g., Amazon™, Google™, Microsoft™, etc.) may be excluded from the event pool. As another example, event data, associated with an entity from which bot-like activity patterns have been detected, may be excluded from the event pool. Bot-like activity patterns may comprise an unreasonably high number of requests, web-scraping behavior, periodic events, and/or the like.

## 2.2. Aggregation into Mappings

**[0059]** In step 320, the application aggregates events, stored in the event pool, into a mapping pool. For ease of understanding, this step will be described using a concrete, non-limiting example. Assume that the event pool comprises the entries represented in Table 1 below. Notably, not all fields of the entries are necessarily shown in the included tables, and the entries in these tables only represent a tiny fraction of the entries that would exist in an actual pool and for only a single entity. The entries of an actual implementation may comprise more, fewer, or different fields and/or different representations of field values than those shown in the tables.

Table 1

| Date       | IP Address | Domain  | Geo-IP        | Source  |
|------------|------------|---------|---------------|---------|
| 12/30/2016 | 1.1.1.1    | xyz.com | United States | Source1 |
| 01/01/2017 | 1.1.1.1    | abc.com | United States | Source2 |
| 01/03/2017 | 1.1.1.1    | abc.com | United States | Source3 |
| 01/04/2017 | 1.1.1.1    | abc.com | United States | Source3 |
| 01/06/2017 | 1.1.1.1    | abc.com | United States | Source3 |
| 01/14/2017 | 1.1.1.1    | abc.com | United States | Source3 |
| 01/15/2017 | 1.1.1.1    | abc.com | United States | Source3 |
| 01/20/2017 | 1.1.1.1    | xyz.com | United States | Source1 |
| 01/28/2017 | 1.1.1.1    | abc.com | United States | Source3 |
| 01/28/2017 | 1.1.1.1    | abc.com | United States | Source3 |
| 01/29/2017 | 1.1.1.1    | abc.com | United States | Source2 |
| 02/02/2017 | 1.1.1.1    | abc.com | United States | Source3 |

**[0060]** In this example, two events, associating the IP address 1.1.1.1 with a domain of xyz.com, have been received from Source1 (e.g., a data vendor), two events, associating the same IP address with a domain of abc.com, have been received from Source2 (e.g., directly from a website, for example, reporting a login), and eight events, associating the same IP address with abc.com, have been received from Source3 (e.g., a marketing automation platform). On February 3, 2017, these twelve events from the event pool may be condensed

into two entries in the mapping pool, as shown in Table 2 below.

Table 2

| IP Address | Company | Span | Delay | Spread | Strength | Overlap | Sources             |
|------------|---------|------|-------|--------|----------|---------|---------------------|
| 1.1.1.1    | XYZ     | 21   | 14    | 2      | 2        | 1       | Source 1            |
| 1.1.1.1    | ABC     | 32   | 1     | 9      | 10       | 2       | Source2,<br>Source3 |

**[0061]** As illustrated in Table 2, all individual events, stored in the event pool, are matched to other events with a matching entity, and aggregated into a single entry in the mapping pool that associates the IP address with an account name. In an embodiment, the account name may be derived from the domain name (e.g., "XYZ" from "xyz.com" and "ABC" from "abc.com"). Statistics (e.g., span value, delay value, spread value, strength value, and overlap value) are calculated for each set of aggregated events and included in the entry in the mapping pool. In this case, two entries with the IP address 1.1.1.1 and the same domain name "xyz.com" have been combined into a single entry with a span value of twenty-one days, a delay value of fourteen days, a spread value of two days, a strength value of two days, an overlap value of one source, and a list comprising an identifier of that one source. Similarly, ten entries with the same IP address, but a different domain name "abc.com" have been combined into a single entry with a span value of thirty-two days, a delay value of one day, a spread value of nine days, a strength value of ten days, an overlap of two sources, and a list comprising an identifier for each of those two sources. In addition, the domain name "xyz.com" has been associated with XYZ company with a normalized company name of "XYZ", and the domain name "abc.com" has been associated with ABC company with a normalized company name of "ABC".

### 2.3. Confidence Calculation for Each Mapping

**[0062]** In step 330, the application calculates confidence values for each mapping in the mapping pool. In an embodiment, an overall confidence value for each mapping is calculated based on a plurality of different confidences. For example, the confidence value may be a combination of a time confidence, a statistical confidence, and/or a source confidence, which are each calculated based on the aggregated information within each mapping.

#### 2.3.1. Time Confidence

**[0063]** The time confidence for a particular mapping may be calculated using Equation 1 below:

$$\frac{\log_3(\text{span}) - \log_3(\text{delay})}{1 + |\log_3(\text{span}) - \log_3(\text{delay})|} \quad [\text{Equation 1}]$$

wherein span is the span value associated with the mapping, and delay is the delay value

associated with the mapping.

**[0064]** Equation 1 produces a sigmoid curve, based on the reasoning that mappings with higher span values should decay at a lower rate than mappings with lower span values. However, at the same time, mappings with low span values should still be captured. For example, if a particular mapping was active yesterday and only yesterday, it will have a high time confidence value (e.g., 0.3), since it is new, but that time confidence value will fall off rapidly if it is not active again in the future. Conversely, a competing mapping that has not been active for the past month, but which was active over an entire year prior to the past month will have an even higher time confidence value (e.g., 0.8), such that it will exceed the new, yet fleeting, mapping in terms of their respective confidence values.

### 2.3.2. Statistical Confidence

**[0065]** The statistical confidence may be calculated using Equation 2 below:

$$\text{weight}_{\text{strength}} * \log_{1000}(\text{strength}) + \text{weight}_{\text{spread}} * \log_{100}(\text{spread}) \quad [\text{Equation 2}]$$

wherein strength is the strength value associated with the mapping, wherein spread is the spread value associated with the mapping, wherein  $\text{weight}_{\text{strength}} + \text{weight}_{\text{spread}} = 1.0$ , and wherein, in an embodiment,  $\text{weight}_{\text{strength}} = \text{weight}_{\text{spread}} = 0.5$ .

**[0066]** Equation 2 uses logarithmic functions to reward statistics such as strength and spread. In practice, the majority of mappings may have low values for these statistics. Thus, those mappings with higher values for these statistics are distinguished and "rewarded" in terms of their respective confidence values.

### 2.3.3. Source Confidence

**[0067]** The source confidence for a particular mapping may be calculated using Equation 3 below, which essentially subtracts the probability of each source, which supplied the event(s) from which the mapping was derived, being wrong:

$$1 - \prod (1 - \text{source\_confidence}_s) \quad [\text{Equation 3}]$$

wherein S is the overlap value for the mapping, and the source\_confidence is a confidence value, between zero and one, that is associated with each source (e.g., a global confidence value for each source based on the past reliability or accuracy of the source) from which events, used to derive the mapping, were received.

