



(19) **United States**

(12) **Patent Application Publication**  
**Khorshid et al.**

(10) **Pub. No.: US 2024/0119932 A1**

(43) **Pub. Date: Apr. 11, 2024**

(54) **SYSTEMS AND METHODS FOR IMPLEMENTING SMART ASSISTANT SYSTEMS**

filed on Jul. 27, 2023, provisional application No. 63/516,948, filed on Aug. 1, 2023.

(71) Applicant: **Meta Platforms, Inc.**, Menlo Park, CA (US)

(72) Inventors: **Mokhtar Mohamed Khorshid**, Kirkland, WA (US); **Brian Moran**, Seattle, WA (US); **James Benefits McCready**, Columbia, MO (US); **Ahmed Kamal Atwa Mohamed**, Redmond, WA (US); **Katherina Nguyen**, San Jose, CA (US); **Gary Warren Barbon**, London (GB); **Ryan Bailey**, Rockaway, NJ (US); **Irina-Elena Veliche**, San Mateo, CA (US); **Frank Torsten Bernd Seide**, Yarrow Point, WA (US)

**Publication Classification**

(51) **Int. Cl.**  
*G10L 15/22* (2006.01)  
*G10L 15/18* (2006.01)  
*G10L 15/26* (2006.01)  
(52) **U.S. Cl.**  
CPC ..... *G10L 15/22* (2013.01); *G10L 15/18* (2013.01); *G10L 15/26* (2013.01); *G10L 2015/223* (2013.01); *G10L 2015/225* (2013.01)

(57) **ABSTRACT**

In one embodiment, a system includes an automatic speech recognition (ASR) module, a natural-language understanding (NLU) module, a dialog manager, one or more agents, an arbitrator, a delivery system, one or more processors, and a non-transitory memory coupled to the processors comprising instructions executable by the processors, the processors operable when executing the instructions to receive a user input, process the user input using the ASR module, the NLU module, the dialog manager, one or more of the agents, the arbitrator, and the delivery system, and provide a response to the user input.

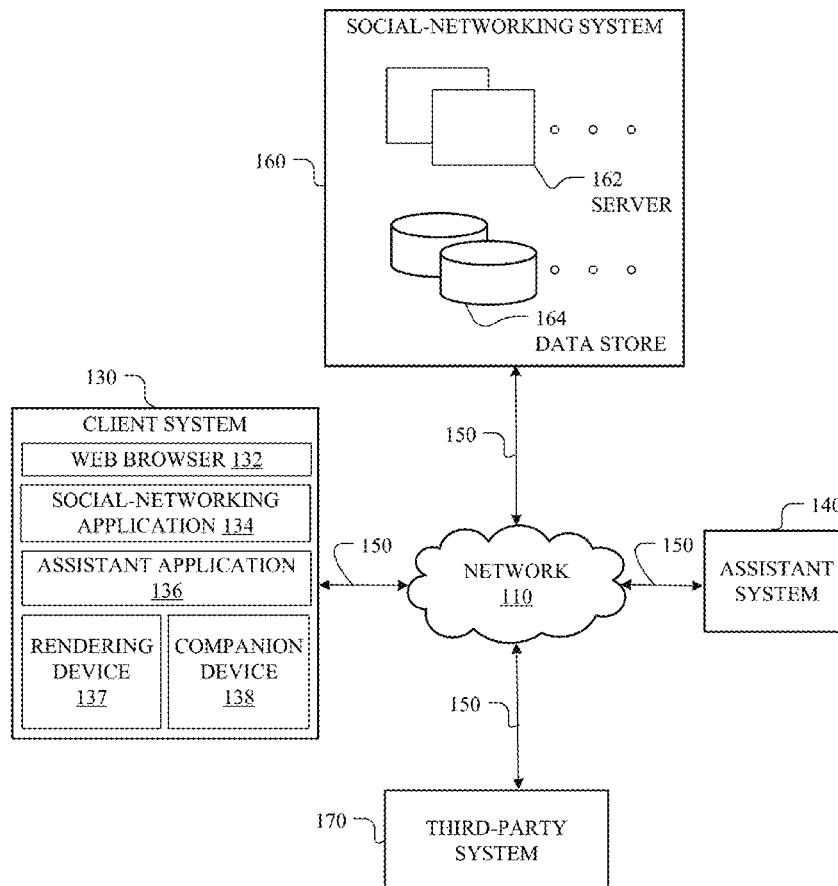
(21) Appl. No.: **18/466,568**

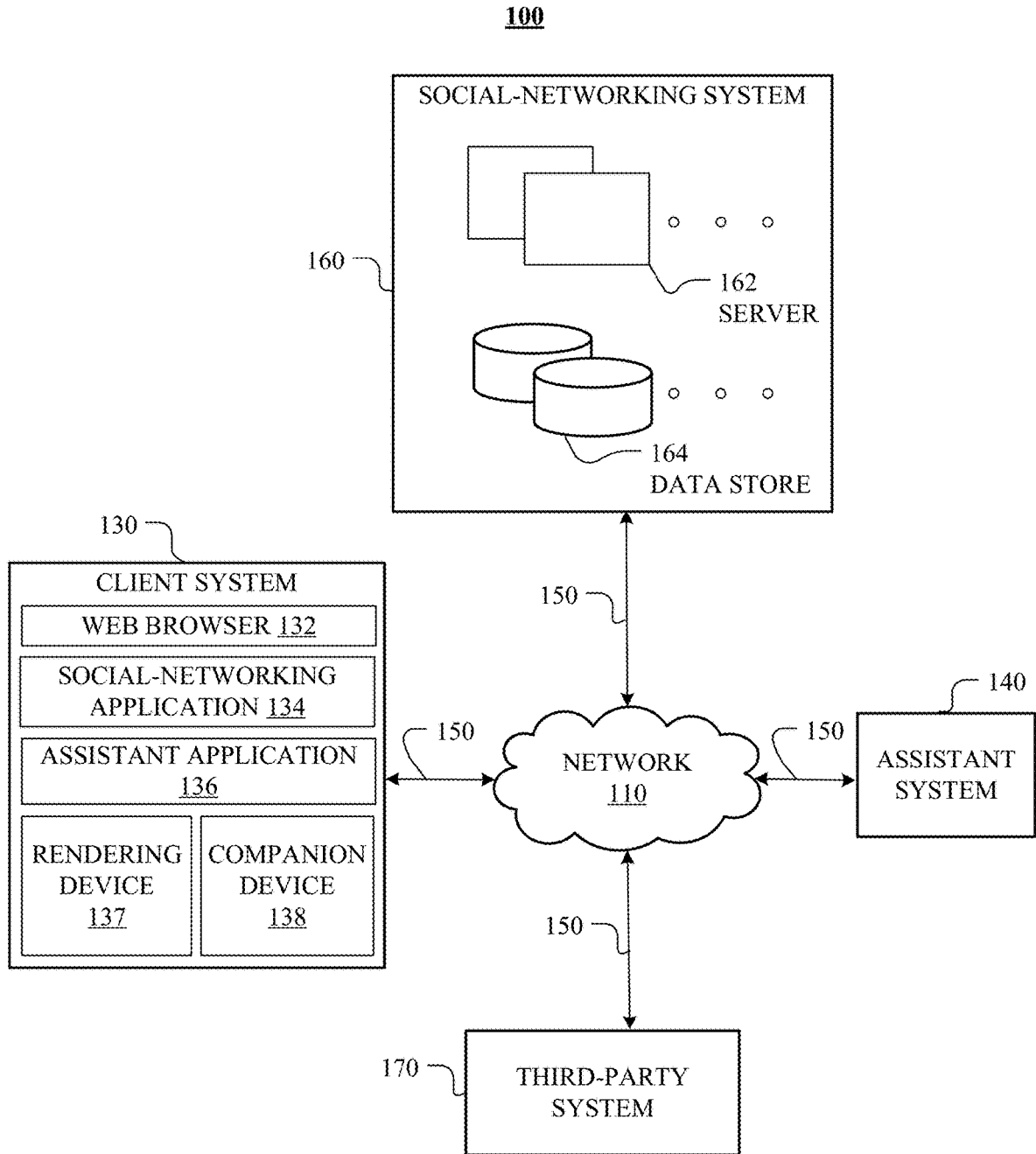
(22) Filed: **Sep. 13, 2023**

**Related U.S. Application Data**

(60) Provisional application No. 63/376,887, filed on Sep. 23, 2022, provisional application No. 63/516,019,

