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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
18 September 2014 (18.09.2014)

WIPO | PCT

(10) International Publication Number
WO 2014/145076 A2

(51) International Patent Classification: Not classified

(21) International Application Number:
PCT/US2014/029737

(22) International Filing Date:
14 March 2014 (14.03.2014)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

61/799,846	15 March 2013 (15.03.2013)	US
61/800,036	15 March 2013 (15.03.2013)	US
61/799,817	15 March 2013 (15.03.2013)	US
61/799,986	15 March 2013 (15.03.2013)	US
61/799,131	15 March 2013 (15.03.2013)	US

(72) Inventors; and
(71) Applicants : **SU, Jeffrey** [US/US]; 1714 Stoner Ave, #3, Los Andeles, CA 90025 (US). **SHIMANOVSKY, Boris** [US/US]; 2417 Bagley Avenue, Los Angeles, CA 90034 (US).

(74) Agents: **GOLDENBERG, Richard, A.** et al.; Wilmer Cutler Pickering Hale And Dorr LLP, 60 State Street, Boston, MA 02109 (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, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, 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, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, 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, 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).

Published:

— *without international search report and to be republished upon receipt of that report (Rule 48.2(g))*

(54) **Title**: APPARATUS, SYSTEMS, AND METHODS FOR CROWDSOURCING DOMAIN SPECIFIC INTELLIGENCE

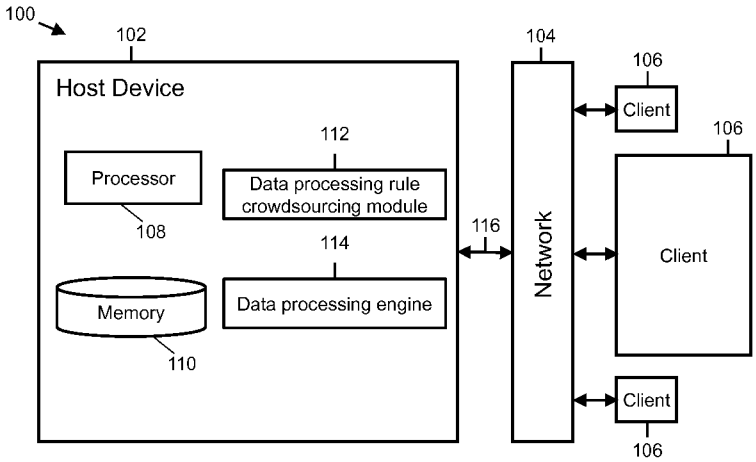


FIG. 1

(57) **Abstract**: The present disclosure provides apparatus, systems, and methods for crowdsourcing domain specific intelligence. The disclosed crowdsourcing mechanism can receive domain specific intelligence as a data processing rule module. For example, a data analytics system can request a crowd of software developers to provide a data processing rule module tailored to process a particular type of information from a particular domain. When the data analytics system receives the data processing rule module from one of the software developers for the particular domain, the data analytics system can use the received data processing rule module to process information associated with the particular domain.

WO 2014/145076 A2

APPARATUS, SYSTEMS, AND METHODS FOR CROWDSOURCING DOMAIN SPECIFIC INTELLIGENCE

Cross reference to related applications

[0001] This application claims benefit of the earlier filing date, under 35 U.S.C. §119(e), of:

- U.S. Provisional Application No. 61/799,986, filed on March 15, 2013, entitled "SYSTEM FOR ANALYZING AND USING LOCATION BASED BEHAVIOR";
- U.S. Provisional Application No. 61/800,036, filed on March 15, 2013, entitled "GEOGRAPHIC LOCATION DESCRIPTOR AND LINKER";
- U.S. Provisional Application No. 61/799,131, filed on March 15, 2013, entitled "SYSTEM AND METHOD FOR CROWD SOURCING DOMAIN SPECIFIC INTELLIGENCE";
- U.S. Provisional Application No. 61/799,846, filed March 15, 2013, entitled "SYSTEM WITH BATCH AND REAL TIME DATA PROCESSING"; and
- U.S. Provisional Application No. 61/799,817, filed on March 15, 2013, entitled "SYSTEM FOR ASSIGNING SCORES TO LOCATION ENTITIES".

[0002] This application is also related to:

- U.S. Patent Application No. 14/214,208, entitled "APPARATUS, SYSTEMS, AND METHODS FOR ANALYZING MOVEMENTS OF TARGET ENTITIES," filed on the even-date herewith;
- U.S. Patent Application No. 14/214,296, entitled "APPARATUS, SYSTEMS, AND METHODS FOR PROVIDING LOCATION INFORMATION," filed on the even-date herewith;
- U.S. Patent Application No. 14/214,213, entitled "APPARATUS, SYSTEMS, AND METHODS FOR CROWDSOURCING DOMAIN SPECIFIC INTELLIGENCE," filed on the even-date herewith;
- U.S. Patent Application No. 14/214,219, entitled "APPARATUS, SYSTEMS, AND METHODS FOR BATCH AND REALTIME DATA PROCESSING," filed on the even-date herewith;
- U.S. Patent Application No. 14/214,309, entitled "APPARATUS, SYSTEMS, AND METHODS FOR ANALYZING CHARACTERISTICS OF ENTITIES OF INTEREST," filed on the even-date herewith; and

- U.S. Patent Application No. 14/214,231, entitled "APPARATUS, SYSTEMS, AND METHODS FOR GROUPING DATA RECORDS," filed on the even-date herewith.

[0003] The entire content of each of the above-referenced applications (including both the provisional applications and the non-provisional applications) is herein incorporated by reference.

Field of the Invention

[0004] The present disclosure generally relates to systems and methods for crowdsourcing domain specific intelligence.

Background

[0005] A large amount of information is created every day. Social networking sites and blogging sites receive millions of new postings every day, and new webpages are constantly being created to provide information about a person, a landmark, a business, or any other entities that people are interested in. Furthermore, the information is usually not available from a single repository, but is usually distributed across millions of repositories, often located around the world.

[0006] Because of the sheer volume and the distributed nature of information, it is difficult for people to consume information efficiently. To address this issue, data analytics systems can (1) gather the information using a crawler and (2) create a meaningful summary of the information so that the information can be consumed easily.

[0007] To create such a meaningful summary, the data analytics system often pre-processes (or cleans) the information to detect (e.g. find or anchor) and retrieve (e.g., extract) relevant data from the gathered information. To this end, the data analytics system can use a data processing module to search for data having known formats or structures. Unfortunately, data in certain domains can be formatted or structured in a non-conventional manner. Therefore, the data processing module has to be tailored to the particular domain using domain specific intelligence so that the data processing module can detect relevant data from the large amount of information.

[0008] Unfortunately, a single software programmer may not have the domain specific intelligence nor the capacity to adequately tailor the data processing module to all domains of interest. Therefore, there is a need for an effective mechanism for providing domain specific intelligence to the data processing module.

Summary

[0009] In general, in an aspect, embodiments of the disclosed subject matter can include an apparatus. The apparatus is configured to crowdsource domain specific intelligence from a plurality of persons. The apparatus can include one or more interfaces configured to provide communication with a first plurality of computing devices and a second plurality of computing devices, wherein one of the first plurality of computing devices is operated by one of the plurality of persons having knowledge of a particular domain. The apparatus can also include a processor, in communication with the one or more interfaces, and configured to run one or more modules. The one or more module are operable to cause the apparatus to receive a plurality of data processing rule (DPR) modules from the first plurality of computing devices, wherein one of the plurality of DPR modules is tailored for use in a particular domain, and the one of the plurality of DPR modules is provided by one of the plurality of persons based on the knowledge of the particular domain; and group the plurality of DPR modules into a first DPR module package to provide the knowledge of the particular domain as a package.

[0010] In general, in an aspect, embodiments of the disclosed subject matter can include a method for crowdsourcing domain specific intelligence from a plurality of persons. The method can include providing, by one or more interfaces in an apparatus, communication with a first plurality of computing devices and a second plurality of computing devices, wherein one of the first plurality of computing devices is configured to be operated by one of the plurality of persons having knowledge of a particular domain; receiving, at a data processing rule crowdsourcing (DPRC) module in the apparatus, a plurality of data processing rule (DPR) modules from the first plurality of computing devices, wherein one of the plurality of DPR modules is tailored for use in a particular domain, and one of the plurality of DPR modules is provided by one of the plurality of persons based on the knowledge of the particular domain; an grouping the plurality of DPR modules into a first DPR module package to provide the knowledge of the particular domain as a package.

[0011] In general, in an aspect, embodiments of the disclosed subject matter can include a non-transitory computer readable medium. The non-transitory computer readable medium can include executable instructions operable to cause a data processing apparatus to provide, by one or more interfaces in the apparatus, communication with a first plurality of computing devices and a second plurality of computing devices, wherein one of the first plurality of computing devices is configured to be operated by one of the plurality of persons having knowledge of a particular domain; receive, at a data processing rule crowdsourcing (DPRC)

module in the apparatus, a plurality of data processing rule (DPR) modules from the first plurality of computing devices, wherein one of the plurality of DPR modules is tailored for use in a particular domain, and one of the plurality of DPR modules is provided by one of the plurality of persons based on the knowledge of the particular domain; and group the plurality of DPR modules into a first DPR module package to provide the knowledge of the particular domain as a package.

[0012] In any one of the embodiments disclosed herein, the apparatus, the method, or the non-transitory computer readable medium can include modules, steps, or executable instructions for sending a DPR module request, to the second plurality of computing devices, requesting the second plurality of computing devices to provide a DPR module for a predetermined domain, wherein the DPR module request includes information indicative of functional requirements of the requested DPR module.

[0013] In any one of the embodiments disclosed herein, the apparatus, the method, or the non-transitory computer readable medium can include modules, steps, or executable instructions for receiving the requested DPR module from one of the second plurality of computing devices and to determine that the received DPR module satisfies the functional requirements.

[0014] In any one of the embodiments disclosed herein, the apparatus, the method, or the non-transitory computer readable medium can include modules, steps, or executable instructions for receiving the requested DPR module from one of the second plurality of computing devices, wherein the one of the second plurality of computing devices is configured to determine that the DPR module received by the apparatus satisfies the functional requirements.