**[0068]** The source confidence can be an important variable in the overall confidence value. While some sources may supply trustworthy data based on a reliable methodology, other sources may be incredibly noisy (e.g., lots of inaccurate data mixed in with accurate data). In an embodiment, the source\_confidence value for a particular source may be determined

based on the assumption that, if N separate sources agree on a mapping, then the mapping is true. N should be at least two, and preferably, more than three.

**[0069]** FIGS. 4A and 4B illustrate an example process 400 for determining a source confidence value for each source of events, according to an embodiment. While process 400 is illustrated with a certain arrangement and ordering of steps, process 400 may be implemented with fewer, more, or different steps and a different arrangement and/or ordering of steps.

**[0070]** In step 410, to evaluate the accuracy of a particular source, the application may construct a set of "true" mappings by creating a pool of mappings from the mapping pool that are based on at least N (e.g., three) sources. In an embodiment, the source to be evaluated may be excluded from this pool. In such an embodiment, step 410 may occur after decision block 420, such that a new pool of "true" mappings is created each time a source is evaluated. In either case, these selected mappings represent a "truth set," i.e., a subset of mappings which are assumed to be true.

**[0071]** In step 420, the application determines whether any sources remain to be considered. It should be understood that all sources, from which events are obtained, may be evaluated according to process 400. If at least one source remains to be considered (i.e., "YES" in step 420), the next source is selected as the current source, and process 400 proceeds to step 430. Otherwise, if no sources remain to be considered (i.e., "NO" in step 420), process 400 ends.

**[0072]** In step 430, the overlap or intersection is determined between the mappings, derived from the current source being evaluated, and the truth set of mappings. This intersection of mappings is illustrated in FIG. 4B, according to an embodiment.

**[0073]** In step 440, the application uses this intersection to determine a measure of confidence for the current source being evaluated, based on how many true mappings and/or how many untrue mappings have been derived from the current source being evaluated. In other words, a mapping, derived from the current source, that matches a mapping in the truth set (e.g., mapping in both entity and account) is assumed to be a true mapping derived from the current source. The determined measure can then be used as, or to derive, a source confidence (i.e., value of `source_confidence` in Equation 3) for the current source being evaluated. For example, the source confidence may be a ratio (e.g., percentage) of the number of "true" mappings in the overlap to the total number of mappings derived from the current source. After the source confidence has been determined for the current source, process 400 returns to step 420 to evaluate the next source, if any, or else end.

#### **2.3.4. Overall Confidence**

**[0074]** The overall confidence for a particular mapping may be calculated, based on each of the time confidence, statistical confidence, and source confidence for that mapping, using Equation 4 below:

$$weight_1(conf_{source} + (1 - conf_{source})conf_{stat}) + weight_2(conf_{time}) \quad [Equation \quad 4]$$

wherein  $conf_{time}$  is the calculated time confidence,  $conf_{stat}$  is the calculated statistical confidence, and  $conf_{source}$  is the calculated source confidence, wherein  $weight_1 + weight_2 = 1.0$ , and wherein, in an embodiment,  $weight_1 = 0.7$  and  $weight_2 = 0.3$ .

## 2.4. Filtering and Cleaning the Mappings

[0075] In step 340, the mappings, with their calculated confidence values, may be filtered and cleaned or corrected. In an embodiment, step 340 comprises ISP filtering and/or location cleaning.

### 2.4.1. ISP Filtering

[0076] Normally, if an entity includes an IP address that belongs to an ISP (e.g., which dynamically assigns IP addresses to different customers), there may be many contradicting mappings for that same entity. Thus, in an embodiment, the application filters mappings by calculating the ratio of the confidence value for the mapping with the maximum confidence value for a particular entity to the sum of all confidence values for all mappings for that particular entity, and then comparing this ratio to a threshold. For example, this comparison may be represented as Equation 5 below:

$$\frac{\max(confidence_{entity})}{\sum(confidence_{entity})} > 0.5 (1 - \max(confidence_{entity})) \quad [Equation \quad 5]$$

[0077] Thus, for example, if the maximum confidence value is 0.5, then that confidence value would need to represent 25% of the cumulative confidence values in order to be considered a valid mapping. If the maximum confidence value were 0.1, it would need to represent 45% of the cumulative confidence values. This thresholding filters out conflicting mappings in cases in which no single mapping clearly stands out in terms of its confidence value.

[0078] In an embodiment,  $\sum(confidence_{entity})$  may represent a sum of all distinct confidence values, rather than all confidence values, to avoid over-counting. Specifically, two or more mappings for the same entity may end up having the same confidence values. In this case, dividing by the sum of all distinct confidence values, rather than the sum of all confidence values (i.e., including distinct confidence values and identical confidence values), can make the ISP filter more conservative. Alternatively,  $\sum(confidence_{entity})$  may represent the sum of all confidence values, regardless of distinctness.

[0079] In an embodiment, if Equation 5 is not satisfied, all of the mappings associated with that

particular entity may be filtered (e.g., discarded or ignored). Otherwise, if Equation 5 is satisfied, the mapping having the maximum confidence value for the entity may be kept, while all of the other mappings associated with the entity may be filtered. Essentially, Equation 5 ensures that mappings with lower confidences need a greater percentage of the total confidence value in order to avoid being discarded, whereas mappings with higher confidences will remain unless they are competing against an unreasonable number of mappings for the same entity.

#### **2.4.2. Location Cleaning**

**[0080]** Since ranges of IP addresses are associated with certain countries, if an entity includes an IP address, this IP address can be used to determine the associated country or other geographical information. The application may use this geographical information to correct or clean mappings. For instance, an event may associate an online activity (e.g., website visit) to a contact for an account (e.g., based on a login to the website). In this case, if the contact is incorrect, the mapping will be incorrect. In addition, if the contact is employed by a firm with different branches (e.g., a multi-national company, a consulting firm, etc.), the contact may be performing activity related to one branch, while linking that activity to another branch (e.g., the branch at which the contact is employed).

**[0081]** In an embodiment, the application addresses these issues by determining what portion of events for a particular account, from a particular source, are coming from a different country. Specifically, the application may collect statistics on the account level, instead of by IP address, in order to determine whether or not the account has been correctly linked. As an example, if thousands of IP addresses have been previously mapped to an account for "ABC, U.S.A." in the United States, it may be presumed that the majority of new IP addresses, received in new events, should also originate in the United States.