**100**





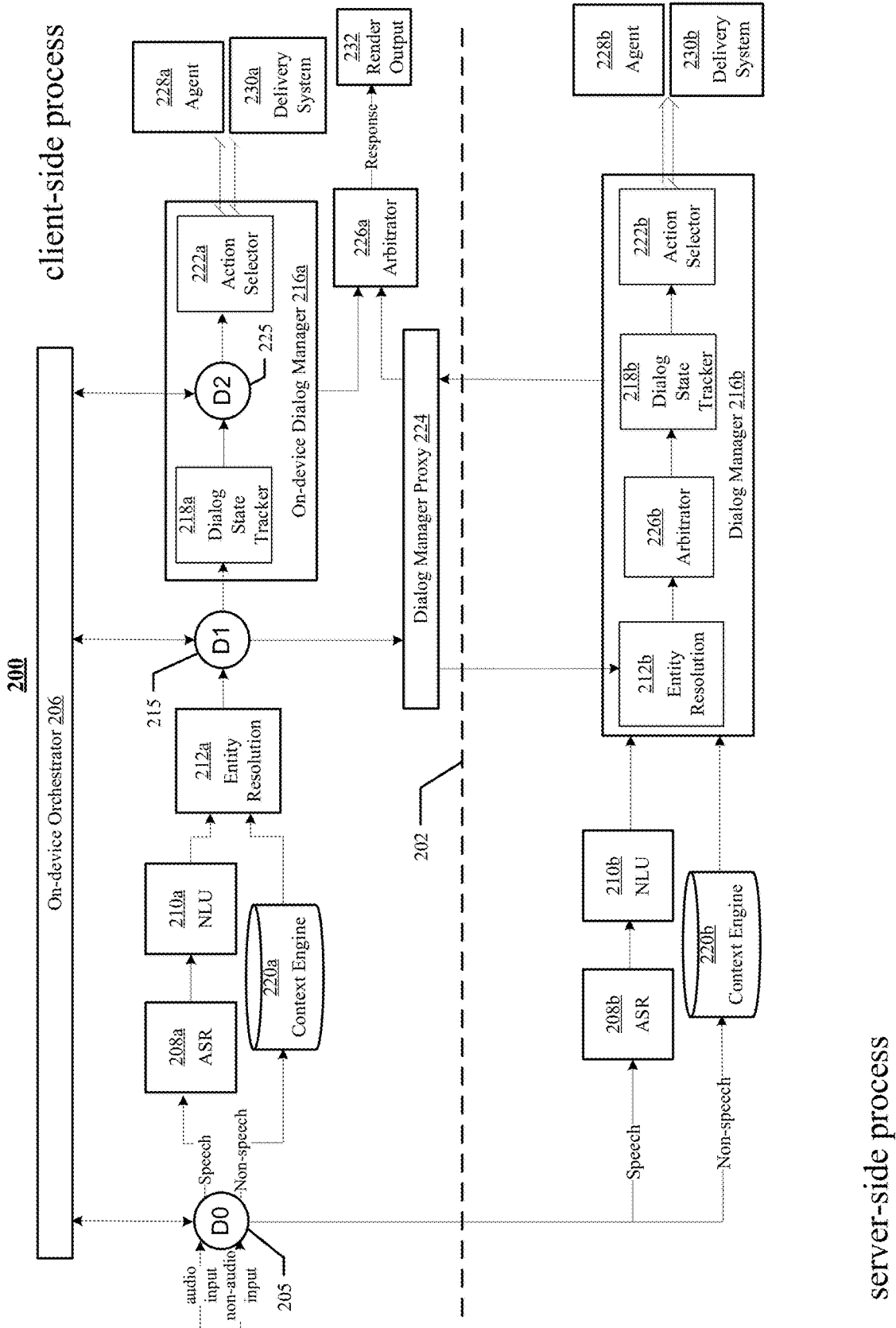


FIG. 2

server-side process

300

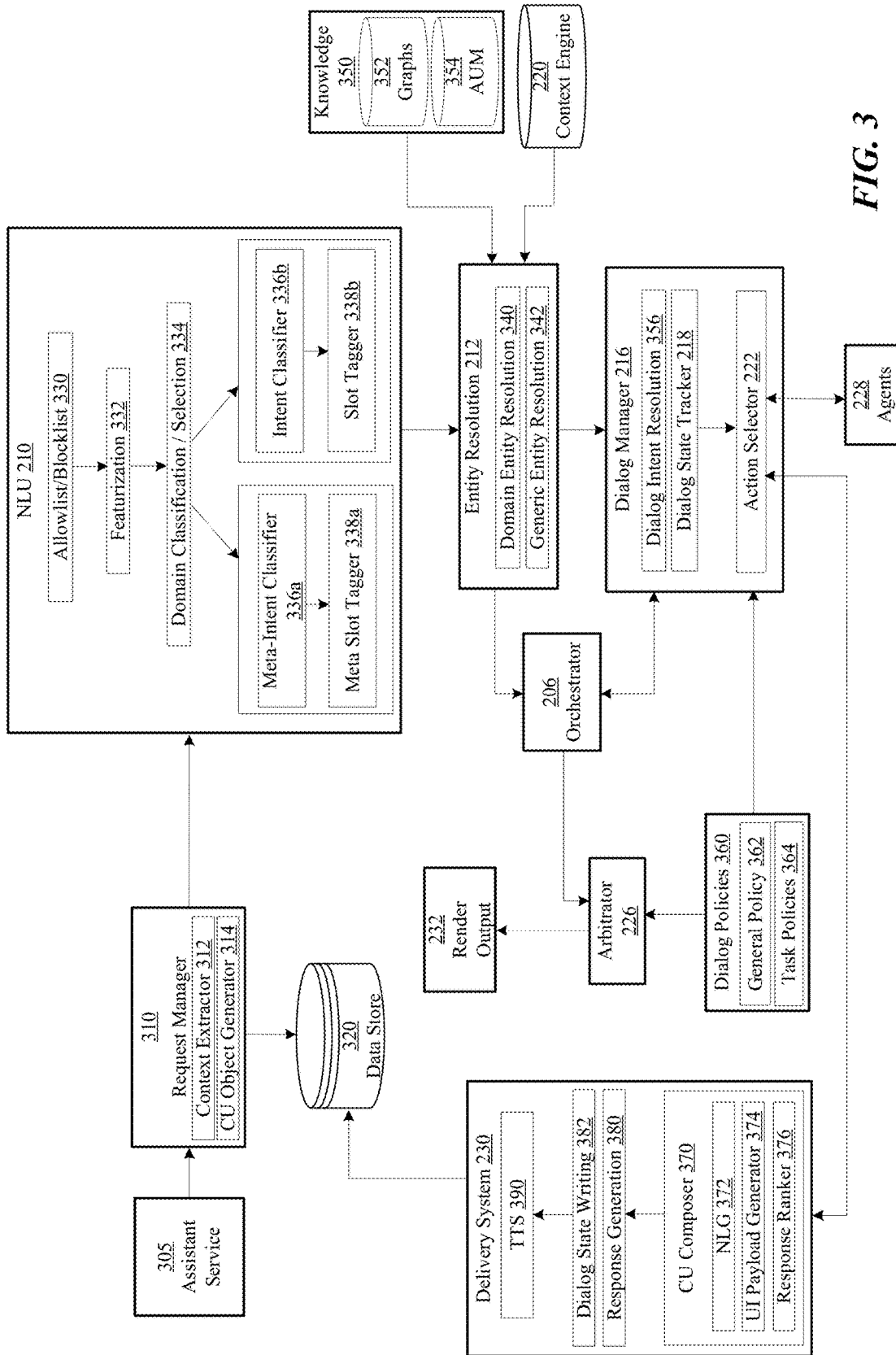


FIG. 3

400

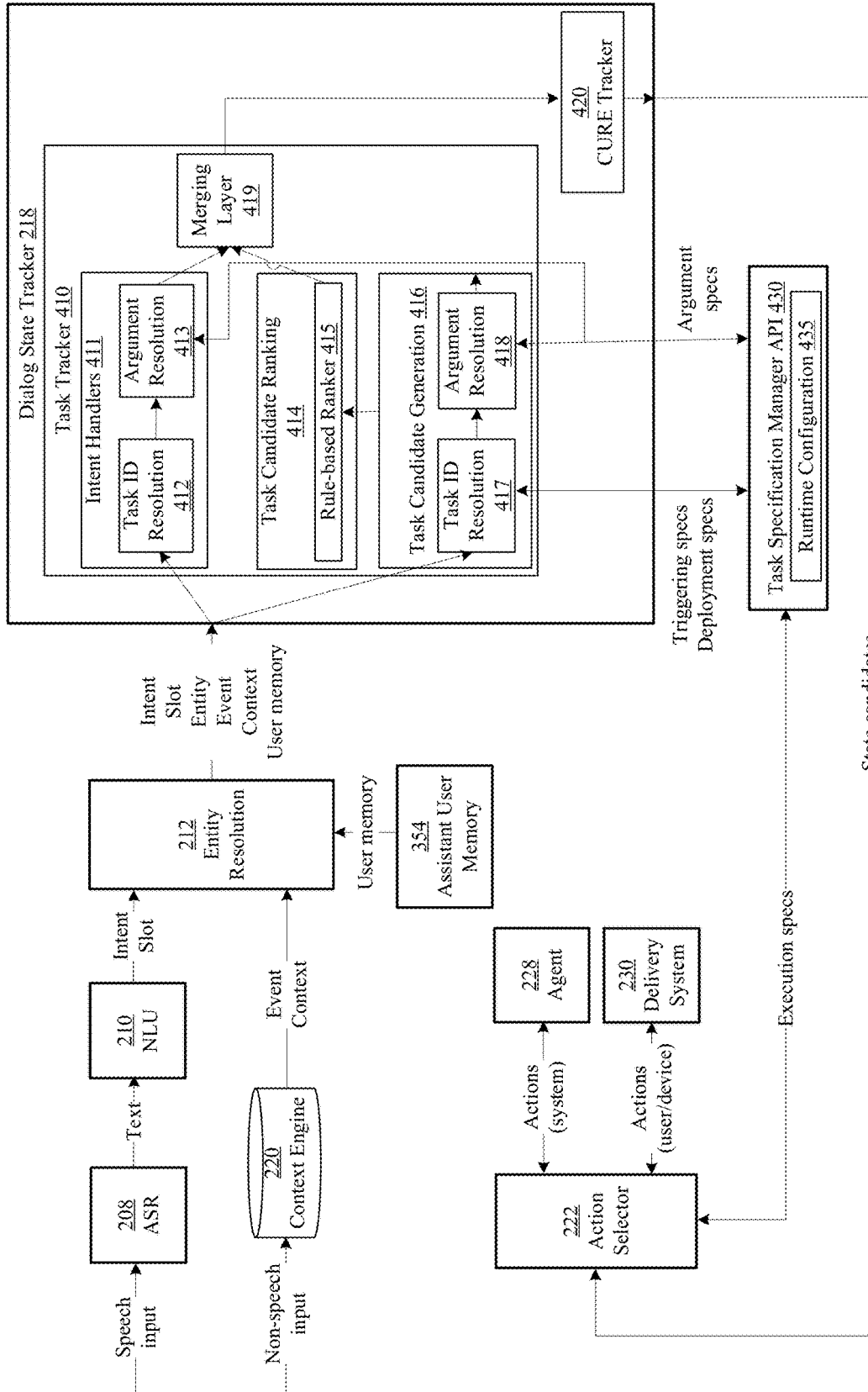