[0015] In any one of the embodiments disclosed herein, the plurality of DPR modules is configured to operate on a virtual machine.

[0016] In any one of the embodiments disclosed herein, the plurality of DPR modules is configured to operate on a system capable of running machine code compiled from two or more languages.

[0017] In any one of the embodiments disclosed herein, the apparatus, the method, or the non-transitory computer readable medium can include modules, steps, or executable instructions for sending the first DPR module package to a server in communication with the apparatus for use at the server.

[0018] In any one of the embodiments disclosed herein, one of the plurality of DPR modules is configured to call a DPR module in a second DPR module package, and the apparatus, the method, or the non-transitory computer readable medium can further include modules, steps, or executable instructions for maintaining a dependency between the first DPR module package and the second DPR module package.

[0019] In any one of the embodiments disclosed herein, the apparatus, the method, or the non-transitory computer readable medium can include modules, steps, or executable instructions for sending, in addition to the first DPR module package, the second DPR module package to the server.

[0020] In any one of the embodiments disclosed herein, the apparatus, the method, or the non-transitory computer readable medium can include modules, steps, or executable instructions for maintaining a resource, and one of the plurality of DPR modules is configured to use the resource to provide a context-aware functionality.

[0021] In any one of the embodiments disclosed herein, the apparatus, the method, or the non-transitory computer readable medium can include modules, steps, or executable instructions for providing an application programming interface (API) to enable an external system to use one of the plurality of DPR modules maintained by the apparatus.

Description of the Figures

[0022] Various objects, features, and advantages of the present disclosure can be more fully appreciated with reference to the following detailed description when considered in connection with the following drawings, in which like reference numerals identify like elements. The following drawings are for the purpose of illustration only and are not intended to be limiting of the disclosed subject matter, the scope of which is set forth in the claims that follow.

[0023] FIG. 1 illustrates a data analytics system in accordance with some embodiments.

[0024] FIG. 2 illustrates a process for gathering data processing rule (DPR) modules in accordance with some embodiments.

[0025] FIG. 3 illustrates an exemplary DPR module in accordance with some embodiments.

[0026] FIG. 4 illustrates a tree structured dependency of packages in accordance with some embodiments.

[0027] FIG. 5 illustrates a relationship between components of the DP engine in accordance with some embodiments.

[0028] FIG. 6 illustrates a process for instantiating a universe to call a DPR module in a package in accordance with some embodiments.

[0029] Description of the Disclosed Subject Matter

[0030] To process information from a particular domain, a data analytics system may use intelligence specific to that particular domain. For example, a data analytics system may receive a web page that includes phone numbers formatted in accordance with the Italian standard. According to the Italian standard, all landline phone numbers begin with a "4", whereas all mobile phone numbers begin with a "3". Unless the data analytics system is aware of such domain specific intelligence, the data analytics system may not be able to adequately process the Italian phone numbers to determine whether a phone number is a landline number or a mobile phone number.

[0031] In some cases, such domain specific intelligence can be provided to the data analytics system as a data processing rule module. The data processing rule module can include instructions that are operable detect information having a predetermined format.

[0032] In some cases, the data processing rule module can be provided by a single person. However, when there are many domains from which the information can be received, a single person may not be able to build data processing rule modules for all domains of interest. Even if the person could learn all domain-specific rules and build data processing rule modules for all domains of interest, this may not be the most efficient use of the person's time.

[0033] The present disclosure provides apparatus, systems, and methods for crowdsourcing domain specific intelligence. Because the data analytics system can receive the domain specific intelligence as a data processing rule module, the data analytics system can request a crowd of software developers or other individuals capable of learning a domain specific language that can express simplified rules for expressing domain specific knowledge to provide a data processing rule module tailored to process a particular type of information from a particular domain. When the data analytics system receives the data processing rule module from one of the software developers for the particular domain, the data analytics system can use the received data processing rule module to process information known to be associated with the particular domain. The disclosed crowdsourcing mechanism can facilitate a collaboration of software developers from a variety of domains by providing, to software

developers, various pieces of a large problem. The disclosed crowdsourcing mechanism can be used within a single organization by requesting software developers of the same organization to provide domain-specific data processing rule modules.

[0034] When referring to the domain specific intelligence, a domain can refer to an area of knowledge or activity. For example, a domain can include a geographical area (e.g., Europe), a field of expertise (e.g., computer science, law), an application with distinct data types (e.g., Italian phone number system), information about video games, subjects that tend to have topic specific slang or dialects, or any area of knowledge of activity from which data can be gathered.

[0035] In some embodiments, the data processing rule module can be configured to identify domain specific information that is formatted in accordance with a particular domain. For example, in Italy, a telephone number is represented by 6, 7, or 8 consecutive numbers (e.g., XXXXXXXX, where X indicates a digit), whereas in the US, a telephone number is represented with three digits followed by four digits (e.g., XXX-XXXX, where X indicates a digit). Therefore, a data processing rule module for detecting an Italian telephone number can be configured to search for 6, 7, or 8 consecutive numbers.

[0036] In some embodiments, the data processing rule module can be configured to identify domain specific information whose value has a particular meaning in a particular domain. Referring back to the Italian phone number example, the data processing rule module specific to the Italian phone number system can include a rule that, when a phone number begins with a "4", the phone number is a landline number; and that when a phone number begins with a "3", the phone number is a mobile phone number.

[0037] In some embodiments, a data processing system includes two subsystems: one or more data processing rule modules and a data processing engine. The data processing engine can be configured to receive information from a data source, such as a web page, and to detect domain specific information from the received information. To this end, once the data processing engine receives information from the data source, the data processing engine is configured to use one or more data processing rule modules to detect the domain specific information. Subsequently, the data processing engine can use the detected domain specific information to identify meaningful features of the received information.

[0038] In some embodiments, a data processing rule module can be dynamic. For example, the data processing module can be easily modified, replaced, or removed from the data processing system. In some sense, the data processing rule module can be considered to

be an expression of a data type. In contrast to the data processing rule module, the data processing engine can be static. The data processing engine can form a backbone of the data processing system, and may not be easily modified, replaced, or removed from the data processing system.

[0039] In some embodiments, the data processing rule module received from the crowd of software developers can be implemented in a language that can be operated on a virtual machine. For example, the data processing rule module can be implemented in a variety of programming languages, including one or more of Java, Lisp, Clojure, JRuby, Scala, or JavaScript languages, and can operate in a Java Virtual Machine (JVM) using JVM's interface, for example, to JRuby and Lisp functions.

[0040] In some embodiments, the data processing rule module received from the crowd of software developers can be implemented in a language that can be accommodated by a system capable of compiling different languages into the same type of machine code or have multi-language properties. For example, the data processing rule module can be implemented in a variety of programming languages that can be accommodated by a Common Language Runtime (CLR), developed by MICROSOFT CORPORATION of Redmond, WA. The CLR provides a machine environment (e.g., an operating platform) that is capable of running machine code compiled from two or more programming languages. As another example, the data processing rule module can be implemented in the python language and the C language, which may be accommodated together by cython.

[0041] The data analytics system can crowd-source data processing rule modules from a plurality of software developers using a data processing rule crowdsourcing module. The data processing rule crowdsourcing module is configured to receive or determine a specification for a data processing rule module, and send a data processing rule module request, which includes the specification, to a plurality of client devices at which a software developer is operating. When a software developer receives the data processing rule module request, the software developer can use her/his domain expertise to develop the requested data processing rule module, and provide the requested data processing rule module to the client device. Then the data processing rule module can provide the requested data processing rule module to the data processing rule crowdsourcing module, thereby completing the transaction.

[0042] In some cases, before sending the data processing rule module to the data processing rule crowdsourcing module, the client device can locally test the data processing

rule module to determine whether the received data processing rule module satisfies the specification of the requested data processing rule module. If the received data processing rule module does not satisfy the specification, then the client device can provide a warning signal, and request the software developer to provide a revised data processing rule module. In other cases, when the client device does not perform such local test, the data processing rule crowdsourcing module can be configured to determine whether the received data processing rule module satisfies the specification of the requested data processing rule module.

[0043] In some embodiments, the data processing rule crowdsourcing module can send the data processing rule module request to a plurality of client devices using a crowdsourcing platform. For example, the data processing rule crowdsourcing module can use Amazon Mechanical Turk to send the data processing rule module request to a plurality of software developers. As another example, the data processing rule crowdsourcing module can use an enterprise network to send the data processing rule module request to a plurality of software developers within the same organization.

[0044] FIG. 1 illustrates a data analytics system in accordance with some embodiments. The data analytics system 100 includes a host device 102, a communication network 104, and one or more client devices 106. The host device 102 can include a processor 108, a memory device 110, a data processing rule crowdsourcing (DPRC) module 112, a data processing (DP) engine 114, and one or more interfaces 116.

[0045] The processor 108 of the host device 102 can be implemented in hardware. The processor 108 can include an application specific integrated circuit (ASIC), programmable logic array (PLA), digital signal processor (DSP), field programmable gate array (FPGA), or any other integrated circuit. The processor 108 can also include one or more of any other applicable processors, such as a system-on-a-chip that combines one or more of a CPU, an application processor, and flash memory, or a reduced instruction set computing (RISC) processor. The memory device 110 of the processor 108 can include a computer readable medium, flash memory, a magnetic disk drive, an optical drive, a programmable read-only memory (PROM), and/or a read-only memory (ROM).

[0046] The DPRC module 112 is configured to coordinate the crowdsourcing of data processing rule (DPR) modules that are tailored to particular application domains. For example, the DPRC module 112 is configured to request one or more clients 106 to provide one or more DPR modules, and to receive, from the one or more clients 106, the requested

DPR modules. The DPRC module 112 can subsequently provide the requested DPR modules to the DP engine 114.