**[0082]** In an embodiment, the application can correct, clean, or filter mappings as follows:

1. (1) If the IP addresses in the majority of new events are associated with a different country than the country in an existing mapping, change the country in the existing mapping to the new, different country;
2. (2) If the account in a mapping represents a large company with multi-national offices and the IP addresses in events are associated with a plurality of different countries, with no single country have a clear majority, split the mapping into a plurality of mappings, with one mapping for each of the plurality of countries or for each of the main countries (e.g., and discard mappings for countries with a relatively trivial number of events); and/or
3. (3) If the IP addresses in new events are associated with a different country than the country in an existing mapping, but the source and/or mapping confidence value is low, do not change the existing mapping and do not create a new mapping for the new events (i.e., discard, ignore, or otherwise filter the new events).

### **2.4.3. Valid Domain Checking**

**[0083]** In an embodiment, one of the key fields used to identify an account is the domain. Due to the noise in the event data, some mappings may map an invalid and/or faulty domain to an account. Thus, in an embodiment, the application may validate one or more domains in the mappings by sending a request to each domain. For example, the application may validate the domain name in each mapping associated with a low confidence value (e.g., a confidence value below a predetermined threshold value) by sending a request to each domain name in the low-confidence mappings. If the request to a domain associated with a particular mapping is redirected to a site that is not indicative of a valid account (e.g., a domain seller, hosting service, social media platform, etc.) and/or the request results in an error or some other reason to believe that the mapping is likely faulty, the mapping may be discarded or excluded as invalid.

### **2.5. Selection of Mapping**

**[0084]** In step 350, for each entity, the application selects, from the mappings that were filtered and cleaned in step 340, the remaining mapping with the highest confidence value determined in step 330. In an embodiment, the application may select a mapping only if its confidence value exceeds a minimum threshold. For example, in some cases, the application may discard all of the mappings for a particular entity, including the mapping with the highest confidence value, if none of the mappings have a confidence value above a certain minimum threshold value.

**[0085]** In an embodiment, the minimum threshold may depend on the particular context in which the mappings are to be used. For example, some contexts or users of platform 110 may require a higher level of confidence for the mappings, in which case the minimum threshold can be set higher. Other contexts or users may not require such a high level of confidence, in which case the minimum threshold can be set lower or eliminated altogether.

**[0086]** In an embodiment, the output of process 300 is a graph, list, table, or other representation of each remaining mapping between entities and accounts. It should be understood that this representation will only include the mappings that remain after the filtering and cleaning in step 340 and the selection and/or thresholding in step 350. The representation may be stored and/or transmitted as a data structure, displayed in a graphical user interface, used as an input into another application (e.g., via an API), and/or the like. In an embodiment, the representation is a database of the final mappings that is indexed by entity. The entity index may be a primary, unique index, such that a single entity never maps to more than one account.

**[0087]** Furthermore, in an embodiment, each of the final mappings may be linked, by the account represented in the mapping, to contact information for the account. For example, the account identifier in the mapping may be used (e.g., as an index to a database) to retrieve contact information associated with that account identifier.

**[0088]** Thus, for example, a user may witness online activity, at the user's website, from a particular IP address. The user may initiate a query of the final mappings, through the application, using the IP address as search criteria, and the application may responsively return an account name (e.g., company name) and the associated contact information for that account to the user. Advantageously, this enables the user to contact an Internet visitor, who would otherwise have been uncontactable due to the inherent anonymity of the Internet.

**[0089]** This lookup process could also be automated. For example, a website operator could develop third-party software that utilizes an API of the application to automatically query the final mappings for each IP address seen in online activity at the website (e.g., in real time, periodically, etc.). The application may then automatically return the account names and/or contact information associated with those IP addresses for further processing by the third-party software and/or use by the website operator.

## **2.6. Example Implementation**

**[0090]** FIG. 5 illustrates the interactions between various data structures in one example implementation of process 300, according to an embodiment. It should be understood that this process may be implemented by the disclosed application (e.g., server application 112 and/or client application 132). While the process is illustrated with certain data structures and steps, the process may be implemented with fewer, more, or different data structures and/or different steps.

**[0091]** Events collected in step 310 are initially stored in event pool buffer 555, prior to being stored in event pool 560. As events are added to event pool buffer 555, they may each be given a status that indicates they are new. In addition, the events may be normalized as discussed elsewhere herein.

**[0092]** In step 505, the events in event pool buffer 555 are aggregated into mappings, as described elsewhere herein. These mappings are initially stored in mini mapping pool 565, to facilitate a smoother merge in step 520. In addition, in step 510, the unique entities from the events in event pool buffer 555 are stored in changed entity buffer 570. Changed entity buffer 570 may be partitioned by the date of the events. As the events in event pool buffer are aggregated into mappings, their statuses may be changed to indicate that they are being aggregated and/or the time that they have been aggregated (e.g., to prevent redundant processing).

**[0093]** In step 515, the mappings stored in existing mapping pool 575 are split on the entities stored in changed entity buffer 570 to produce a split mapping pool 580. This may involve sorting mapping pool 575 based on the entities in the mappings, or extracting the mappings for only those entities stored in changed entity buffer 570.

**[0094]** In step 520, mini mapping pool 565 is merged with split mapping pool 580 to produce changed mapping pool 585. This merge may comprise, for each mapping that is present in both mini mapping pool 565 and split mapping pool 580, calculating the combined statistic values and new confidence value, as described elsewhere herein, and adding a new entry for the mapping, with the combined statistic values and confidence value, to changed mapping pool 585. To facilitate the calculations of the statistic and confidence values, the pools may be split on both account (e.g., company name) and domain. Mappings that are only present in mini mapping pool 565 may simply be added to changed mapping pool 585 with their respective statistic and confidence values.

**[0095]** In step 525, mapping pool 575 is updated with the entries in changed mapping pool 585. For example, the entries in changed mapping pool 585 may overwrite any corresponding entries in mapping pool 575. In addition, in step 530, the events in event pool buffer 555, that were aggregated in step 505 (e.g., those whose statuses indicate that they were being aggregated), are emptied into event pool 560.

**[0096]** It should be understood that this implementation of process 300 may be performed periodically as new events are added to event pool buffer 555 (e.g., at specific time intervals, when event pool buffer 555 reaches a predetermined size, etc.). In addition, in an embodiment, all changes to mapping pool 575 and/or event pool 560 may be logged (e.g., by date and data) for auditing purposes and/or for restoring or rolling back mapping pool 575 and/or event pool 560 to a prior state if necessary.