FIG. 4

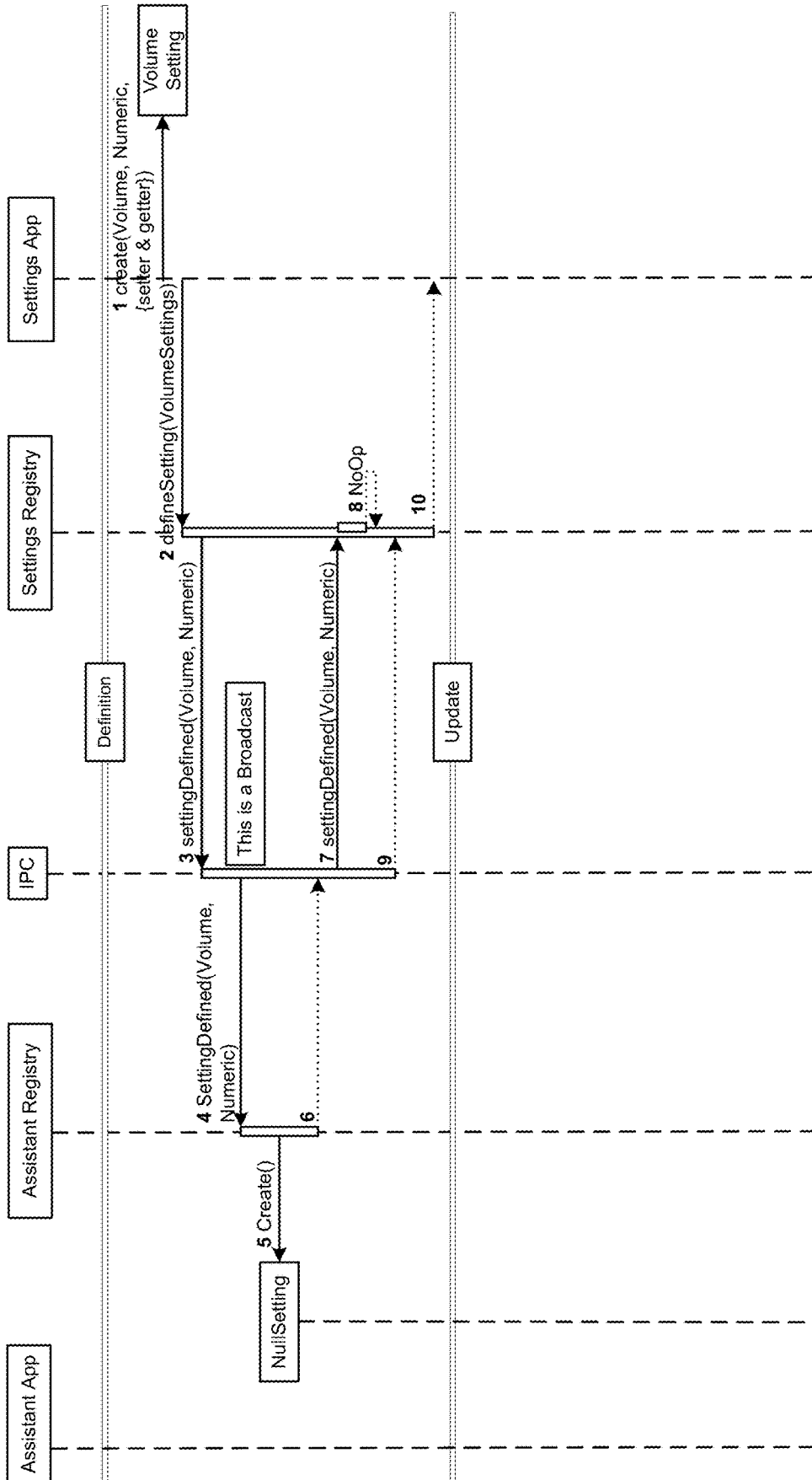


FIG. 5A

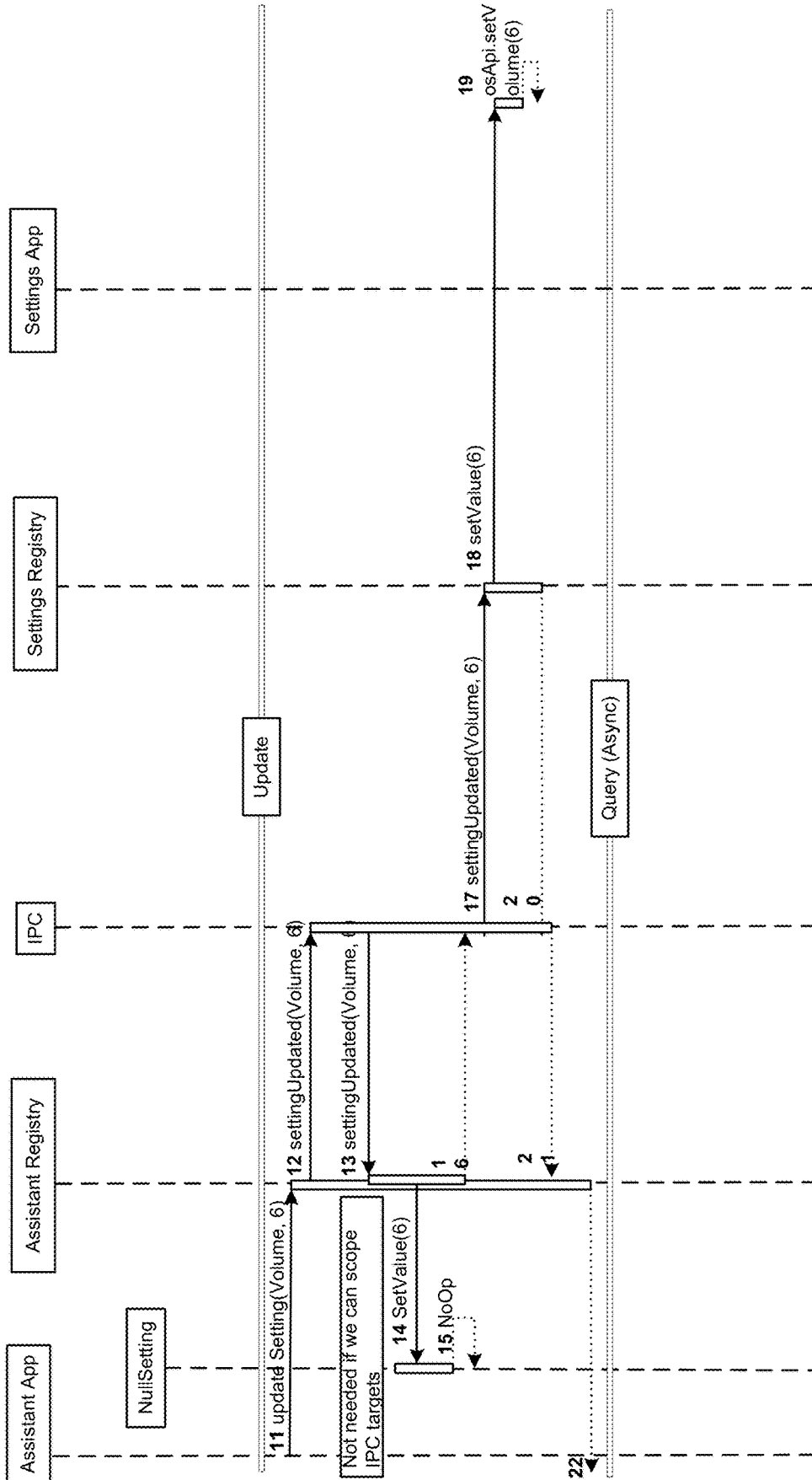


FIG. 5B

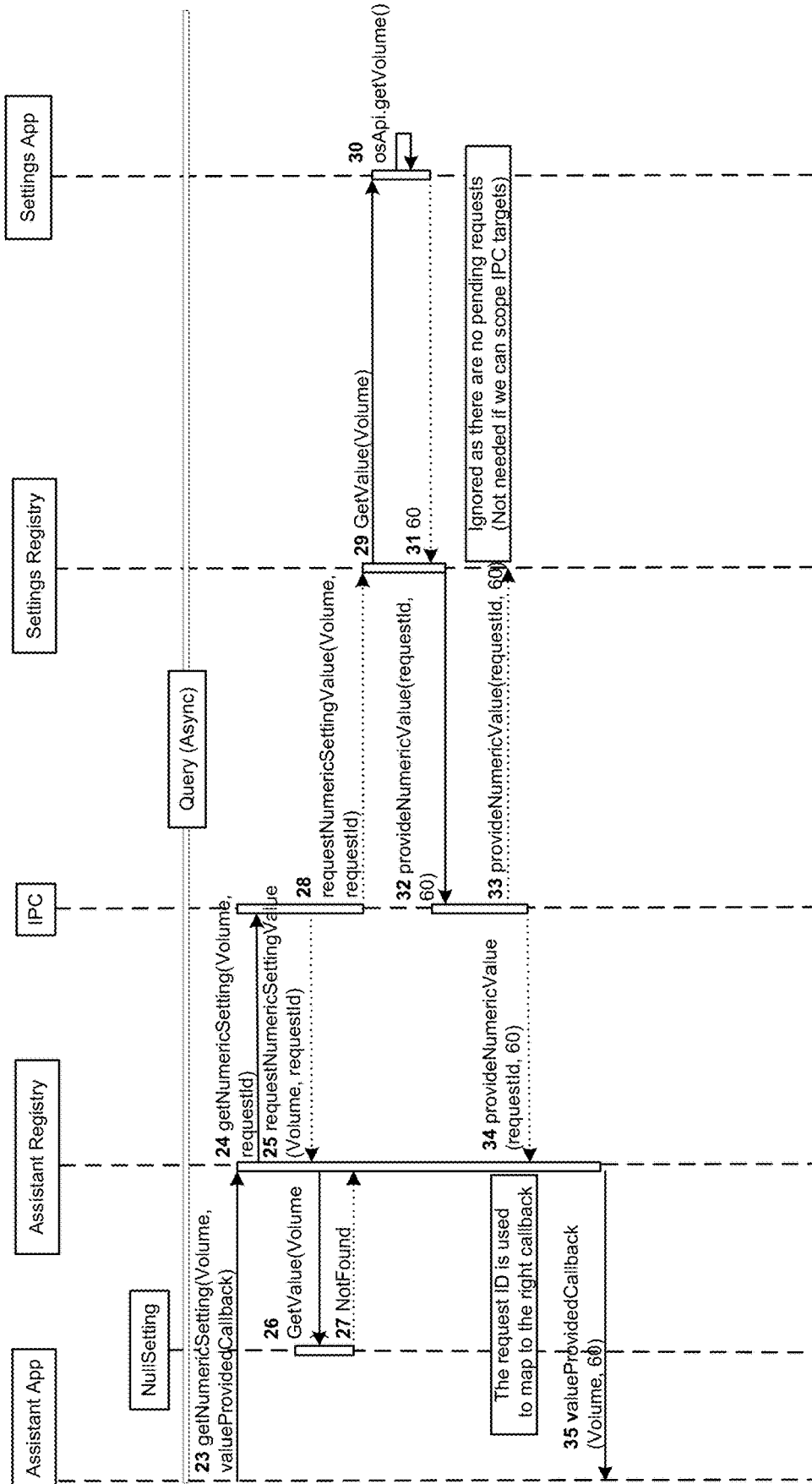


FIG. 5C