[0047] The DP engine 114 can be configured to receive (1) DPR modules from the DPRC module 112 and (2) information from a variety of data sources, and process the received information using the one or more DPR modules to provide a feature of the received information. The DP engine 114 can be configured to operate a virtual machine, such as a Java Virtual Machine (JVM), that can interface with the one or more DPR modules. For example, the virtual machine can interface with DPR modules implemented using one of Java, Lisp, Clojure, JRuby, and JavaScript languages. The DP engine 114 can also be configured to operate a system capable of compiling different languages into the same type of machine code or have multi-language properties. For example, the DP engine 114 can operate a Common Language Runtime, developed by MICROSOFT CORPORATION of Redmond, WA. As another example, the DP engine 114 can operate cython, which can accommodate the python language and the C language together.

[0048] In some embodiments, the DPRC module 112, the DP engine 114, and/or one or more DPR modules can be implemented in software stored in the non-transitory memory device 110, such as a non-transitory computer readable medium. The software stored in the memory device 110 can run on the processor 108 capable of executing computer instructions or computer code.

[0049] In some embodiments, one or more of the DPRC module 112, the DP engine 114, and/or one or more DPR module can be implemented in hardware using an ASIC, PLA, DSP, FPGA, or any other integrated circuit. In some embodiments, one or more of the DPRC module 112, the DP engine 114, and/or one or more DPR module can both be implemented on the same integrated circuit, such as ASIC, PLA, DSP, or FPGA, thereby forming a system on chip.

[0050] The host device 102 can include one or more interfaces 116. The one or more interfaces 116 provide a communication mechanism to communicate internal to, and external to, the host device 102. For example, the one or more interfaces 116 enable communication with clients 106 over the communication network 104. The one or more interfaces 116 can also provide an application programming interface (API) to other host devices, or computers coupled to the network 104 so that the host device 102 can receive location information, such as geo-location coordinates. The one or more interfaces 116 are implemented in hardware to

send and receive signals in a variety of mediums, such as optical, copper, and wireless, and in a number of different protocols some of which may be non-transitory.

[0051] In some embodiments, the host device 102 can reside in a data center and form a node in a cloud computing infrastructure. The host device 102 can also provide services on demand. A module hosting a client is capable of migrating from one host device to another host device seamlessly, without causing program faults or system breakdown. The host device 102 on the cloud can be managed using a management system. Although FIG. 1 represents the host device 102 as a single device, the host device 102 can include more than one device.

[0052] The client 106 can include any platforms capable of computations. Non-limiting examples can include a computer, such as a desktop computer, a mobile computer, a tablet computer, a netbook, a laptop, a server, a tablet computer, a cellular device, or any other computing devices having a processor and memory and any equipment with computation capabilities. The client 106 is configured with one or more processors that process instructions and run software that may be stored in memory. The processor also communicates with the memory and interfaces to communicate with other devices. The processor can be any applicable processor such as a system-on-a-chip that combines a CPU, an application processor, and flash memory. The client 106 can also provide a variety of user interfaces such as a keyboard, a touch screen, a trackball, a touch pad, and/or a mouse. The client 106 may also include speakers and a display device in some embodiments.

[0053] In some embodiments, the host device 102 can communicate with clients 106 directly, for example via a software application programming interface (API). In other embodiments, the host device 102 and the one or more client devices 106 can communicate via the communication network 104.

[0054] The communication network 104 can include the Internet, a cellular network, a telephone network, a computer network, a packet switching network, a line switching network, a local area network (LAN), a wide area network (WAN), a global area network, or any number of private networks currently referred to as an Intranet, and/or any other network or combination of networks that can accommodate data communication. Such networks may be implemented with any number of hardware and software components, transmission media and network protocols. Although FIG. 1 represents the network 104 as a single network, the network 104 can include multiple interconnected networks listed above.

[0055] The apparatus, systems, and methods disclosed herein are useful for crowdsourcing domain specific intelligence. As an example, a web-based system to respond to user queries may utilize DPR modules to process user queries. The development of the DPR modules may benefit from domain specific knowledge. As such, there is a need for apparatus, systems, and method for crowdsourcing domain specific knowledge.

[0056] By way of example, a system might use DPR modules for interpreting a variety of queries about information specific to a variety of countries. For example, addresses in different countries use different formats for street addresses, and as such, different DPR modules may be used for processing queries regarding addresses in different countries. Additionally, individuals in the United States, for example, may be more familiar with street address formats, phone number formats, and other conventions specific to the United States, whereas individuals in Italy, for example, may be more familiar with conventions specific to Italy. In this example, it would be beneficial for individuals with more domain specific knowledge of the United States to develop DPR modules for processing queries related to the United States, and for individuals with more domain specific knowledge of Italy to develop DPR modules for processing queries related to Italy.

[0057] Additionally, it is desirable to accumulate many such "micro" DPR modules into a larger set of knowledge to govern complex systems.

[0058] Additionally, it is desirable to be able to combine the results of distributed tasks with an existing system without updating the entire system. For example, in a DPR module database, it is desirable for individuals with knowledge about a specific domain, for example China, to be able to develop DPR modules specific to that domain, and to be able to combine those DPR modules with an existing system for responding to queries.

[0059] Accordingly, it is desirable to be able to distribute tasks between individuals with domain specific knowledge relevant to certain tasks, and then to be able to combine the results with an existing system and with the results of other distributed tasks.

[0060] In some embodiments, the disclosed apparatus, systems, and methods allow tasks to be distributed between individuals and allow for the results of the tasks to be combined with other results and systems.

[0061] In some embodiments, the disclosed apparatus, systems, and methods allow software developers having different knowledge or expertise to write various pieces of a software system in one or more programming languages. For example, DPR modules can be implemented in one or more of Java, Clojure, JRuby, Scala, and JavaScript languages. The

disclosed apparatus, systems, and methods can beneficially allow both data engineers and data labs teams to collaborate by working on various pieces of a larger problem.

[0062] One of the advantages of the disclosed framework is the ability to crowdsource domain specific intelligence. For instance, if there was a need to parse a country's phone number, a person from the country of interest can program a DPR module specific to that country. Similarly, different people can write DPR modules for parsing phone numbers (or other information) for other countries. Each person can program the DPR modules using a computer language of their preference. The programmed DPR modules can then be tested locally (e.g., each person can test their own code) without having to work within an existing rules-based system. After testing, the team could then merge the code into the code base and make the new DPR module(s) available to the rest of the company.

[0063] As an additional example, if a team wanted to build a database of landmarks of the world based off of Wikipedia pages, they might need to handle several steps. One team member might use an advanced natural language processing algorithm to determine if the page was about a landmark. This person might also need someone to write a parser to figure out name, country, city, and date built. The disclosed apparatus, systems, and methods provide ways for these steps to be done by different people using the language that they're most comfortable with or that lends itself well for the task. A team could even outsource other easier tasks.

[0064] The domain intelligence crowdsourcing mechanism can involve the DPR module gathering step, the DPR module packaging step, and the DPR module deployment step. The DPR module gathering step can include receiving, at a DPRC module 112, one or more DPR modules associated with one or more domains. The DPR module packaging step can include collecting, by the DPRC module 112, the received DPR modules and packaging the DPR modules into a package based on the functionalities associated with the DPR modules. The DPR module deployment step can involve processing data, at a DP engine 114, using DPR modules in the package.

[0065] FIG. 2 illustrates a process for gathering DPR modules in accordance with some embodiments. The process 200, however, is exemplary. The process 200 may be altered, e.g., by having steps added, removed, or rearranged.

[0066] In step 202, the client 106 is configured to receive a DPR module operable to perform a data processing functionality in a particular domain. The client 106 can receive the DPR module from a user of the client 106, such as a software developer; the client 106 can

receive the DPR module from another computing device over a communication network. In some embodiments, the client 106 can present, to the user of the client 106, the functional requirements of the DPR module. For example, the client 106 can request the user to provide a DPR module that is capable of parsing an Italian phone number and determining whether the Italian phone number is associated with a landline or a mobile device.

[0067] In step 204, once the client 106 receives the DPR module, the client 106 can optionally test the functionality of the received DPR module to determine whether the received DPR module satisfies the functional requirements. For example, the client 106 can run the DPR module on a known list of Italian phone numbers to determine whether the DPR module is capable of identifying all Italian phone numbers having a variety of formats and is capable of correctly determining whether the phone number is associated with a landline or a mobile device. If the received DPR module satisfies the functional requirements, the client 106 can be triggered to move to step 206. If the received DPR module does not satisfy the functional requirements, the client 106 can notify the user that the DPR module has error and that it should be revised.

[0068] In some embodiments, the client 106 can test the functionality of the DPR modules using a test module. In some cases, the client 106 can receive the test module from the user of the client 106. In other cases, the client 106 can receive the test module from the DPRC module 112.

[0069] In step 206, the client 106 can send the DPR module to the DPRC module 112 so that the DPR module can be packaged with other DPR modules into a DPR package.

[0070] In some embodiments, the DPRC module 112 can optionally cause the client 106 to receive the DPR module from, for example, the user of the client 106. For example, in step 208, prior to step 202, the DPRC module 112 can be configured to send a DPR module request to the client 106, requesting the client 106 to provide a DPR module. The DPR module request can include the functional requirements of the DPR module (e.g., a specification of a function to be performed by the DPR module.) For example, the DPR module request can indicate that the DPR module should be able to parse Italian phone numbers and to determine whether an Italian phone number is associated with a landline or a mobile device. The DPR module request can also indicate a list of program languages that can be used to implement the DPR module.

[0071] In some embodiments, the DPRC module 112 can be configured to send the DPR module request to a plurality of clients 106 and receive one or more DPR modules from at

least one of the plurality of clients 106. Subsequently, the DPRC module 112 can be configured to select, from the one or more DPR modules, the final DPR module for the function specified in the DPR module request. In some cases, the DPRC module 112 can be configured to select, as the final DPR module, the DPR module that was first received by the DPRC module 112. In other cases, the DPRC module 112 can be configured to select, as the final DPR module, the DPR module that has the lowest computational complexity or the lowest computation time.