## **2.7. Example Web Service**

**[0097]** As discussed elsewhere herein, platform 110 may implement a web service, for example, as part of server application 112. FIG. 6 illustrates the interactions between various data structures in one example implementation of a web service, according to an embodiment. While the process is illustrated with certain data structures and steps, the process may be implemented with fewer, more, or different data structures and/or different steps.

**[0098]** As illustrated, platform 110 may comprise a web server 680, which serves a graphical user interface, comprising a user interface 690 for interacting with a queue of inquiries 682, and a user interface 695 for submitting inquiries. Users may have access to one or both of user interfaces 690 and 695, based on their permissions. For example, an administrative user may have access to user interface 690 for managing the queue of inquiries submitted by clients, and a plurality of client users may have access to user interface 695 for submitting inquiries.

**[0099]** In step 605, a user of user interface 690 may submit a request for information regarding queue 682 (e.g., to view all inquiries in queue 682). The request is sent to web server 680 (e.g., via network(s) 120), and, in response, web server 680 queries queue 686 in step 610. In response to the query, a representation of queue 682 is returned to web server 680 in step 615. Web server 680 then parses the representation of queue 682 to generate a renderable representation of queue 682, and returns the renderable representation of queue 682 to user interface 690 for visual presentation to the user (e.g., on a display of the user's user system 130).

**[0100]** In step 625, a user of user interface 695 may submit an inquiry. The inquiry may comprise an entity, account name, and/or any other indexable field of the mappings in mapping pool 575. For example, a user who has noticed increased activity from a particular IP address may wish to tie that otherwise anonymous IP address to a specific company, so that the user can then reach out and contact that company directly (e.g., for marketing or sales purposes). Accordingly, the user may submit an inquiry, via user interface 695, for a company name that has been mapped to the otherwise anonymous IP address. The inquiry is sent to web server 680 (e.g., via network(s) 120). Web server 680 may validate the inquiry (e.g., to ensure it satisfies formatting requirements, informational requirements, etc.), and check whether or not the inquiry already exists in queue 682. If the inquiry cannot be validated, web server 680 may return an error message to user interface 695. Otherwise, if the inquiry is validated and is not already present in queue 682, web server 680 adds the inquiry to queue 682 in step 630. If the inquiry is already present in queue 682, web server 680 may discard the inquiry. Alternatively, web server 680 could add all validated inquiries to queue 682, regardless of whether or not they are redundant. It should be understood that, when web server 680 adds an inquiry to queue 682, it may parse the inquiry from user interface 695, and construct an inquiry in a new format to be added to queue 682.

**[0101]** In step 636, an inquiry from the front of queue 682 is provided to inquiry runner 684. Inquiry runner 684 may comprise a software module that converts the inquiry into a query to mapping pool 575. Thus, in step 640, inquiry runner 684 queries mapping pool 575, and receives the result of the query in step 645. In step 650, inquiry runner 684 stores the result of the query in results 686, which may comprise a local cache of platform 110. Results of queries may be stored in results 686 in JSON or any other suitable format.

**[0102]** In step 655, a user of user interface 695 may submit a request for the results of a previous inquiry (e.g., submitted in step 625). The request is sent to web server 680 (e.g., via network(s) 120), and, in response, web server 680 queries results 686 in step 660. In response to the query, the results of the inquiry (if any) are returned to web server 680 in step 665. Web server 680 then parses the results to generate a renderable representation of the results, and returns the renderable representation of the results to user interface 695 for visual presentation to the user (e.g., on a display of the user's user system 130). Web server 680 may construct the renderable representation of results using stored templates for reporting.

**[0103]** In an alternative or additional embodiment, an API may exist for usage in a similar manner as user interface 695. For example, an application may submit inquiries via the API (e.g., similar to step 625) and receive results to those inquiries in response (e.g., similar to step 670). In this manner, third-party software (e.g., executed by an external system 140) may be used to automate the inquiry process for various users of platform 110.

**[0104]** 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 as long as encompassed by the scope of the invention as defined in the appended claims. 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 defined in the appended claims.

## REFERENCES CITED IN THE DESCRIPTION

### Cited references

This list of references cited by the applicant is for the reader's convenience only. It does not form part of the European patent document. Even though great care has been taken in compiling the references, errors or omissions cannot be excluded and the EPO disclaims all liability in this regard.

### Patent documents cited in the description

- [US62609753](#) **[0001]**
- [US20130204663A](#) **[0006]**
- [US2016142272A1](#) **[0007]**
- [US2016239868A1](#) **[0008]**
- [US2017364931A1](#) **[0009]**

**Patentkrav**

**1.** Fremgangsmåde til de-anonymisering af online-besøgende eller -kontakter ved mapning af anonyme Internet Protocol-(IP)-adresser til kendte konti, hvilken fremgangsmåde omfatter anvendelse af mindst én hardware-processor af en server-plattform til:

at modtage en flerhed af begivenheder fra en flerhed af kilder, hvor hver af flerheden af begivenheder repræsenterer en online-aktivitet og omfatter en IP-adresse og begivenhedsinformationer;

at aggregere delmængder af flerheden af begivenheder til en flerhed af mapninger, hvor hver af flerheden af mapninger associerer IP-adressen, der deles af en delmængde af flerheden af begivenheder, med en konto og er associeret med en flerhed af statistikker vedrørende delmængden af begivenheder;

for hver af flerheden af mapninger, at beregne en konfidensværdi for mapningen baseret på den associerede flerhed af statistikker, hvor konfidensværdien for en bestemt mapning repræsenterer et konfidensmål for, at en besøgende eller kontakt, der har anvendt, anvender eller vil anvende den tilhørende IP-adresse i løbet af en online-aktivitet, er associeret med en tilhørende konto, hvor beregning af en konfidensværdi for hver mapning baseret på den associerede flerhed af statistikker omfatter

beregning af en tidskonfidensværdi, der angiver et tidsinterval for begivenheder, af hvilke mapningen blev afledt,

beregning af en statistisk konfidensværdi, der angiver en frekvens for de begivenheder, af hvilke mapningen blev afledt,

beregning af en kildekonfidensværdi, der angiver en nøjagtighed af én eller flere af flerheden af kilder, fra hvilke de begivenheder, af hvilke mapningen blev afledt, blev modtaget, og

beregning af konfidensværdien baseret på tidskonfidensværdien, den statistiske konfidensværdi og kildekonfidensværdien;

at vælge en endelig delmængde af flerheden af mapninger baseret på

konfidensværdierne for flerheden af mapninger, hvor der kun vælges en mapning, hvis konfidensværdien overstiger en forudbestemt minimumstærskelværdi; og at svare på én eller flere anmodninger, der omfatter en anmodet IP-adresse, ved i den endelige delmængde af mapninger at søge efter den anmodede IP-adresse og returnere en angivelse af en konto, der er associeret med den anmodede IP-adresse i den endelige delmængde af mapninger, og de associerede kontaktinformationer for kontoen, hvilket gør det muligt at kontakte den besøgende eller kontakten.