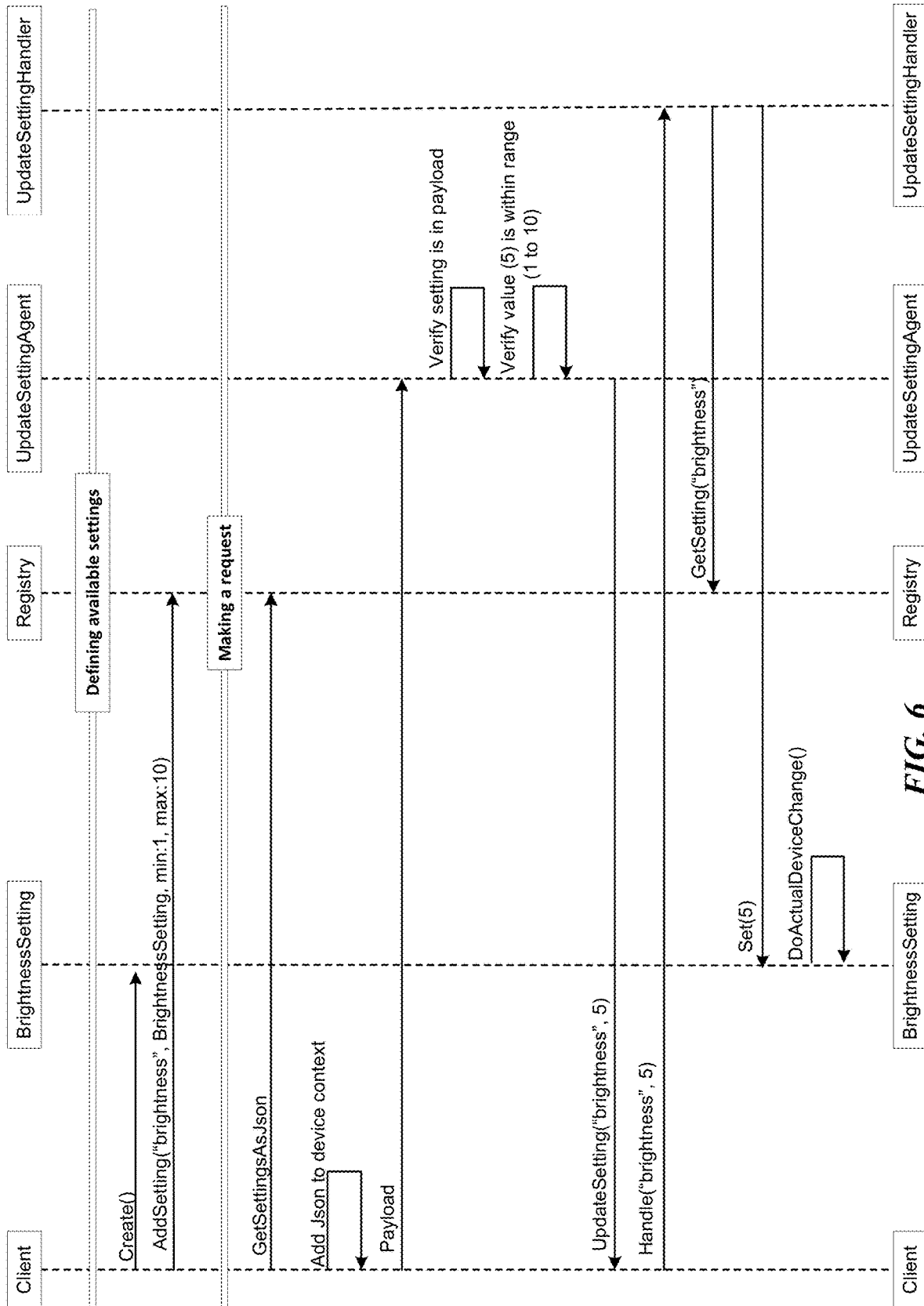


FIG. 6

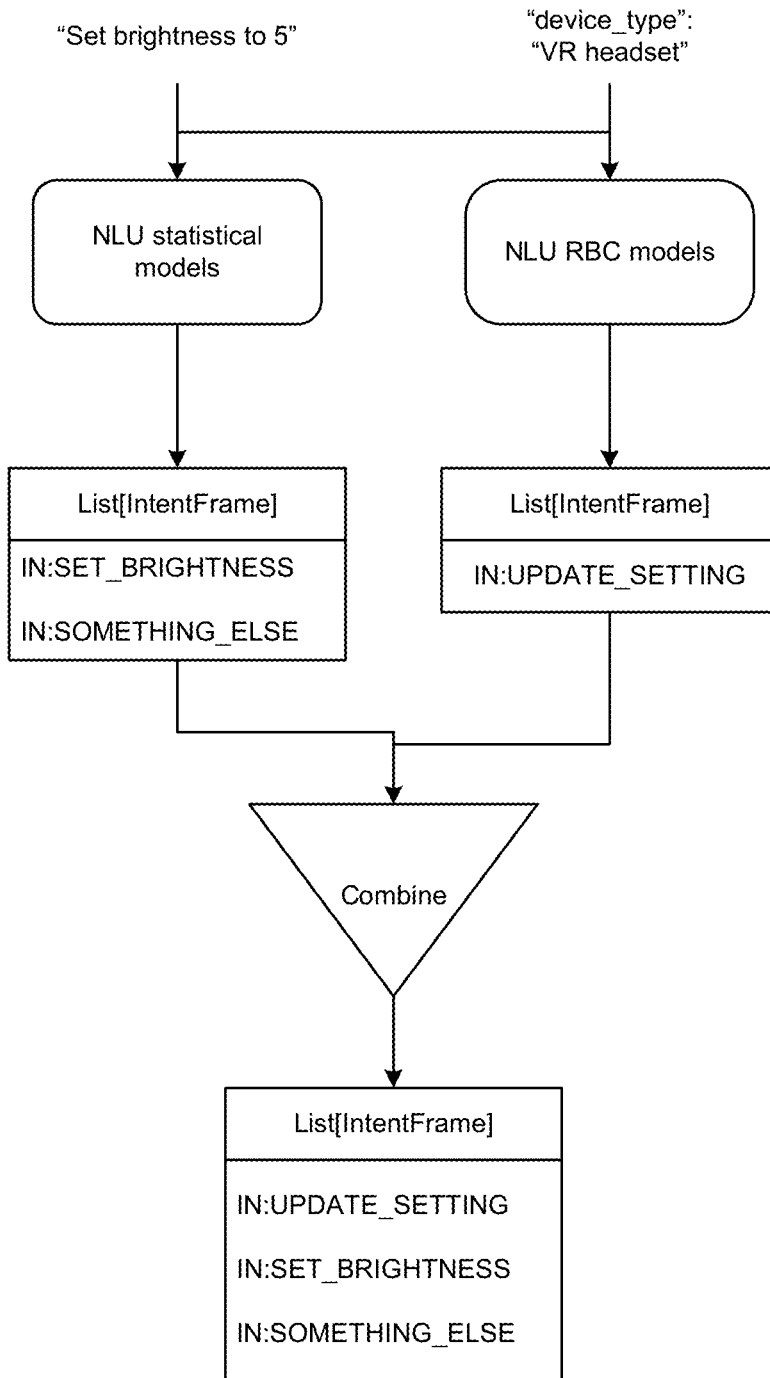
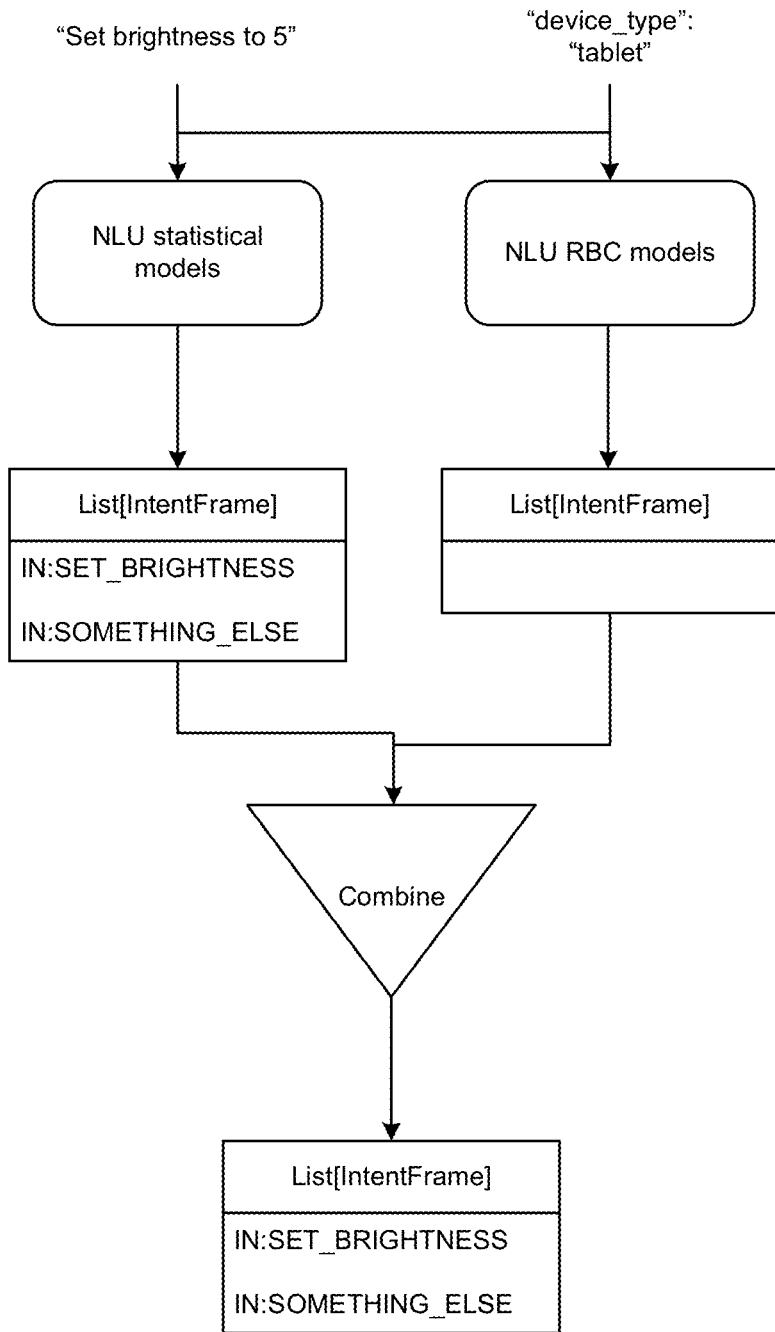
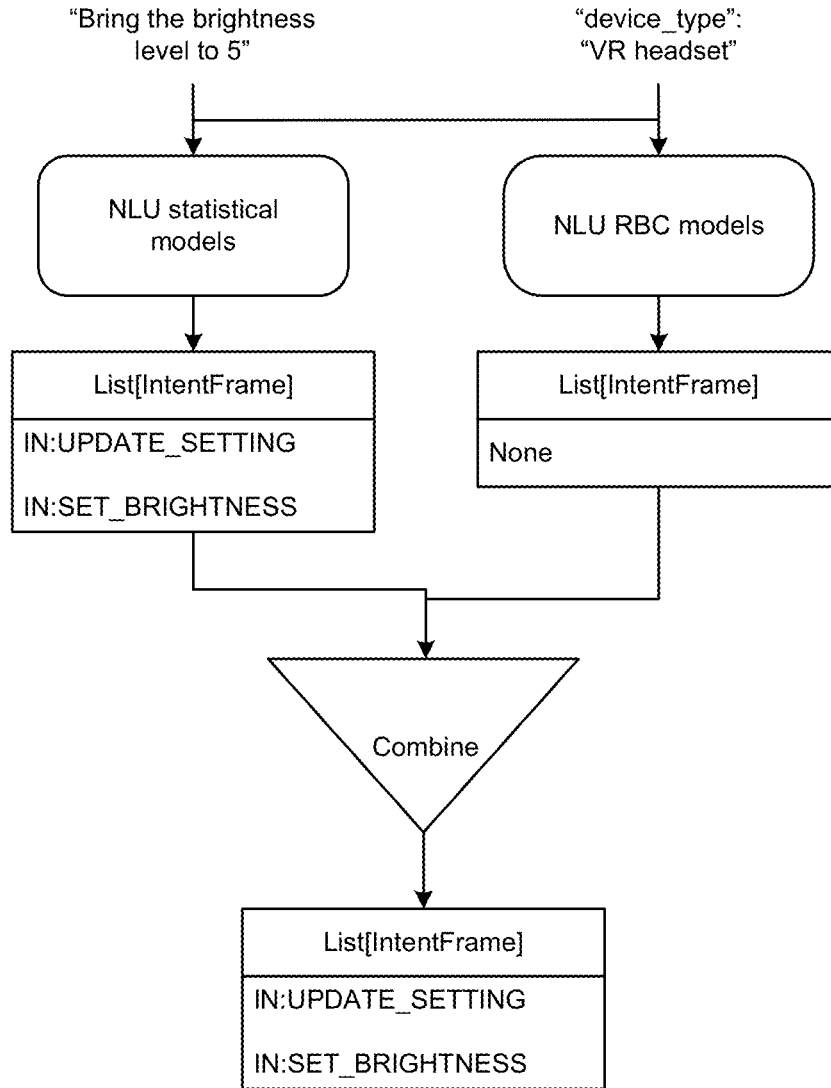