[0072] FIG. 3 illustrates an exemplary DPR module in accordance with some embodiments. FIG. 3 provides a class 302 within which the DPR module 304 is provided. While the exemplary DPR module 304 is provided in JavaScript, a DPR module can be provided in a variety of programming languages including, for example,one or more of Java, Lisp, Clojure, Ruby, JRuby, Scala, or JavaScript languages.

[0073] The DPR module 304 can be provided within a class 302. The class 302 can include, in addition to the DPR module 304, a header. The header can include a class name 306 of the class 302. The class name 306 can be used to refer to functions (e.g., DPR modules) that are defined within the class 302. For example, the DPR module "cleanCity" 304 can be referred to as "City#cleanCity". The header can also include references to one or more packages 308 that are imported into the class 302. For example, the class 302 is configured to import a package (or a class) 308 called "common.Cleaners." This way, the class 302 (and any DPR modules defined in the class 302) can use functions (e.g., DPR modules) provided in the package "common.Cleaners." The package 308 is also known as "dependencies".

[0074] The exemplary DPR module "cleanCity" 304 is configured to receive a city name as an argument "value" and to canonicalize the city name into a predetermined representation. For example, the exemplary DPR module 304 is configured to convert common abbreviations of city names into a full name, for example, NY into New York, or LA into Los Angeles.

[0075] In some embodiments, the DPR module 304 can be configured to execute (e.g., call) DPR modules from other packages without knowing the underlying implementation of the DPR modules or the language(s) in which the DPR modules are written. For example, the function "$p.execute('common.Cleaners#trim', value)" is configured to call the DPR module "trim" from the package "common.Cleaners", which was imported into this class 302. The function "$p.execute('common.Cleaners#trim', value)" does not need to understand the implementation of "common.Cleaners#trim" and does not need to understand the

programming language in which the DPR module "trim" is implemented. In this case, the DPR module "trim" in the package "common.Cleaners" 308 is configured to strip out all leading and trailing whitespace in the input argument "value" of the DPR module 304.

[0076] In some embodiments, the DPR module 304 can include an embedded test module 310. The embedded test module 310 can be executed when the test is invoked and can report an error if the test is not passed.

[0077] In some embodiments, once the DPRC module 112 receives DPR modules from one or more clients 106, the DPRC module 112 is configured to group the received DPR modules into a DPR module package. In particular, the DPRC module 112 is configured to group DPR modules that are programmed in one or more languages supported by the DP engine 114. In some cases, a package can include DPR modules programmed using a single programming language. For example, the DPRC module 112 can be configured to group all DPR modules programmed in Clojure as a first package, and to group all DPR modules programmed in JavaScript as a second package.

[0078] In some embodiments, the DPRC module 112 can be configured to maintain resources. Resources can generally include files, databases of values, indexes of information, or other data. Examples of resources include: maps, lists, a list of cities in a particular country, a set of regular expressions for phone number variations, a mapping from abbreviations to full names or values of cities, a set of polygons representing postcodes, and other elements that can be referenced by the rules and program instructions. Such a resource could be used in a variety of applications. For example, the list of cities in a particular country can be used to reject city names that are not on the list. This allows a system to limit values of a city attribute to those on that list. As another example, the set of polygons representing postcodes can be used to check whether a particular location is actually inside the postcode associated with it. As another example, a map can be used to determine in which country a landmark is located based on the city's name.

[0079] In some embodiments, a resource can be utilized by DPR modules to provide context-aware functionalities. For example, a DPR module can refer to the resource to determine a physical location of a computing device on which the DPR module is operating, and the DPR module can adapt its functionality to the physical location to provide a location-aware functionality. As another example, a DPR module is configured to determine a phone number in a document. The DPR module can use the resource to determine a geographical location from which the document originated, or the language in which the document is

written. Subsequently, the DPR module can use the geographical location information or the language information to determine which one of the sub-DPR modules to use (e.g., a DPR module for extracting an Italian phone number or a US phone number) to extract phone numbers from the document.

[0080] In some embodiments, a package can have dependencies. For example, a DPR module in a first package can include a subroutine that calls a DPR module in a second package. As another example, a DPR module in a first package can include a subroutine that uses a resource.

[0081] In some embodiments, dependencies between packages and resources can be represented as a tree structure. FIG. 4 illustrates a tree structured dependency of packages and resources in accordance with some embodiments. The tree structured dependency 400 includes five elements: A 402, B 404, C 406, and D 408. Each of these elements can correspond to one of a package or a resource. This tree structured dependency represents a scenario in which the element A 402 depends on elements B 404 and C 406, and the element B 404 depends on elements C 406 and D 408. The element C 406 has been duplicated for illustration purposes, but could be represented as a single element in the tree 400 by re-wiring the dependencies between A 402 and C 406. This tree structured dependency can be later flattened by the DP engine 114 or the DPRC module 112 to determine the order in which packages are loaded to the DP engine 114 or to another package. This process is described further below.

[0082] Once the DPRC module 112 prepares one or more DPR module packages, the DPRC module 112 can provide the packages to the DP engine 114. The DP engine 114 can subsequently use the DPR modules in the one or more packages to process input data.

[0083] The DP engine 114 can include instructions that can instantiate one or more of the following software components (or classes): a universe, a package, a resource, and a DPR module. FIG. 5 illustrates a relationship between components of the DP engine in accordance with some embodiments. FIG. 5 includes a universe 502, one or more packages 504A-504C, one or more resources 506A-506B, and one or more DPR modules 508A-508C.

[0084] As described above, the packages 504 can each include one or more DPR modules 508.

[0085] The universe 502 may represent a container or an environment within which a part or all DPR modules 508 reside. Therefore, the universe 502 can include DPR modules 508 that implemented in a variety of programming languages supported by the DP engine 114. In

some embodiments, the universe 502 can provide the DP engine 114 with a directory of DPR modules 508 and/or packages 504 including DPR modules 508. From the DP engine's perspective, the universe 502 may be the only way to execute DPR modules 508. For example, the DP engine 114 may not call any DPR modules 508 in the universe 502 unless the DP engine 114 first instantiates the universe 502 in which the DPR modules 508 reside.

[0086] In some embodiments, the universe 502 may be the gatekeeper for external Application Programming Interfaces (APIs) to access or call DPR modules residing in the universe 502. In particular, an external program may be able to call a DPR module 508 in the universe 502 only by using an API that couples the external program to the DPR module 508 in the universe 502. In the exemplary embodiment of FIG. 5, the DPR module "packageA#DPR Module 1" can be called through the universe 502 by an external Java program or by a DPR module 508 residing internally in the DPR package 504 or the resource 506.

[0087] FIG. 6 illustrates a process for instantiating a universe to call a DPR module in a package in accordance with some embodiments. The process 600, however, is exemplary. The process 600 may be altered, e.g., by having steps added, removed, or rearranged.

[0088] In step, 602, the DP engine 114 can use a header to import one or more DPR module packages that include the desired DPR modules. In this case, the DP engine 114 imports (1) packageA.City so that the DP engine can use DPR modules in packageA.City and (2) packageA.State so that the DP engine can use DPR modules in packageA.State.

[0089] In some embodiments, a plurality of DPR module packages imported by the DP engine 114 can have dependencies between them. The DP engine 114 can be configured to take the dependencies into account to determine the order in which the plurality of DPR module packages is loaded onto the DP engine 114. In particular, when the dependencies are represented as a tree structure, the DP engine 114 can be configured to flatten the dependencies so that leaves of the tree structure can be loaded prior to the root of the tree structure.

[0090] For example, if the DP engine 114 is configured to import DPR module packages having the dependencies of FIG. 3, the DP engine can be configured to load the DPR module packages in the following order: D -> C -> B -> A. This loading order can ensure that all of A's dependencies are loaded onto the DP engine 114 before A.

[0091] In step 604, the DP engine 114 can instantiate a software object that is capable of accessing a package having the desired DPR module. For example, the DP engine 114 can

instantiate an object of the class Scarecrow that is capable of accessing the package "packageA.City".

[0092] In step 606, the DP engine 114 can instantiate a software object of the class Universe that is associated with the Scarecrow object from step 604. For example, the DP engine 114 can instantiate an object of the Universe class in which the package packageA.City resides.

[0093] In step 608, the DP engine 114 can call the desired DPR module through the software object of the Universe class instantiated in step 606. For example, the software object can use a call-back function to call the cleanCity module 612 in the package packageA.City. The call-back function can be used to provide an argument for the cleanCity module 612 as well.

[0094] In some embodiments, a DPR module 112 can be wrapped by a Java object to be proxied by a common interface (e.g., a common set of input arguments and output values). More particularly, the common interface can receive a request, optionally alter it, make the request to the underlying Java object that is wrapping the DPR module 112. Subsequently, the common interface can receive a response from the Java object, optionally handle the retries or exceptions, optionally alter the response, and return the response to the requester that sent the request to the common interface. This allows the DP engine 114 to use DPR modules that may be programmed in different programming languages. For example, in step 610, the DP engine 114 can be configured to run the DPR module "packageA.State#fromCity" 614 implemented in JRuby, not the JavaScript used to implement the DPR module "packageA.City#cleanCity" 612. To this end, the DPR modules can be executed (e.g., called) by the Java Virtual Machine (JVM) using one or more layers, and more particularly, the runtime data area (a layer of the JVM). For example, the runtime data area can provide a function area (e.g., method area) which is shared by multiple threads running in the JVM. This enables functions (e.g., methods) in different languages to be called by the JVM (or any other programs being executed on the JVM) since the functions are in the common area that is accessible by multiple threads.

[0095] For example, the DPR module "cleanCity" 612 written in JavaScript is called from "packageA" but abides by an interface that a DPR module takes in a String as an argument and returns a value that is casted to a String. The DPR module "fromCity" 614, also in "packageA," could have been written in another language, such as Clojure, by another user of the system. Though the two DPR modules are programmed in different languages

and have been implemented independently, the two DPR modules are able to interact. This allows the DP engine 114 to share and cast supported data types (e.g., String, Boolean, Numbers) to different language environments to allow them to share functionality.