**2.** Fremgangsmåde ifølge krav 1, der endvidere omfatter anvendelse af den mindst ene hardware-processor til at normalisere begivenhedsinformationerne over flerheden af begivenheder.

**3.** Fremgangsmåde ifølge krav 1, hvor begivenhedsinformationerne for hver af flerheden af begivenheder omfatter et domænenavn.

**4.** Fremgangsmåde ifølge krav 1, hvor begivenhedsinformationerne, for hver af flerheden af begivenheder, omfatter et geografisk område, der er associeret med IP-adressen for begivenheden.

**5.** Fremgangsmåde ifølge krav 1, hvor hver konto omfatter et firmanavn.

**6.** Fremgangsmåde ifølge krav 1, hvor valg af en endelig delmængde af flerheden af mapninger baseret på konfidensværdierne for flerheden af mapninger, for mindst én helmængde af to eller flere mapninger, der hver især associerer en samme IP-adresse med forskellige konti, omfatter:

bestemmelse af én af de to eller flere mapninger med en maksimal konfidensværdi blandt de to eller flere mapninger;

beregning af en sum af konfidensværdierne for de to eller flere mapninger;

beregning af et forhold mellem den maksimale konfidensværdi og summen af konfidensværdierne; og

udelukkelse, når forholdet ikke overstiger en tærskelværdi, af de to eller flere mapninger fra den endelige delmængde af mapninger.

**7.** Fremgangsmåde ifølge krav 6, hvor tærskelværdien omfatter en konstant værdi multipliceret med forskellen mellem én og den maksimale konfidensværdi.

**8.** Fremgangsmåde ifølge krav 1, hvor flerheden af statistikker omfatter et spænd og en forsinkelse, hvor spændet angiver et tidsrum, i hvilket de begivenheder, af hvilke mapningen blev afledt, optrådte, hvor forsinkelsen angiver et tidsrum siden en seneste begivenhed, af hvilken mapningen blev afledt, optrådte, og hvor tidskonfidensværdien beregnes som:

$$\frac{\log_3(\text{span}) - \log_3(\text{delay})}{1 + |\log_3(\text{span}) - \log_3(\text{delay})|}$$

**9.** Fremgangsmåde ifølge krav 1, hvor flerheden af statistikker omfatter en spredning og en styrke, hvor spredningen angiver et antal tidsrum, i hvilke de begivenheder, af hvilke mapningen blev afledt, optrådte, hvor styrken angiver et antal begivenheder, af hvilke mapningen blev afledt, og hvor den statistiske konfidensværdi beregnes som:

$$\text{weight}_{\text{strength}} * \log_{1000}(\text{strength}) + \text{weight}_{\text{spread}} * \log_{100}(\text{spread}),$$

hvor  $\text{weight}_{\text{strength}}$  og  $\text{weight}_{\text{spread}}$  er konstante værdier, hvis sum er én.

**10.** Fremgangsmåde ifølge krav 9, hvor  $\text{weight}_{\text{strength}}$  og  $\text{weight}_{\text{spread}}$  begge er lig med 0,5.

**11.** Fremgangsmåde ifølge krav 1, hvor kildekonfidensværdien beregnes som:

$$1 - \prod_i (1 - \text{source\_confidence}_s),$$

hvor hver  $\text{source\_confidence}_s$  repræsenterer en kildekonfidensværdi for én af den ene eller flere kilder, fra hvilke de begivenheder, af hvilke mapningen blev afledt, blev modtaget.

**12.** Fremgangsmåde ifølge krav 11, der endvidere omfatter anvendelse af den mindst ene hardware-processor til, for hver af flerheden af kilder, at beregne

kildekonfidensværdien for kilden ved:

identificering af en sandhedsmængde af mapninger, der hver især blev afledt af begivenheder fra mindst  $N$  af flerheden af kilder, hvor  $N$  er større end eller lig med to;

identificering af en overlapning mellem en kildemængde af mapninger, der blev afledt af begivenheder, som blev modtaget fra kilden, og sandhedsmængden af mapninger; og

beregning af kildekonfidensværdien for kilden baseret på den identificerede overlapning mellem kildemængden af mapninger og sandhedsmængden af mapninger.

**13.** Fremgangsmåde ifølge krav 1, hvor konfidensværdien beregnes som:

$$weight_1(conf_{source} + (1 - conf_{source})conf_{stat}) + weight_2(conf_{time}),$$

hvor  $conf_{source}$  er kildekonfidensværdien,  $conf_{stat}$  er den statistiske konfidensværdi,  $conf_{time}$  er tidskonfidensværdien, og hvor  $weight_1$  og  $weight_2$  er konstante værdier, hvis sum er én.

**14.** Fremgangsmåde ifølge krav 13, hvor  $weight_1$  er lig med 0,7, og hvor  $weight_2$  er lig med 0,3.

**15.** Fremgangsmåde ifølge krav 1, der endvidere omfatter anvendelse, efter at en ny helmængde af begivenheder er modtaget fra én eller flere kilder, af den mindst ene hardware-processor til at:

aggregere delmængder af den nye helmængde af begivenheder til en ny helmængde af mapninger;

ekstrahere en del af den endelige delmængde af mapninger, der indbefatter IP-adresser, i den nye helmængde af begivenheder;

sammenflette den nye helmængde af mapninger med den ekstraherede del af den endelige delmængde af mapninger ved, for hver mapning i den sammenflettede helmængde af mapninger, at genberegne flerheden af statistikker og konfidensværdien for mapningen; og

opdatere den endelige delmængde af mapninger med den sammenflettede helmængde af mapninger.

**16.** Fremgangsmåde ifølge krav 15, hvor hver af flerheden af begivenheder lagres i en persistent begivenheds-pool, og hvor den nye helmængde af begivenheder først lagres i den persistente begivenheds-pool, efter at den endelige delmængde af mapninger er blevet opdateret.

**17.** Fremgangsmåde ifølge krav 1, der endvidere omfatter anvendelse af den mindst ene hardware-processor til:

at generere en grafisk brugergrænseflade, der omfatter ét eller flere input til modtagelse af en brugieranmodning, der omfatter en anmodet IP-adresse;  
som reaktion på modtagelse af brugieranmodningen, at tilføje en forespørgsel, der repræsenterer anmodningen, til en kø af forespørgsler;  
efterfølgende, når forespørgslen når forrest i køen, at forespørge i den endelige delmængde af flerheden af mapninger baseret på forespørgslen; og  
efter modtagelse af et resultat af forespørgslen, at generere en grafisk brugergrænseflade, der omfatter en visuel gengivelse af resultatet af forespørgslen.