FIG. 7A



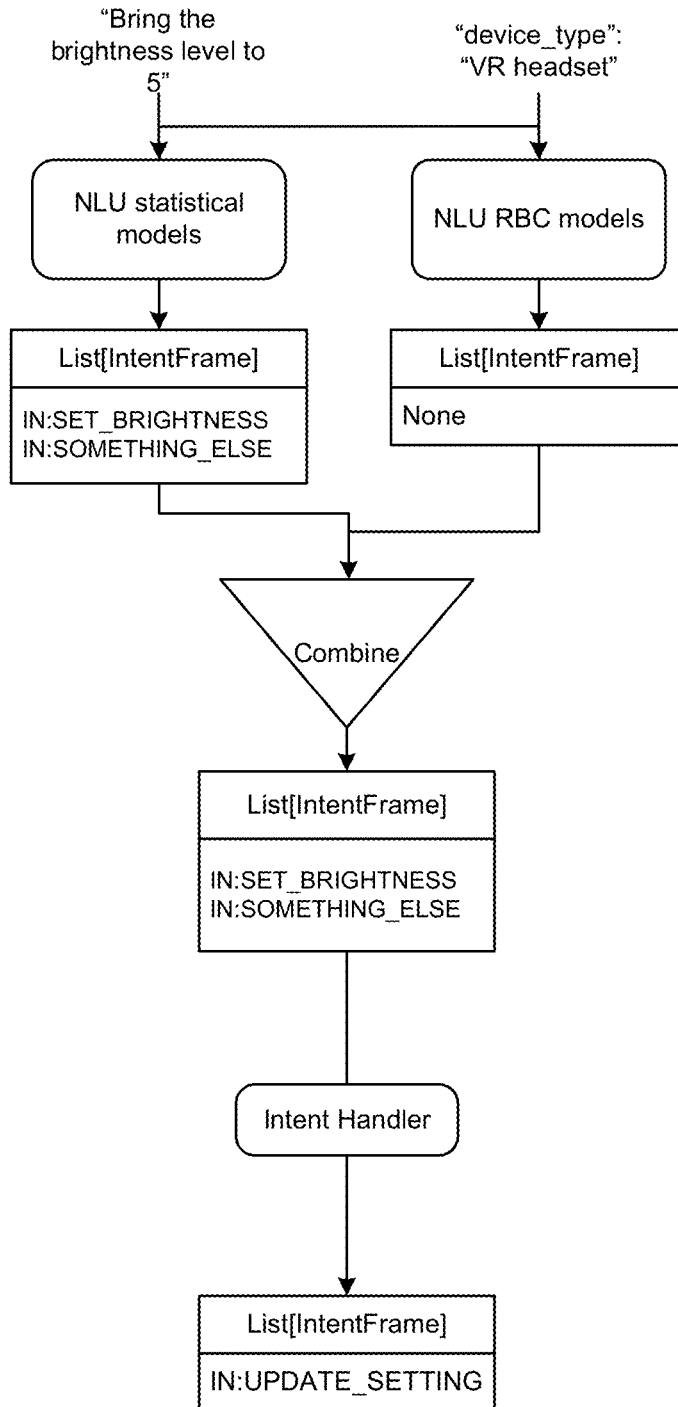
**FIG. 7B**

# Modeled Device Context



**FIG. 8A**

# Intent Handler Transform



**FIG. 8B**

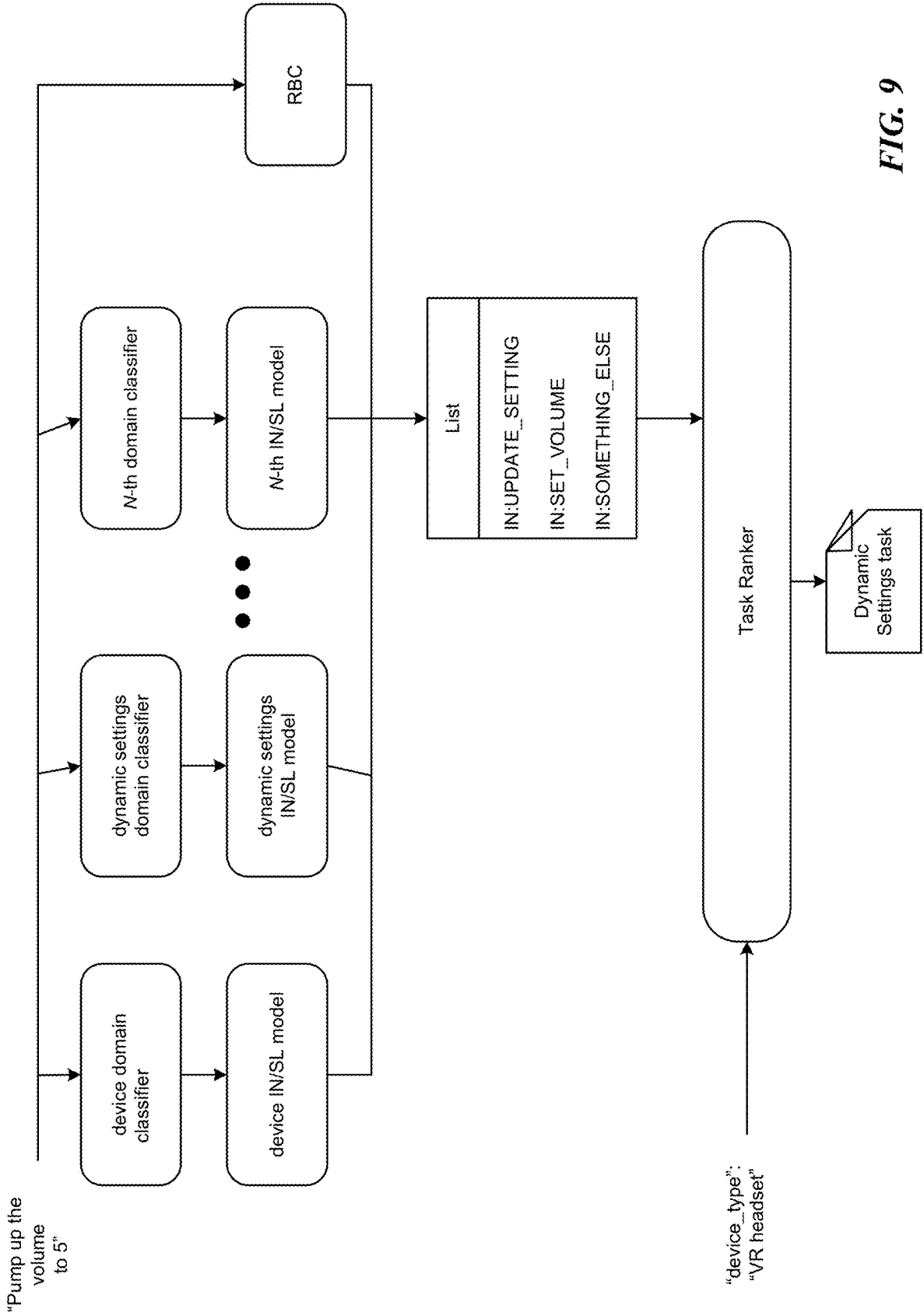


FIG. 9

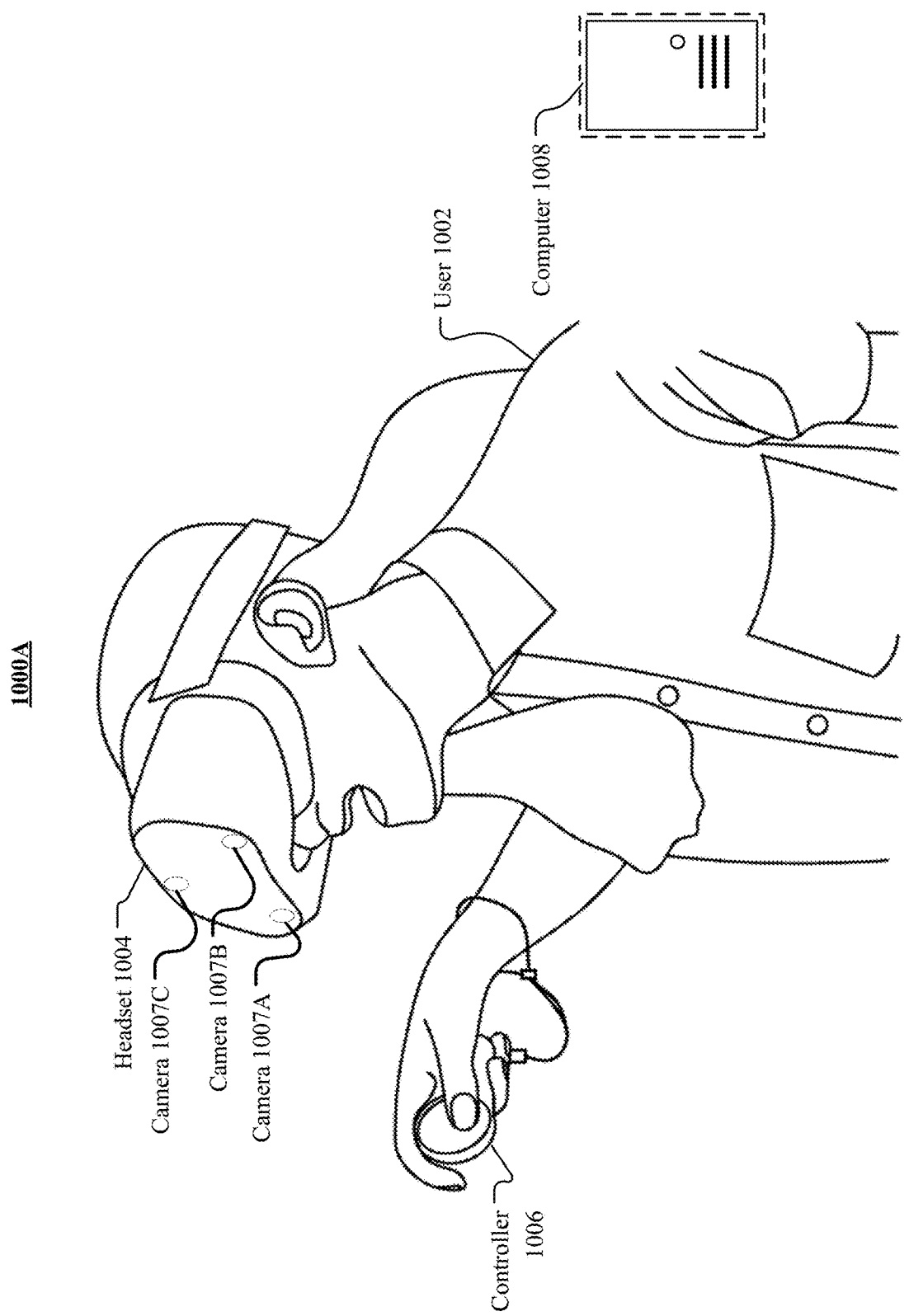
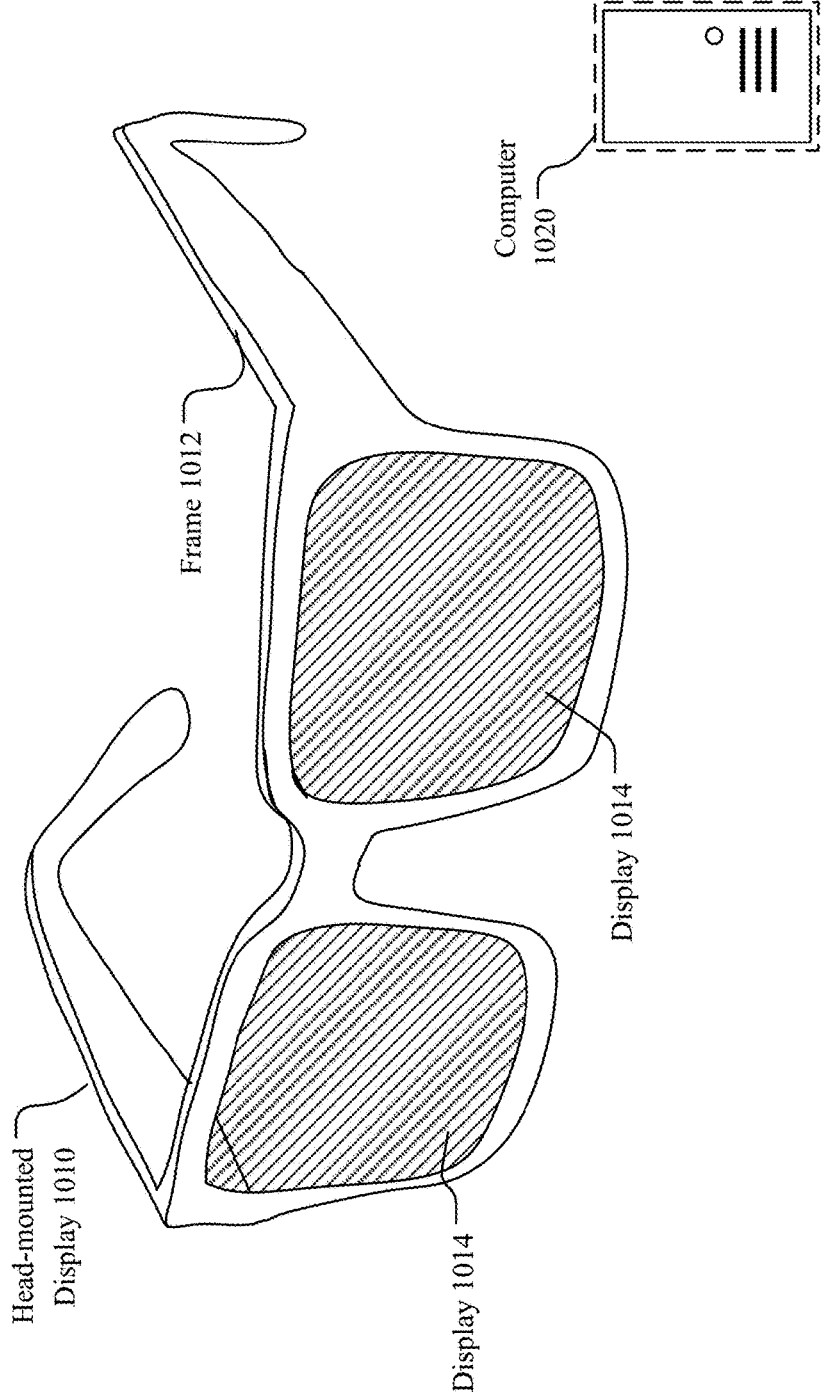