[0096] In some embodiments, the DP engine 114 can instantiate an environment for a particular programming language, such as methods or resources, on a need basis. This is called a lazy instantiation of environments. Lazy instantiation means that objects are not created or loaded until they are used. For example, by step 608 of FIG. 6, the DP engine 114 executes only DPR modules programmed in JavaScript. Therefore, by step 608, the DP engine 114 does not need to instantiate environments for supporting other programming languages, such as Clojure and JRuby. In step 610, however, if the DPR module "packageA.State#fromCity" 614 is implemented in JRuby, the DP engine 114 can instantiate the environment for JRuby prior to executing "packageA.State#fromCity". This way, the DP engine 114 can instantiate the environments only when there is a need to instantiate the environments. As another example, until a program requests the system to process an Italian phone number, the method and resources for processing Italian phone number are not loaded onto the system, and, therefore, do not occupy space in memory.

[0097] In some embodiments, the lazy instantiation of environments can be accomplished, in part, by creating a reentrant object. A reentrant object is an object that can be safely called while the object is in the middle of processing because it either doesn't have an internal state or it properly handles states such that interruptions don't leave state inconsistent. In the example of processing an Italian phone number, the process of loading the phone number method does not replace the existing method until the loading process has completed successfully. Once the reentrant object is created, the DPR modules in the rules file can be evaluated (e.g., invoked) such that they become member functions of the reentrant object with appropriately isolated namespaces so that methods with the same name can be appropriately isolated. For example, by using appropriately isolated namespaces, the method "poi.Italy.phone_number" does not conflict with "poi.USA.phone_number".

[0098] In some embodiments, the process 600 can be incorporated into a DPR module. For example, the steps 604-610 can be nested as a DPR module, as disclosed in FIG. 3, and the DPR module can be grouped into a package, as disclosed above. When the DPR module corresponding to the process 600 is called by a DP engine 114, the DP engine 114 is configured to execute the process 600 as described above.

[0099] In some embodiments, the DPRC module 112 is configured to update a data processing rule package. In some cases, the DPRC module 112 is configured to update the package in batch, for example, when the DPRC module 112 receives a predetermined number of new or updated DPR modules. In some cases, the DPRC module 112 is configured to update the package in substantially real time, for example, when the DPRC module 112 receives a new DPR module. In some cases, the DPRC module 112 is configured to update the package periodically, for example, after a predetermined period of time. For example, a client 106 may contribute, to the DPRC module112, a new DPR module that is configured to identify Italian phone numbers. A DPRC module 112 can check for new DPR modules and/or updates to the existing DPR modules. Once the rebuild criterion is met (e.g., that a predetermined number of new or updated DPR modules has been received, that a single new or updated DPR module has been received, or that a predetermined amount of time has passed since the last update of the package), the DPRC module 112 can rebuild the DPR module package with the new or updated DPR modules. After the rebuild, any new input data received by the host device 102 can be evaluated using the new or updated DPR modules and thus can detect Italian phone numbers.

[0100] In some embodiments, the host device 102 support the use of the same or different DPR modules individually or in combination within a large multi-device batch processing pipeline and real-time server applications where the host device 102 can respond to user actions or new incremental contributions of DPR modules.

[0101] In some embodiments, the host device 102 can be configured to retrieve the latest DPR modules or, alternatively, a specific version of DPR modules, and use them to process previously-received input data. Referring to the Italian phone number example above, the previously-received input data could be re-processed with the newly added DPR modules or a specific version of the DPR modules and the host device can use the newly added DPR modules or the specific version of the DPR modules to recognize Italian phone numbers in previously examined and new web pages or user queries.

[0102] In some embodiments, the host device 102 can be configured to distribute DPR rule packages or individual DPR rules to other computing devices, including, for example, the client 106 or other servers in communication with the host device 102. For example, a server in communication with the host device 102 can request the host device 102 to provide a particular DPR module package, and the host device 102 can, in response, determine package dependencies for using the particular DPR module package. Then, the host device

102 can provide, to the requesting server, the particular DPR module package and any other DPR module packages on which the particular DPR module package depend on.

[0103] Other embodiments are within the scope and spirit of the disclosed subject matter.

[0104] The subject matter described herein can be implemented in digital electronic circuitry, or in computer software, firmware, or hardware, including the structural means disclosed in this specification and structural equivalents thereof, or in combinations of them. The subject matter described herein can be implemented as one or more computer program products, such as one or more computer programs tangibly embodied in an information carrier (e.g., in a machine-readable storage device), or embodied in a propagated signal, for execution by, or to control the operation of, data processing apparatus (e.g., a programmable processor, a computer, or multiple computers). A computer program (also known as a program, software, software application, or code) can be written in any form of programming language, including compiled or interpreted languages, and it can be deployed in any form, including as a stand-alone program or as a module, component, subroutine, or other unit suitable for use in a computing environment. A computer program does not necessarily correspond to a file. A program can be stored in a portion of a file that holds other programs or data, in a single file dedicated to the program in question, or in multiple coordinated files (e.g., files that store one or more modules, sub-programs, or portions of code). A computer program can be deployed to be executed on one computer or on multiple computers at one site or distributed across multiple sites and interconnected by a communication network.

[0105] The processes and logic flows described in this specification, including the method steps of the subject matter described herein, can be performed by one or more programmable processors executing one or more computer programs to perform functions of the subject matter described herein by operating on input data and generating output. The processes and logic flows can also be performed by, and apparatus of the subject matter described herein can be implemented as, special purpose logic circuitry, e.g., an FPGA (field programmable gate array) or an ASIC (application-specific integrated circuit).

[0106] Processors suitable for the execution of a computer program include, by way of example, both general and special purpose microprocessors, and any one or more processor of any kind of digital computer. Generally, a processor will receive instructions and data from a read-only memory or a random access memory or both. The essential elements of a computer are a processor for executing instructions and one or more memory devices for storing instructions and data. Generally, a computer will also include, or be operatively

coupled to receive data from or transfer data to, or both, one or more mass storage devices for storing data, e.g., magnetic, magneto-optical disks, or optical disks. Information carriers suitable for embodying computer program instructions and data include all forms of non-volatile memory, including by way of example semiconductor memory devices, (e.g., EPROM, EEPROM, and flash memory devices); magnetic disks, (e.g., internal hard disks or removable disks); magneto-optical disks; and optical disks (e.g., CD and DVD disks). The processor and the memory can be supplemented by, or incorporated in, special purpose logic circuitry.

[0107] To provide for interaction with a user, the subject matter described herein can be implemented on a computer having a display device, e.g., a CRT (cathode ray tube) or LCD (liquid crystal display) monitor, for displaying information to the user and a keyboard and a pointing device, (e.g., a mouse or a trackball), by which the user can provide input to the computer. Other kinds of devices can be used to provide for interaction with a user as well. For example, feedback provided to the user can be any form of sensory feedback, (e.g., visual feedback, auditory feedback, or tactile feedback), and input from the user can be received in any form, including acoustic, speech, or tactile input.

[0108] The techniques described herein can be implemented using one or more modules. As used herein, the term "module" refers to computing software, firmware, hardware, and/or various combinations thereof. At a minimum, however, modules are not to be interpreted as software that is not implemented on hardware, firmware, or recorded on a non-transitory processor readable recordable storage medium. Indeed "module" is to be interpreted to include at least some physical, non-transitory hardware such as a part of a processor or computer. Two different modules can share the same physical hardware (e.g., two different modules can use the same processor and network interface). The modules described herein can be combined, integrated, separated, and/or duplicated to support various applications. Also, a function described herein as being performed at a particular module can be performed at one or more other modules and/or by one or more other devices instead of or in addition to the function performed at the particular module. Further, the modules can be implemented across multiple devices and/or other components local or remote to one another. Additionally, the modules can be moved from one device and added to another device, and/or can be included in both devices.

[0109] The subject matter described herein can be implemented in a computing system that includes a back-end component (e.g., a data server), a middleware component (e.g., an

application server), or a front-end component (e.g., a client computer having a graphical user interface or a web browser through which a user can interact with an implementation of the subject matter described herein), or any combination of such back-end, middleware, and front-end components. The components of the system can be interconnected by any form or medium of digital data communication, e.g., a communication network. Examples of communication networks include a local area network ("LAN") and a wide area network ("WAN"), e.g., the Internet.

[0110] The terms "a" or "an," as used herein throughout the present application, can be defined as one or more than one. Also, the use of introductory phrases such as "at least one" and "one or more" should not be construed to imply that the introduction of another element by the indefinite articles "a" or "an" limits the corresponding element to only one such element. The same holds true for the use of definite articles.

[0111] It is to be understood that the disclosed subject matter is not limited in its application to the details of construction and to the arrangements of the components set forth in the following description or illustrated in the drawings. The disclosed subject matter is capable of other embodiments and of being practiced and carried out in various ways. Also, it is to be understood that the phraseology and terminology employed herein are for the purpose of description and should not be regarded as limiting.

[0112] As such, those skilled in the art will appreciate that the conception, upon which this disclosure is based, may readily be utilized as a basis for the designing of other structures, methods, and systems for carrying out the several purposes of the disclosed subject matter. It is important, therefore, that the claims be regarded as including such equivalent constructions insofar as they do not depart from the spirit and scope of the disclosed subject matter.

[0113] Although the disclosed subject matter has been described and illustrated in the foregoing exemplary embodiments, it is understood that the present disclosure has been made only by way of example, and that numerous changes in the details of implementation of the disclosed subject matter may be made without departing from the spirit and scope of the disclosed subject matter.

[0114] We claim:

CLAIMS

1. An apparatus configured to crowdsource domain specific intelligence from a plurality of persons, the apparatus comprising:

one or more interfaces configured to provide communication with a first plurality of computing devices and a second plurality of computing devices, wherein one of the first plurality of computing devices is operated by one of the plurality of persons having knowledge of a particular domain; and

a processor, in communication with the one or more interfaces, and configured to run one or more modules that are operable to cause the apparatus to:

receive a plurality of data processing rule (DPR) modules from the first plurality of computing devices, wherein one of the plurality of DPR modules is tailored for use in a particular domain, and the one of the plurality of DPR modules is provided by one of the plurality of persons based on the knowledge of the particular domain; and

group the plurality of DPR modules into a first DPR module package to provide the knowledge of the particular domain as a package.