**18.** System til de-anonymisering af anonyme Internet Protocol-adresser, hvilket system omfatter:

mindst én hardware-processor; og  
ét eller flere software-moduler, der ved afvikling med den mindst ene hardware-processor får den mindst ene hardware-processor til at udføre fremgangsmåden ifølge krav 1.

**19.** Ikke-flygtigt computerlæsbart medium med instruktioner lagret derpå, hvor instruktionerne ved afvikling med en processor får processoren til at:  
udføre fremgangsmåden ifølge krav 1.

## DRAWINGS

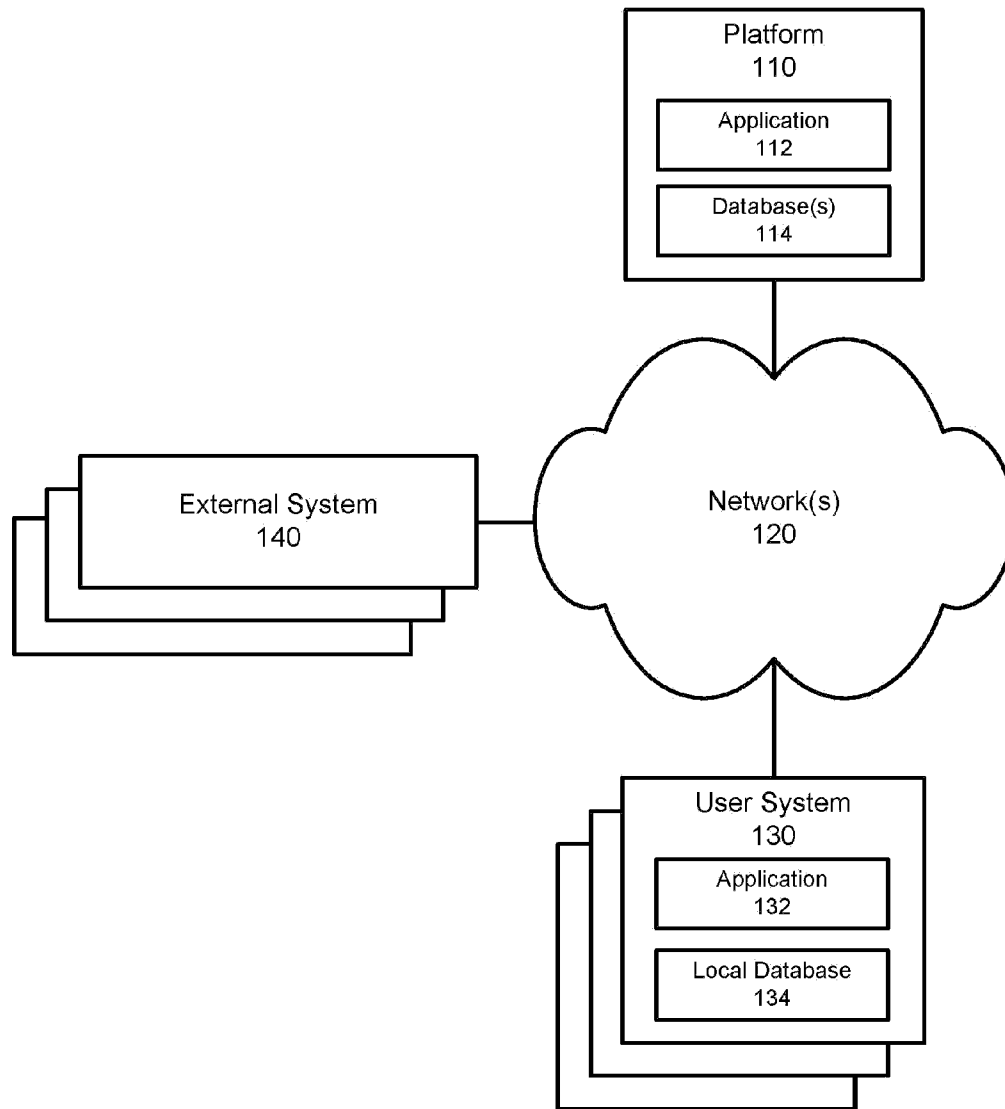


FIG. 1

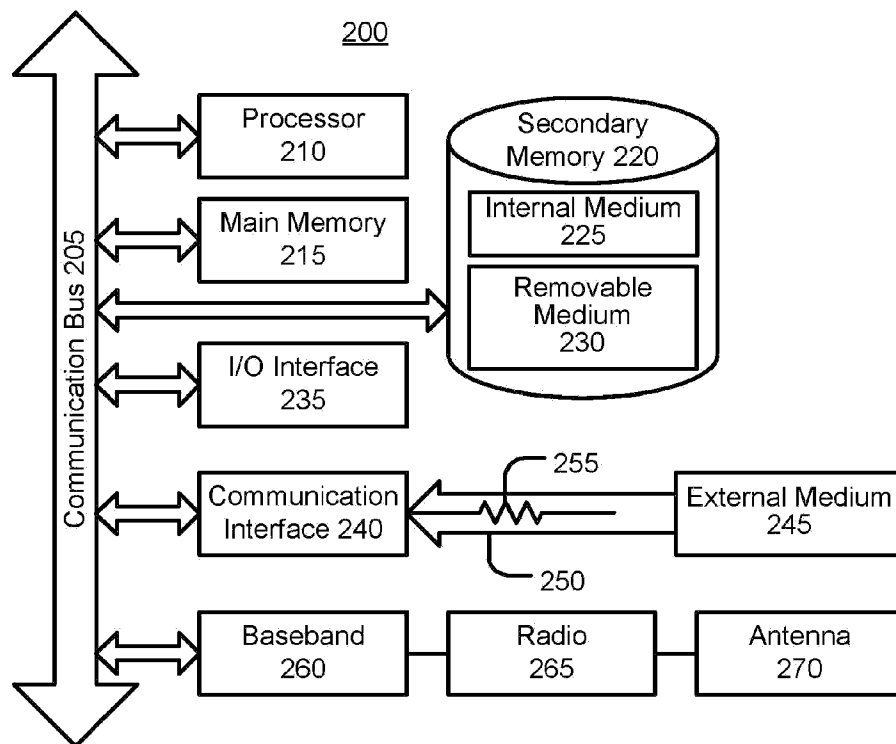


FIG. 2

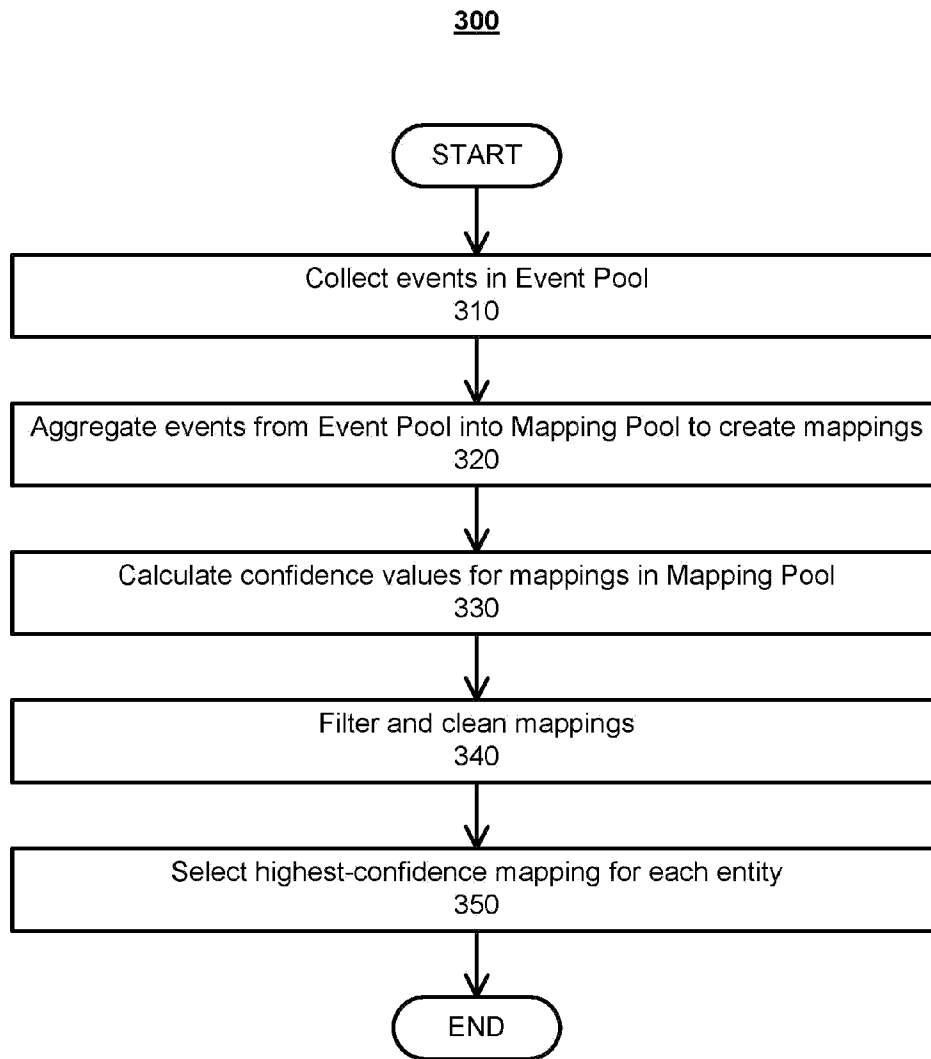
**FIG. 3**

FIG. 4A

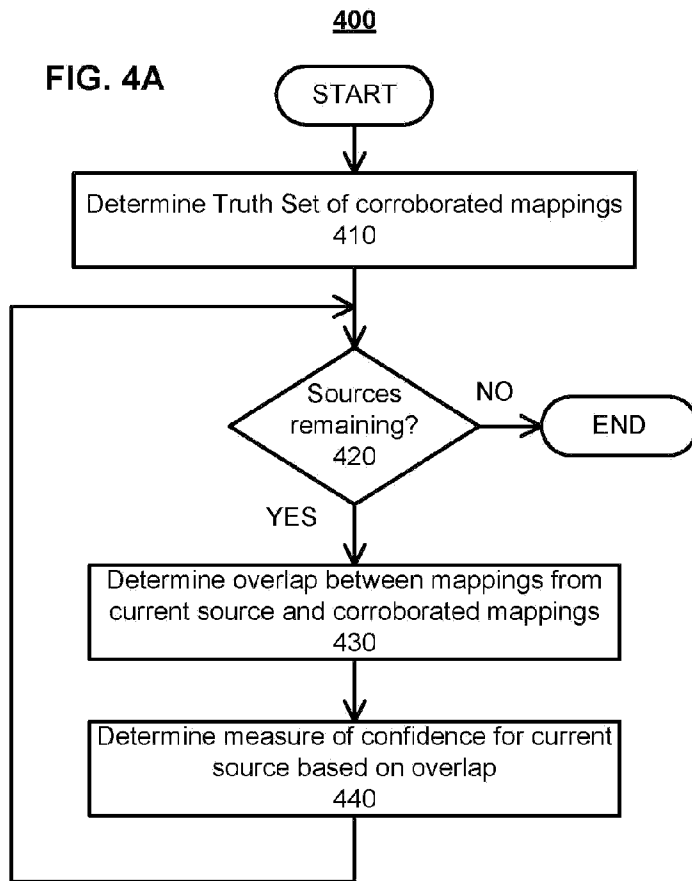
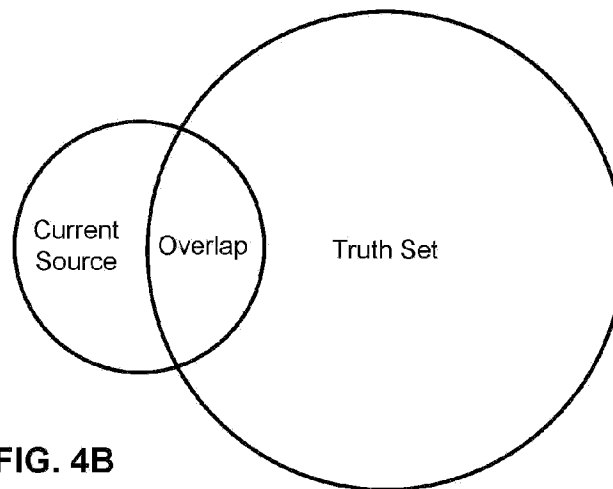