FIG. 10A

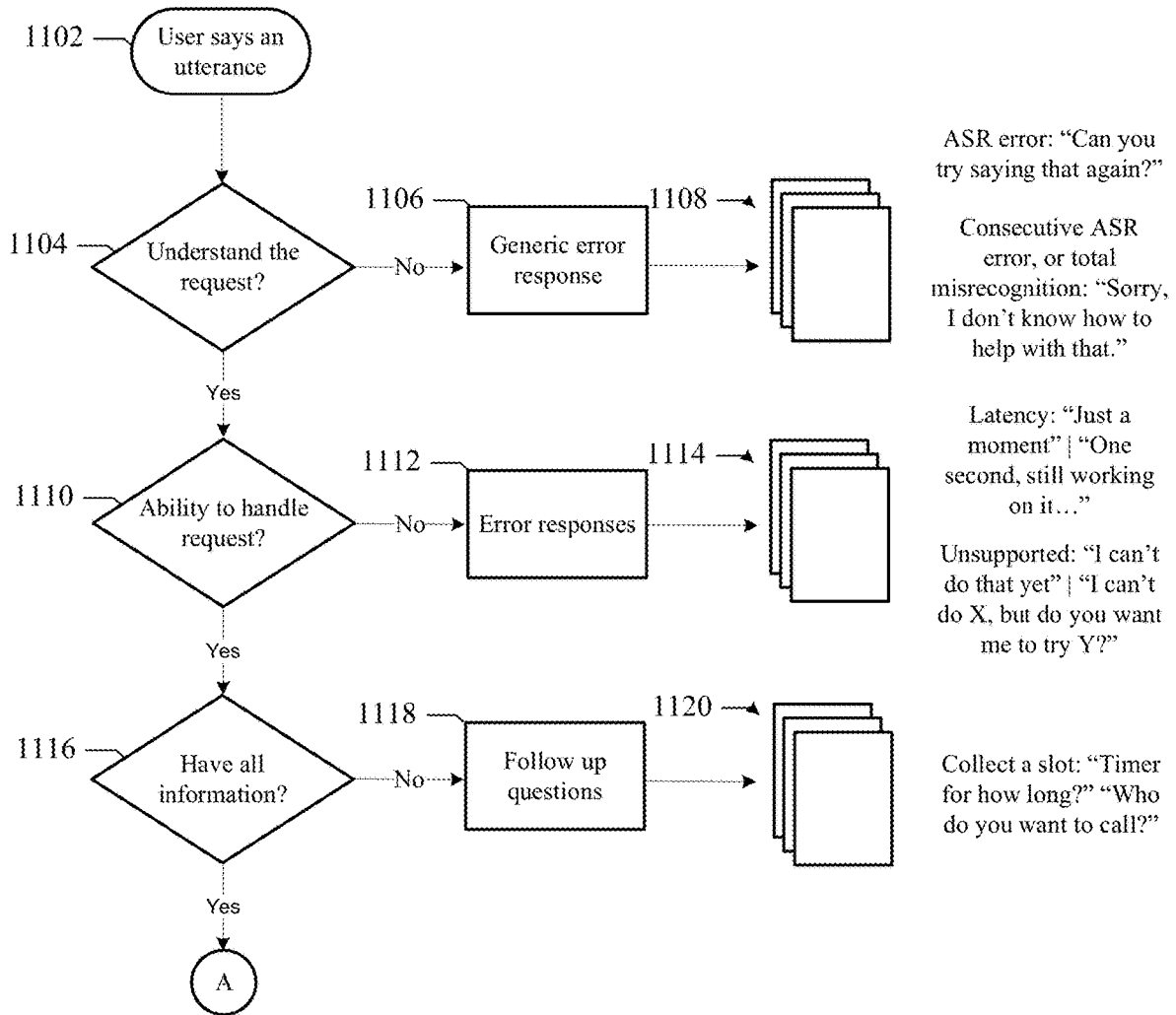
1000B



**FIG. 10B**

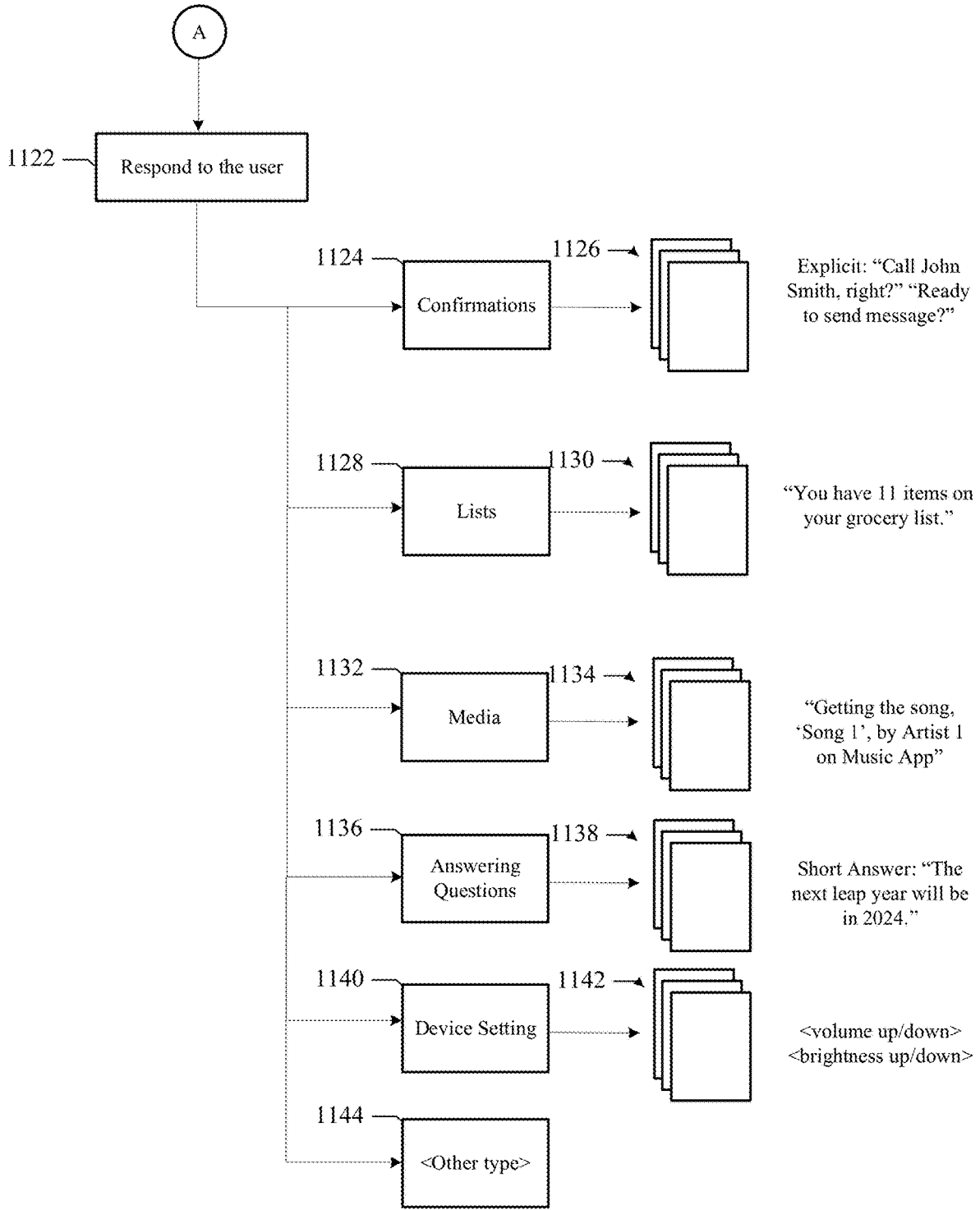


**1100**



**FIG. 11A**

**1100**



**FIG. 11B**