2. The apparatus of claim 1, wherein the modules are operable to cause the apparatus to send a DPR module request to the second plurality of computing devices, requesting the second plurality of computing devices to provide a DPR module for a predetermined domain, wherein the DPR module request includes information indicative of functional requirements of the requested DPR module.

3. The apparatus of claim 2, wherein the modules are operable to cause the apparatus to receive the requested DPR module from one of the second plurality of computing devices and to determine that the received DPR module satisfies the functional requirements.

4. The apparatus of claim 2, wherein the modules are operable to cause the apparatus to receive the requested DPR module from one of the second plurality of computing devices, wherein the one of the second plurality of computing devices is configured to determine that the DPR module received by the apparatus satisfies the functional requirements.

5. The apparatus of claim 1, wherein the plurality of DPR modules is configured to operate on a virtual machine.

6. The apparatus of claim 1, wherein the plurality of DPR modules is configured to operate on a system capable of running machine code compiled from two or more languages.

7. The apparatus of claim 1, wherein the modules are operable to cause the apparatus to send the first DPR module package to a server in communication with the apparatus for use at the server.

8. The apparatus of claim 7, wherein one of the plurality of DPR modules is configured to call a DPR module in a second DPR module package, and the modules are operable to cause the apparatus to maintain a dependency between the first DPR module package and the second DPR module package.

9. The apparatus of claim 8, wherein the modules are operable to cause the apparatus to send, in addition to the first DPR module package, the second DPR module package to the server.

10. The apparatus of claim 1, wherein the modules are operable to cause the apparatus to maintain a resource, and one of the plurality of DPR modules is configured to use the resource to provide a context-aware functionality.

11. The apparatus of claim 1, wherein the modules are operable to cause the apparatus to provide an application programming interface (API) to enable an external system to use one of the plurality of DPR modules maintained by the apparatus.

12. A method of crowdsourcing domain specific intelligence from a plurality of persons, the method comprising:

providing, by one or more interfaces in an apparatus, communication with a first plurality of computing devices and a second plurality of computing devices, wherein one of the first plurality of computing devices is configured to be operated by one of the plurality of persons having knowledge of a particular domain;

receiving, at a data processing rule crowdsourcing (DPRC) module in the apparatus, a plurality of data processing rule (DPR) modules from the first plurality of computing devices, wherein one of the plurality of DPR modules is tailored for use in a particular domain, and one of the plurality of DPR modules is provided by one of the plurality of persons based on the knowledge of the particular domain; and

grouping the plurality of DPR modules into a first DPR module package to provide the knowledge of the particular domain as a package.

13. The method of claim 12, further comprising sending, by the DPRC module, a DPR module request to the second plurality of computing devices, requesting the second plurality of computing devices to provide a DPR module for a predetermined domain, wherein the DPR module request includes information indicative of functional requirements of the requested DPR module.

14. The method of claim 13, further comprising receiving, by the DPRC module, the requested DPR module from one of the second plurality of computing devices, wherein the one of the second plurality of computing devices is configured to determine that the DPR module received by the DPRC module satisfies the functional requirements.

15. The method of claim 12, wherein one of the plurality of DPR modules is configured to call a DPR module in a second DPR module package, and the method further comprises maintaining a dependency between the first DPR module package and the second DPR module package.

16. The method of claim 12, further comprising providing an application programming interface (API) to enable an external system to use one of the plurality of DPR modules maintained by the apparatus.

17. A non-transitory computer readable medium having executable instructions operable to cause a data processing apparatus to:

provide, by one or more interfaces in the apparatus, communication with a first plurality of computing devices and a second plurality of computing devices, wherein one of the first plurality of computing devices is configured to be operated by one of the plurality of persons having knowledge of a particular domain;

receive, at a data processing rule crowdsourcing (DPRC) module in the apparatus, a plurality of data processing rule (DPR) modules from the first plurality of computing devices, wherein one of the plurality of DPR modules is tailored for use in a particular domain, and one of the plurality of DPR modules is provided by one of the plurality of persons based on the knowledge of the particular domain; and

group the plurality of DPR modules into a first DPR module package to provide the knowledge of the particular domain as a package.

18. The non-transitory computer readable medium of claim 17, wherein the executable instructions are further operable to cause the data processing apparatus to send, by the DPRC module, a DPR module request to the second plurality of computing devices, requesting the second plurality of computing devices to provide a DPR module for a predetermined domain, wherein the DPR module request includes information indicative of functional requirements of the requested DPR module.

19. The non-transitory computer readable medium of claim 17, wherein the executable instructions are further operable to cause the data processing apparatus to receive, by the DPRC module, the requested DPR module from one of the second plurality of computing devices, wherein the one of the second plurality of computing devices is configured to determine that the DPR module received by the DPRC module satisfies the functional requirements.

20. The non-transitory computer readable medium of claim 17, wherein the executable instructions are further operable to cause the data processing apparatus to provide an application programming interface (API) to enable an external system to use one of the plurality of DPR modules maintained by the apparatus.

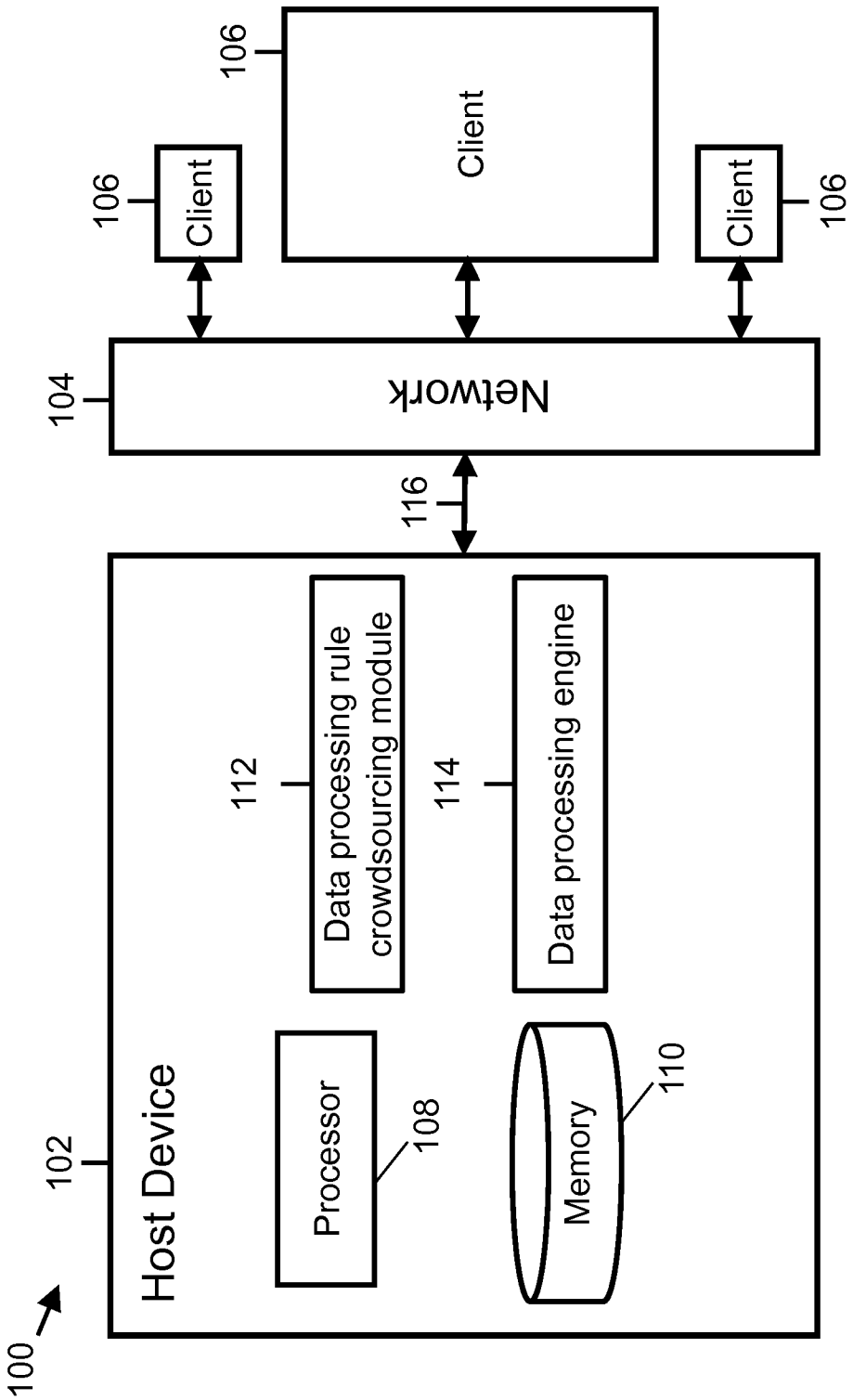

100
102
Host Device
Processor
108
Memory
110
112
Data processing rule crowdsourcing module
114
Data processing engine
116
104
Network
106
Client
106
Client
106
Client