FIG. 4B



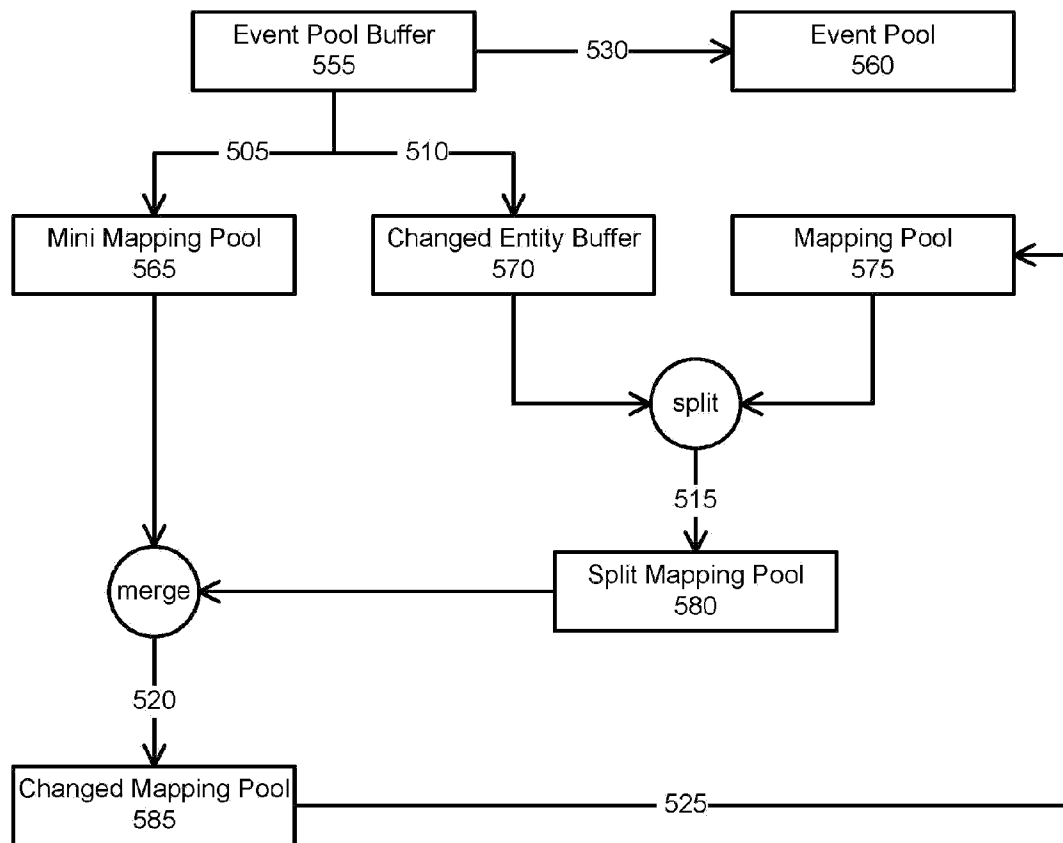


FIG. 5

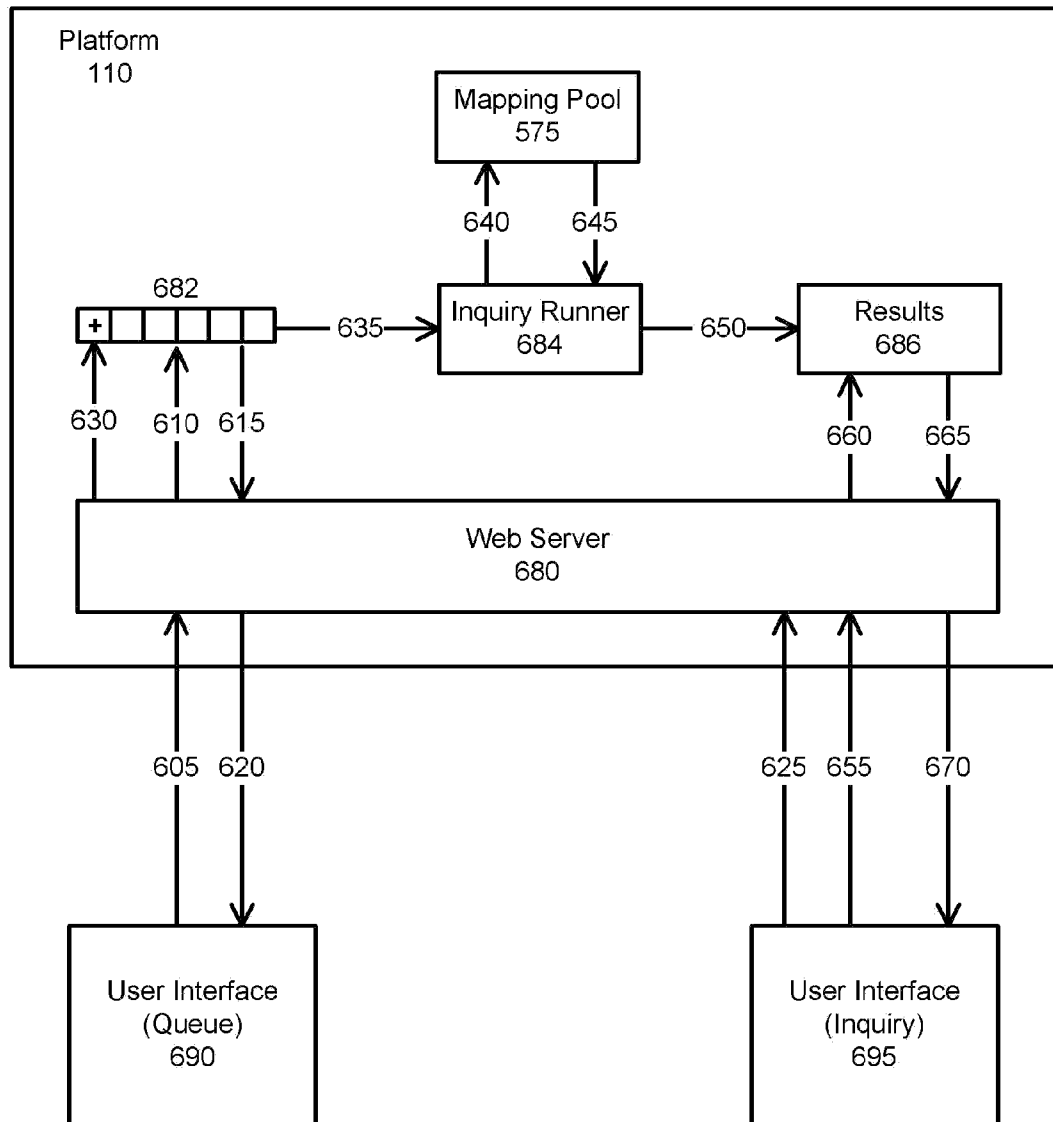


FIG. 6