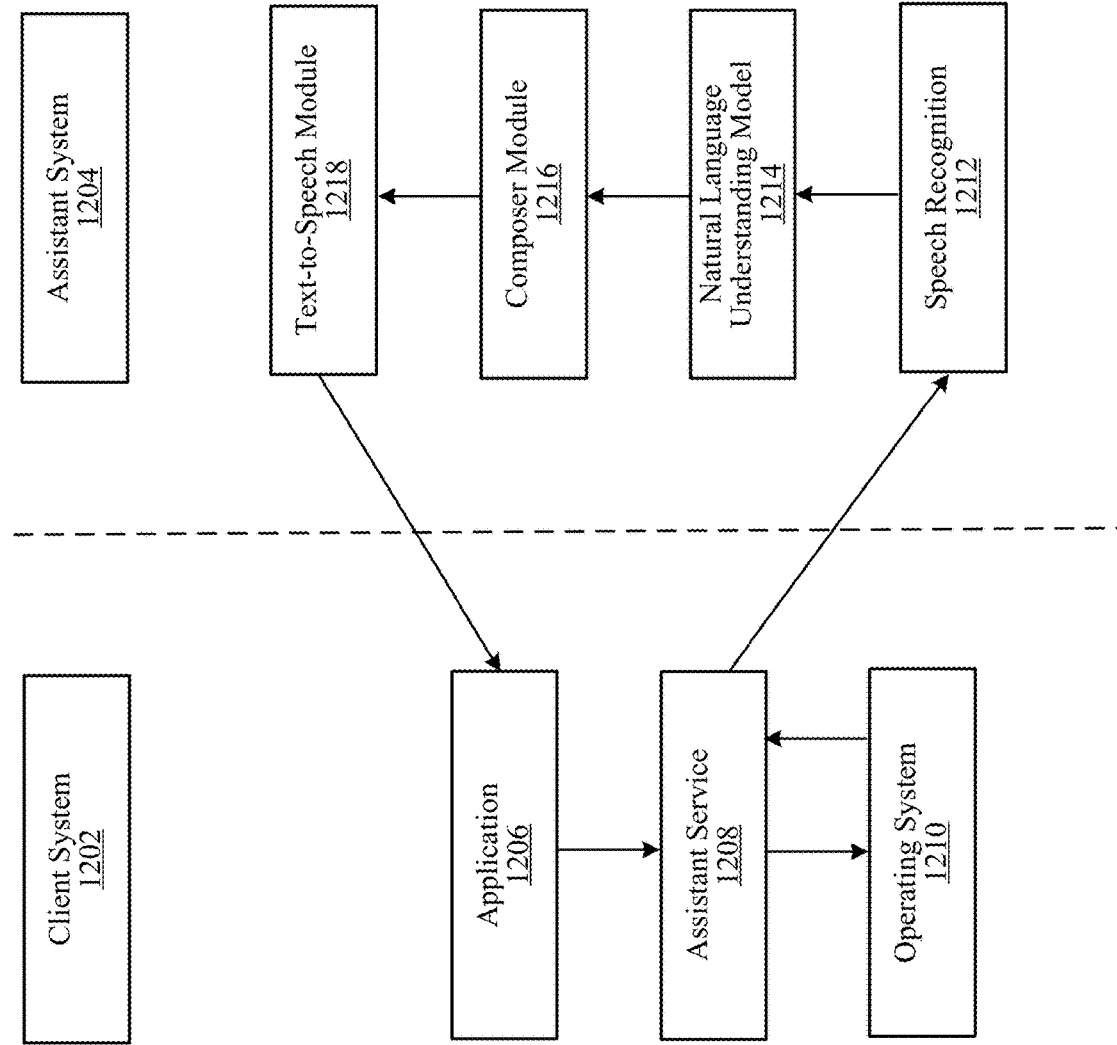


FIG. 12

1200

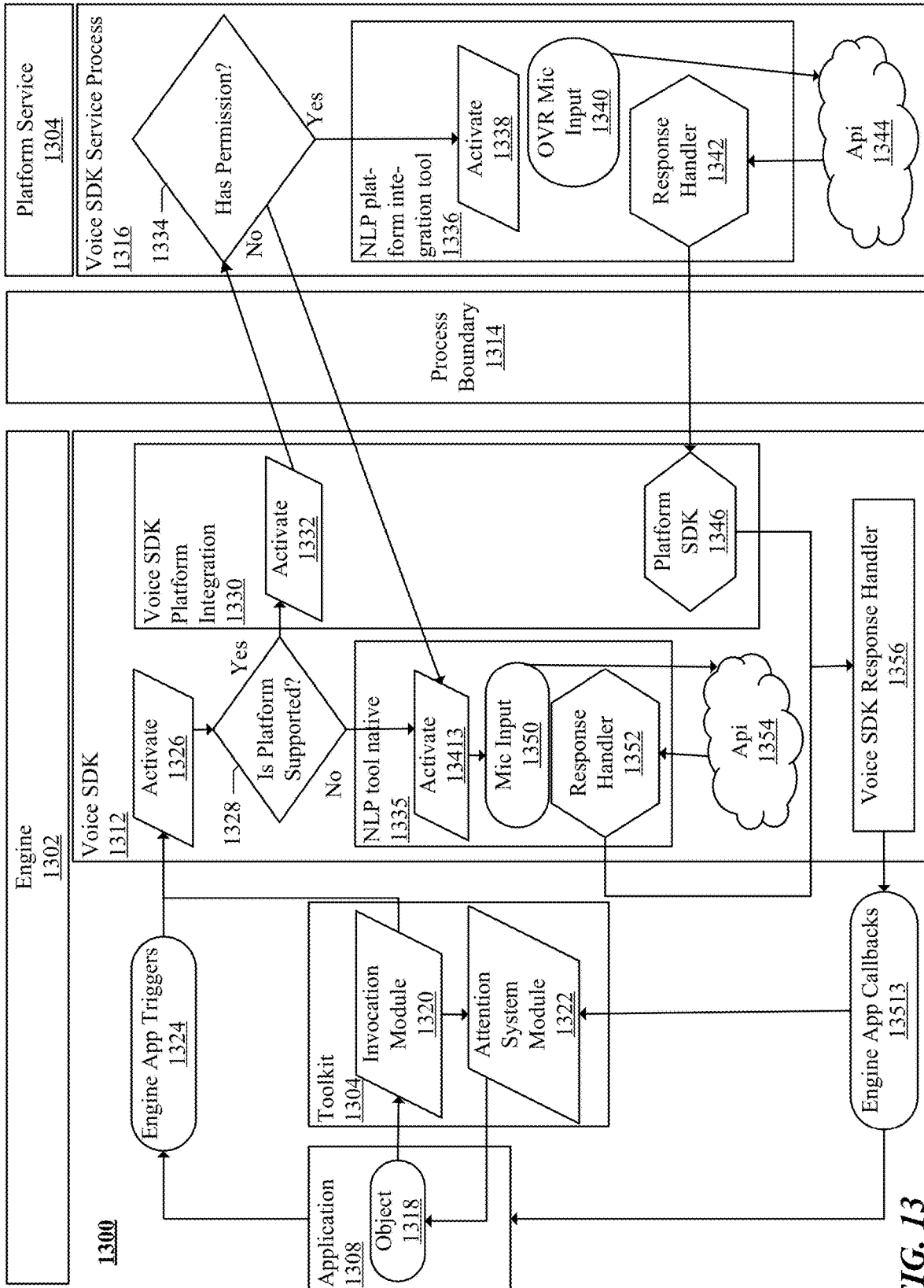


FIG. 13

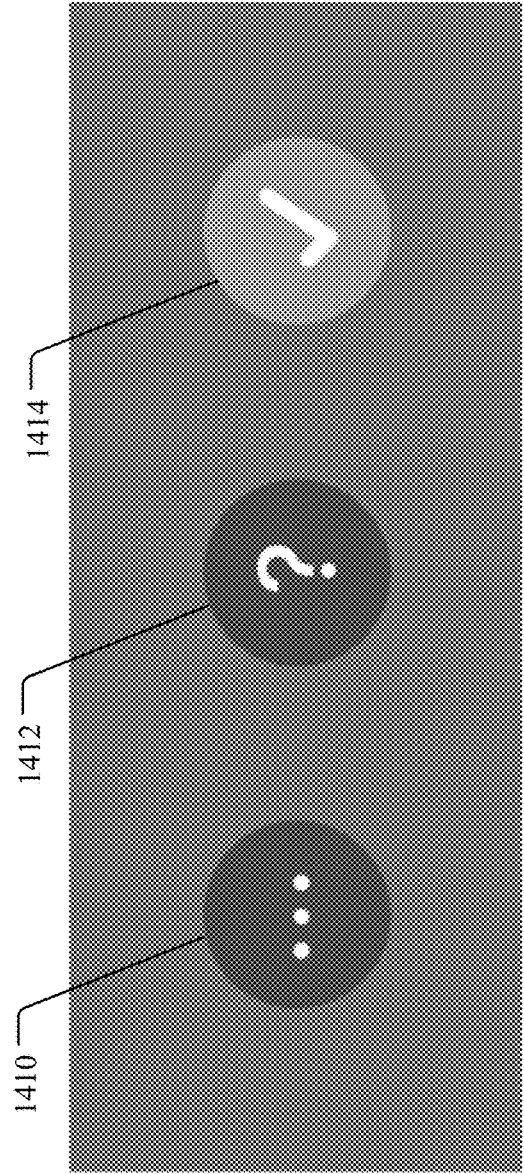
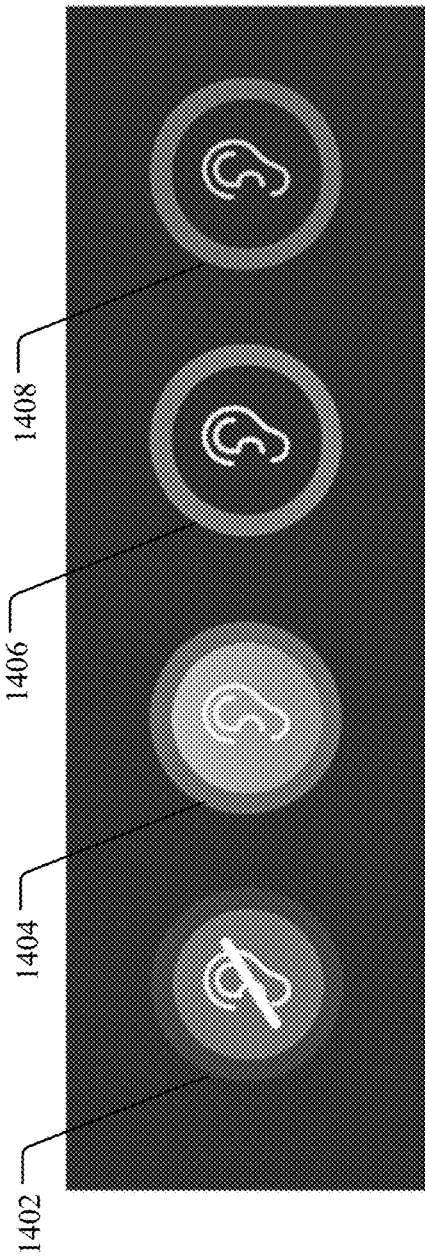