FIG. 1

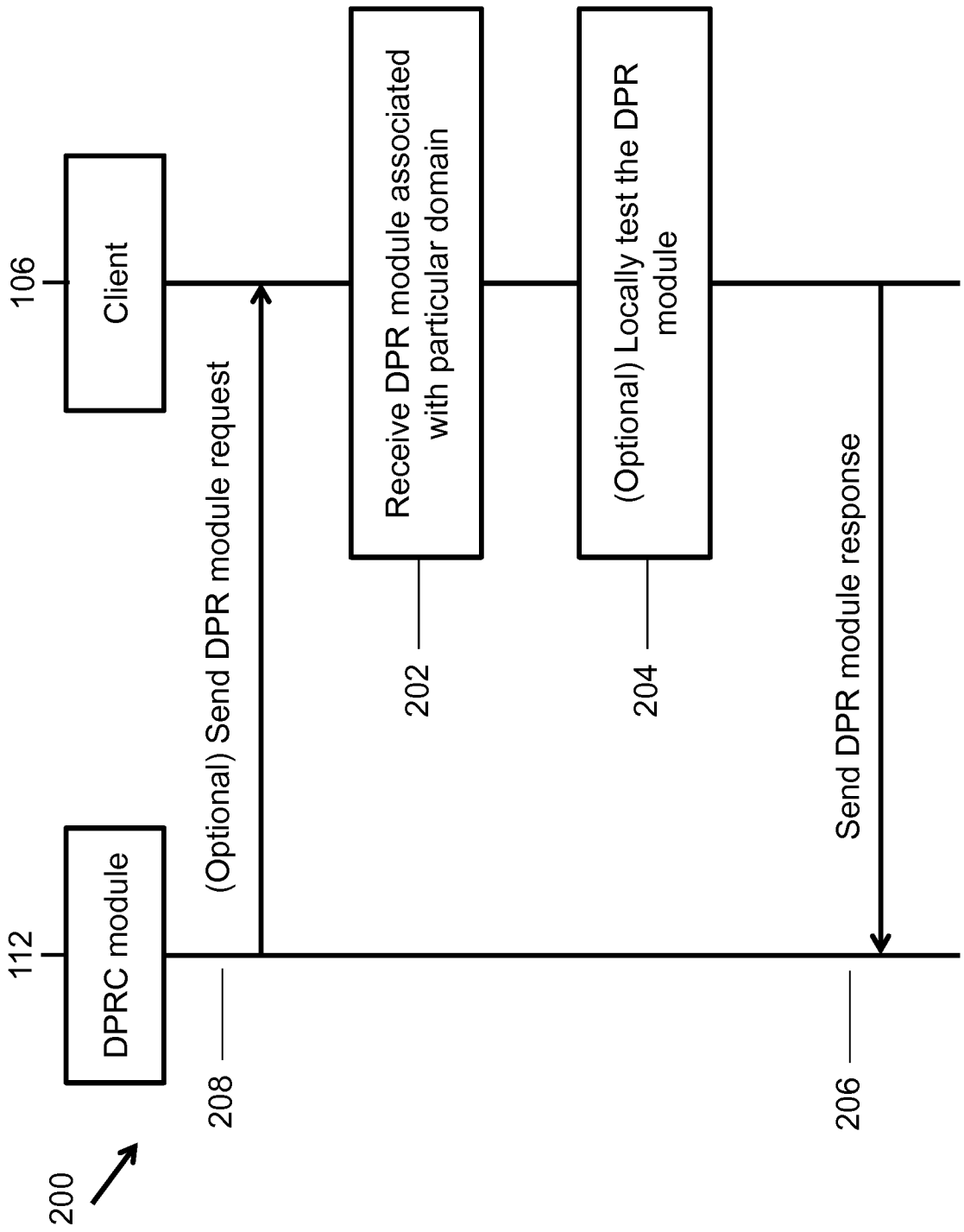
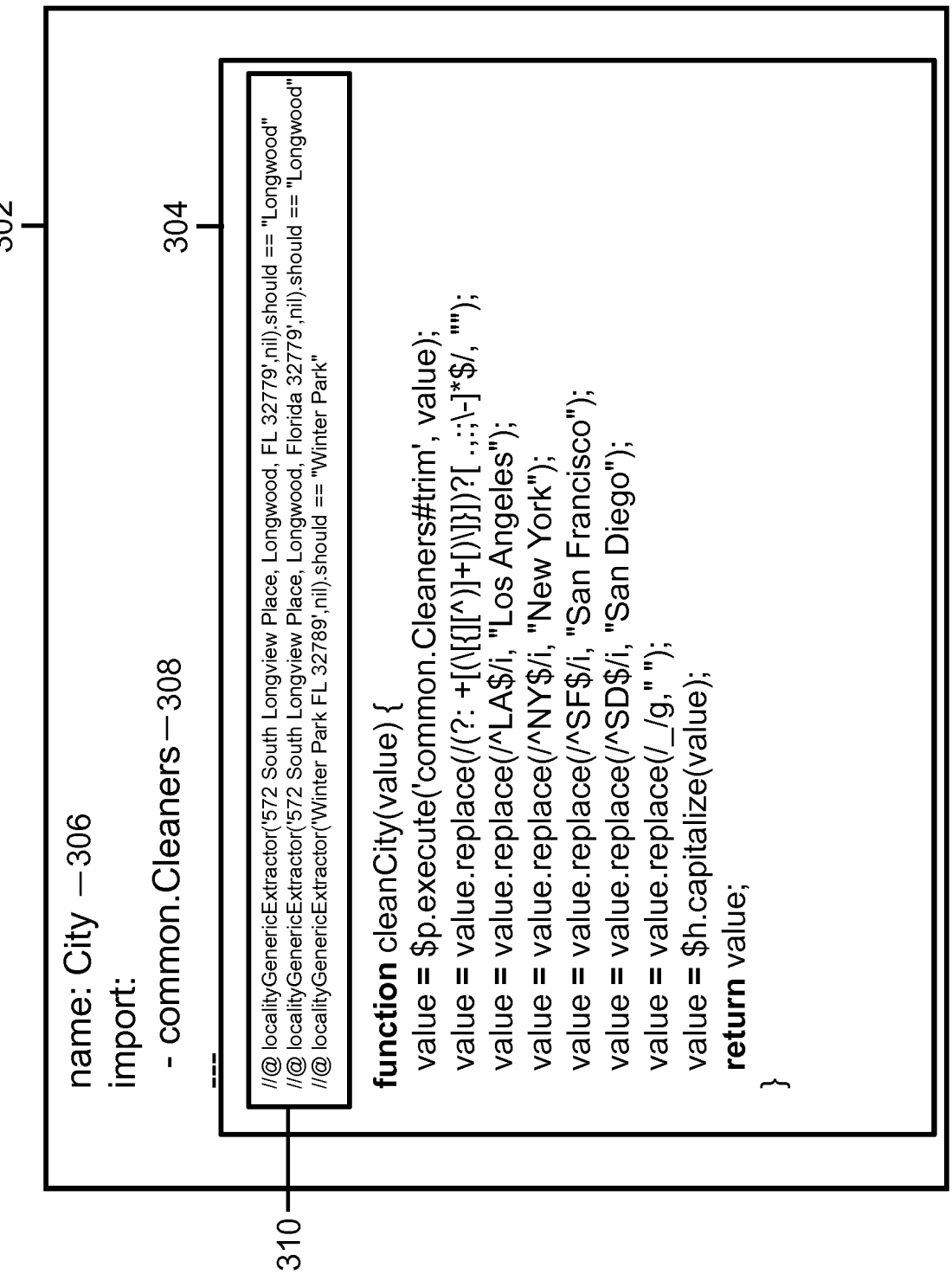

200
112
DPRC module
106
Client
(Optional) Send DPR module request
208
Receive DPR module associated with particular domain
202
(Optional) Locally test the DPR module
204
Send DPR module response
206

FIG. 2


302
name: City —306
import:
- common.Cleaners —308

304
//@ localityGenericExtractor('572 South Longview Place, Longwood, FL 32779',nil).should == "Longwood"
//@ localityGenericExtractor('572 South Longview Place, Longwood, Florida 32779',nil).should == "Longwood"
//@ localityGenericExtractor('Winter Park FL 32789',nil).should == "Winter Park"
function cleanCity(value) {
value = $p.execute('common.Cleaners#trim', value);
value = value.replace(/(?: +[(\[{][^)]+[)\]}])?[.,:;\-]*$/, "");
value = value.replace(/^LA$/i, "Los Angeles");
value = value.replace(/^NY$/i, "New York");
value = value.replace(/^SF$/i, "San Francisco");
value = value.replace(/^SD$/i, "San Diego");
value = value.replace(/_/g," ");
value = $h.capitalize(value);
return value;
}
310

FIG. 3

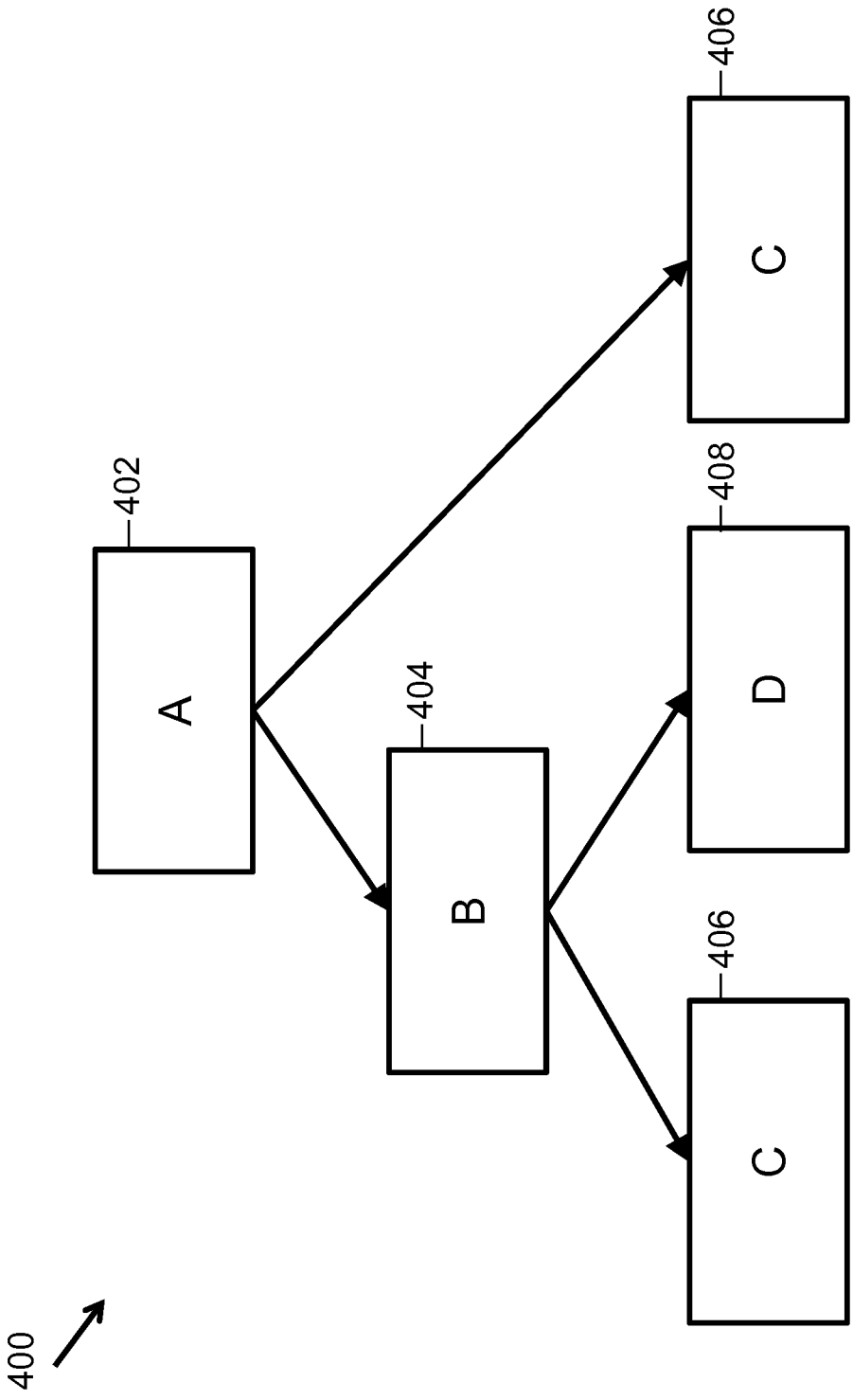


FIG. 4

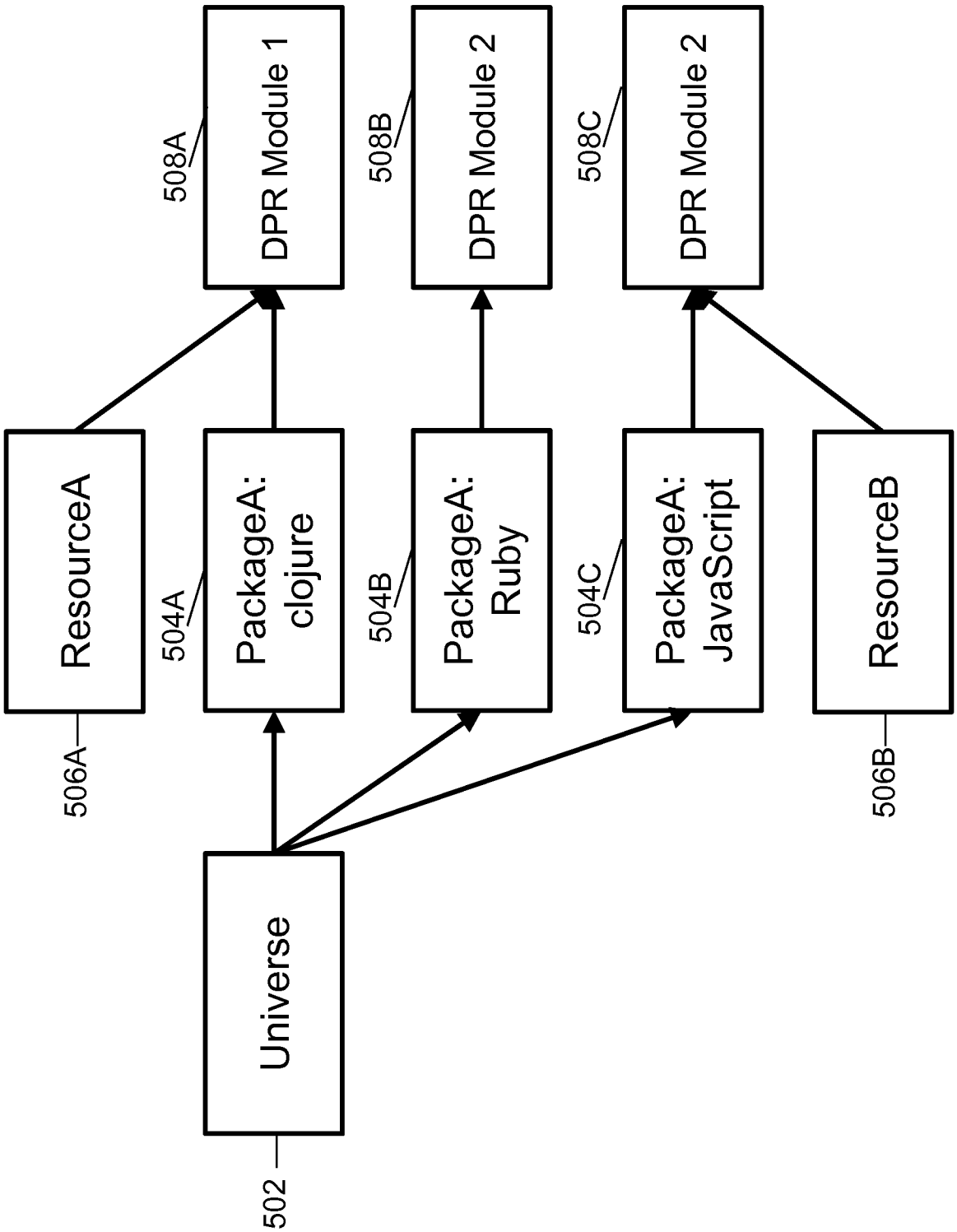
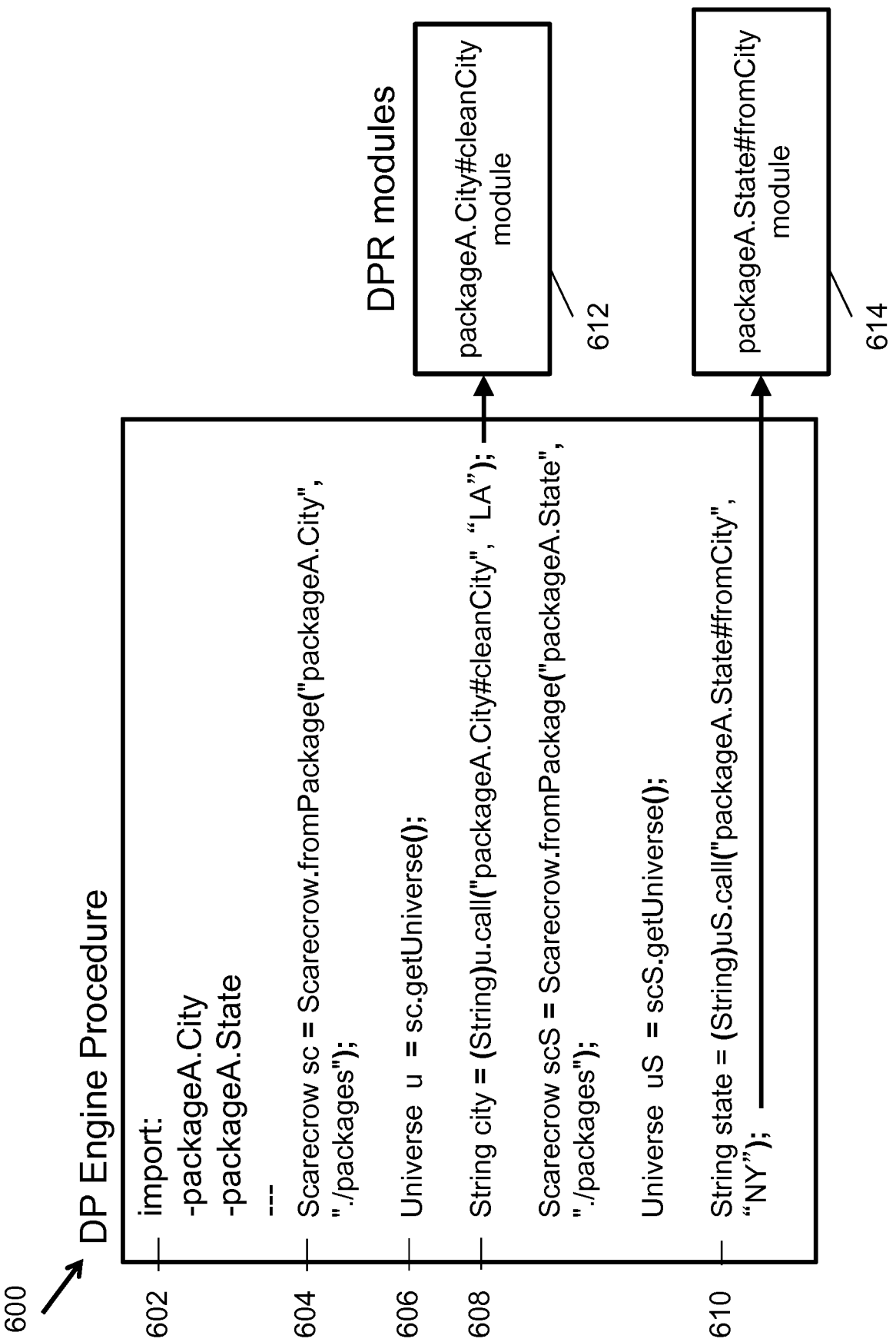

Universe
502
ResourceA
506A
PackageA: clojure
504A
PackageA: Ruby
504B
PackageA: JavaScript
504C
ResourceB
506B
DPR Module 1
508A
DPR Module 2
508B
DPR Module 2
508C

FIG. 5


600
DP Engine Procedure
602
import:
-packageA.City
-packageA.State

604
Scarecrow sc = Scarecrow.fromPackage("packageA.City", "./packages");
606
Universe u = sc.getUniverse();
608
String city = (String)u.call("packageA.City#cleanCity", "LA");
Scarecrow scS = Scarecrow.fromPackage("packageA.State", "./packages");
Universe uS = scS.getUniverse();
610
String state = (String)uS.call("packageA.State#fromCity", "NY");
DPR modules
packageA.City#cleanCity module
612
packageA.State#fromCity module
614

FIG. 6