FIG. 14

1500

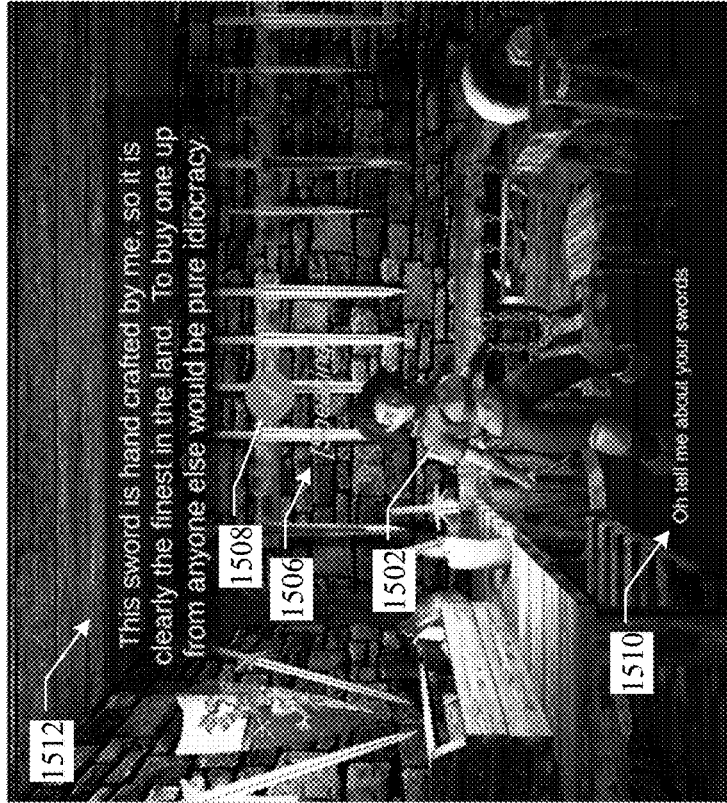


FIG. 15B

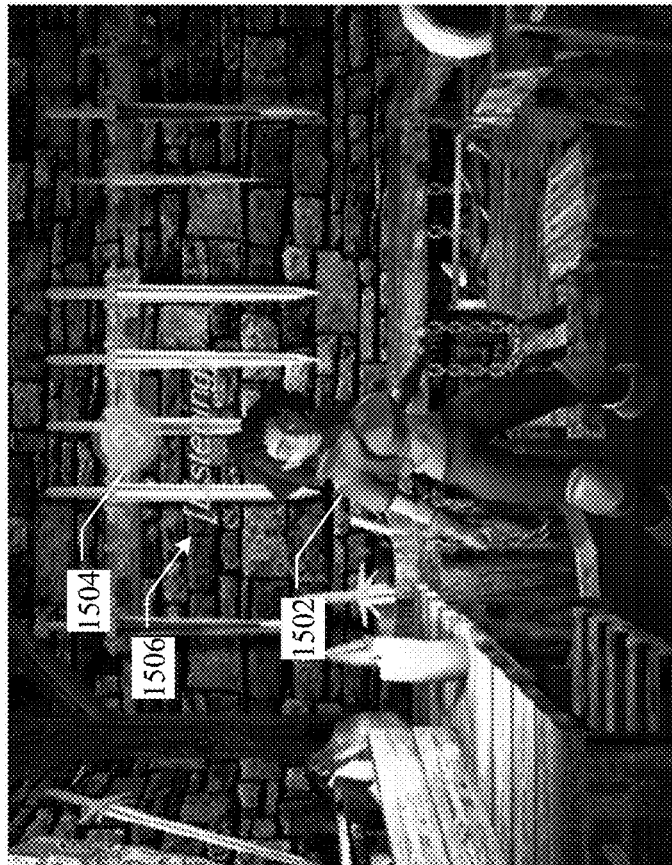
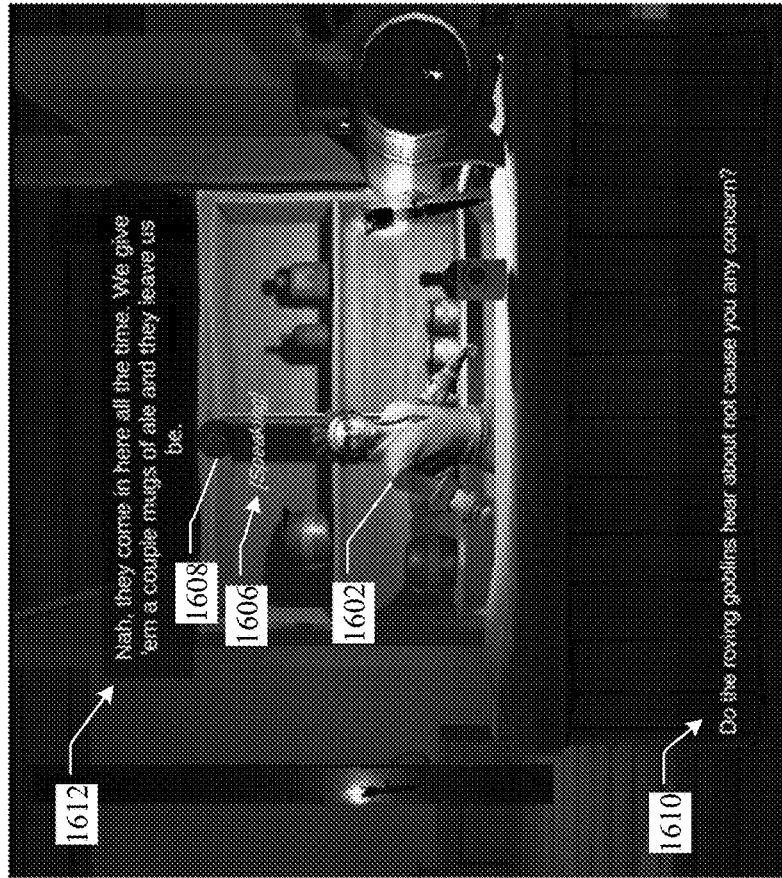
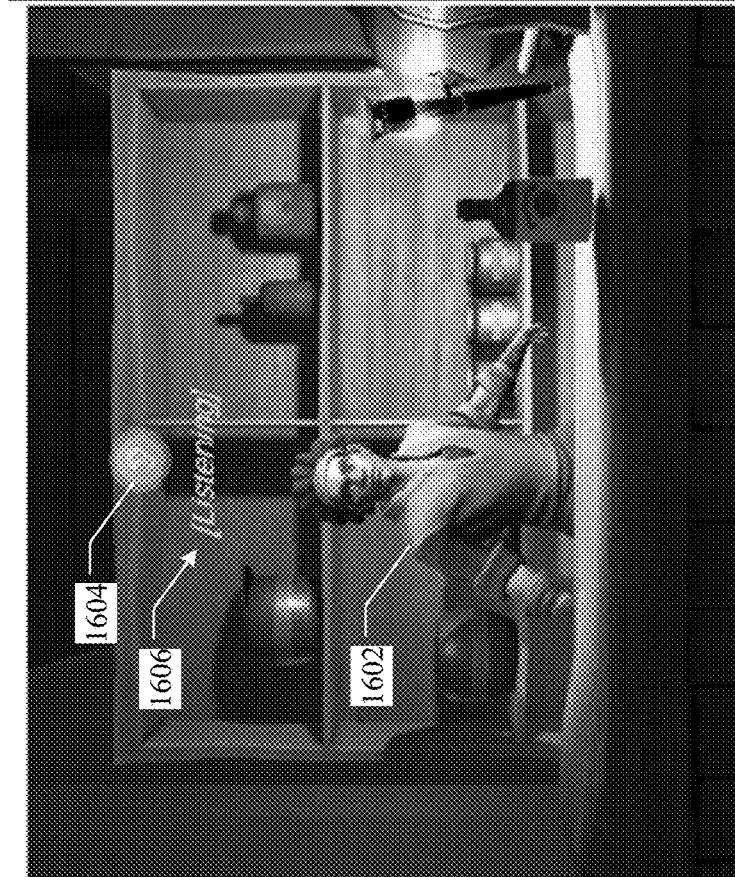


FIG. 15A

1600

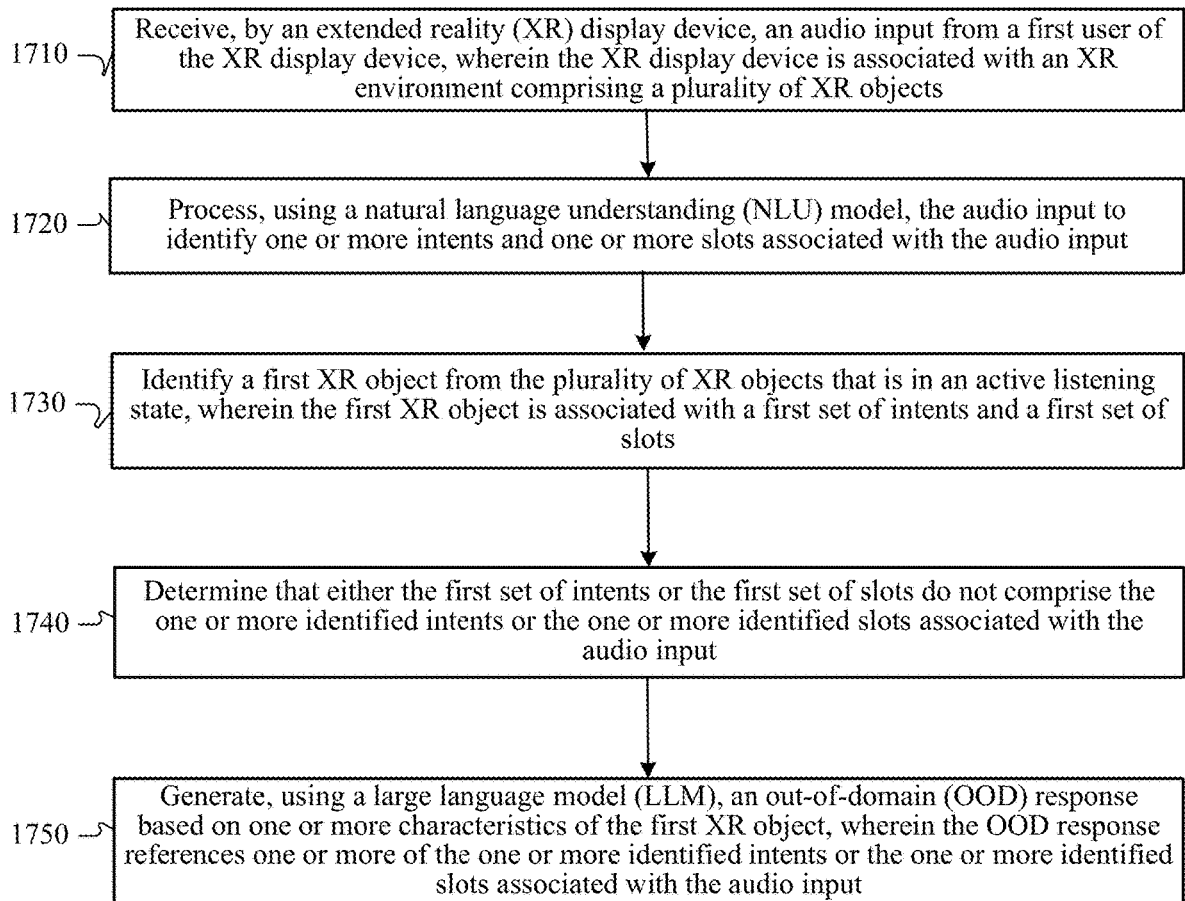


**FIG. 16A**



**FIG. 16B**

**1700**



**FIG. 17**



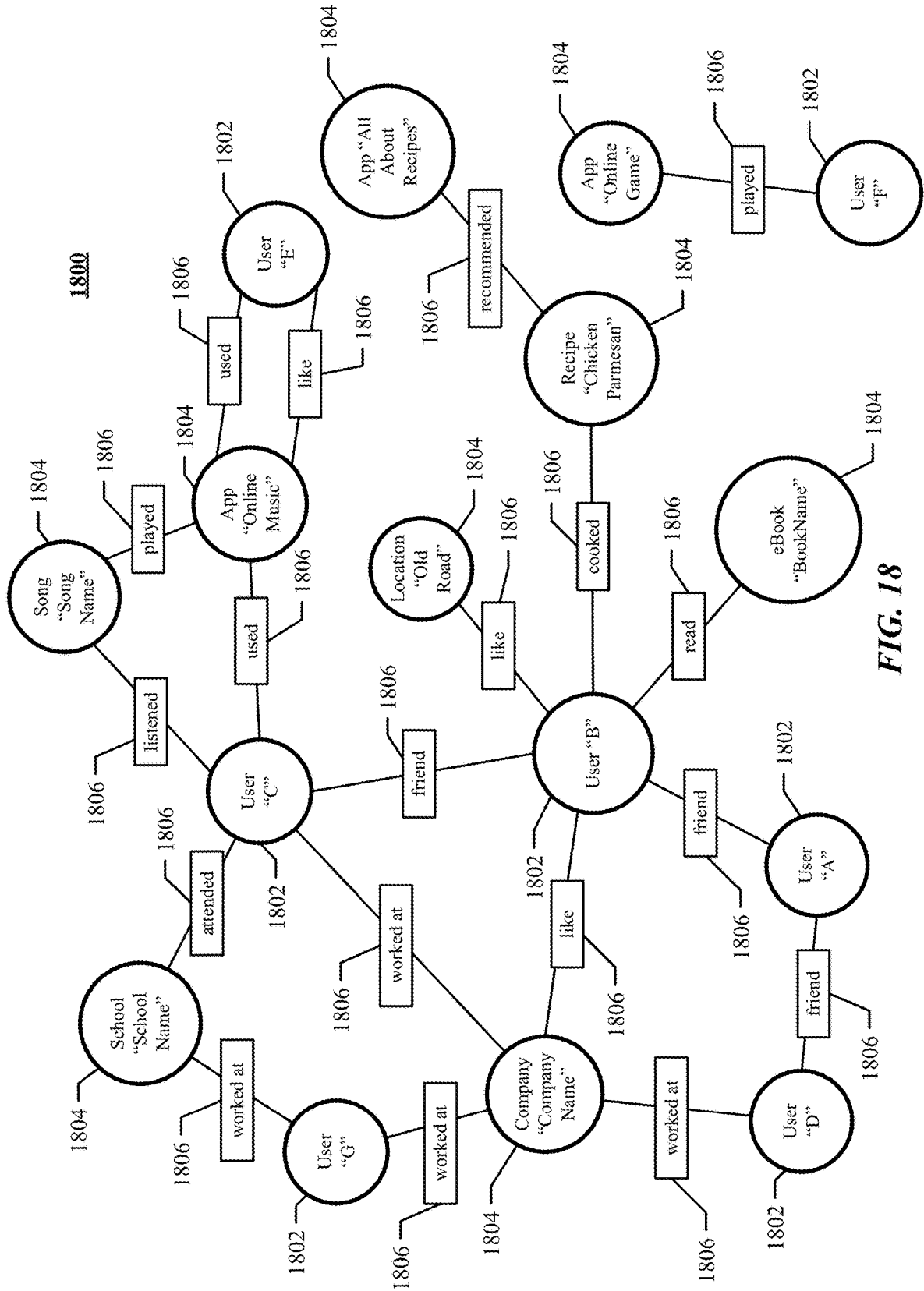
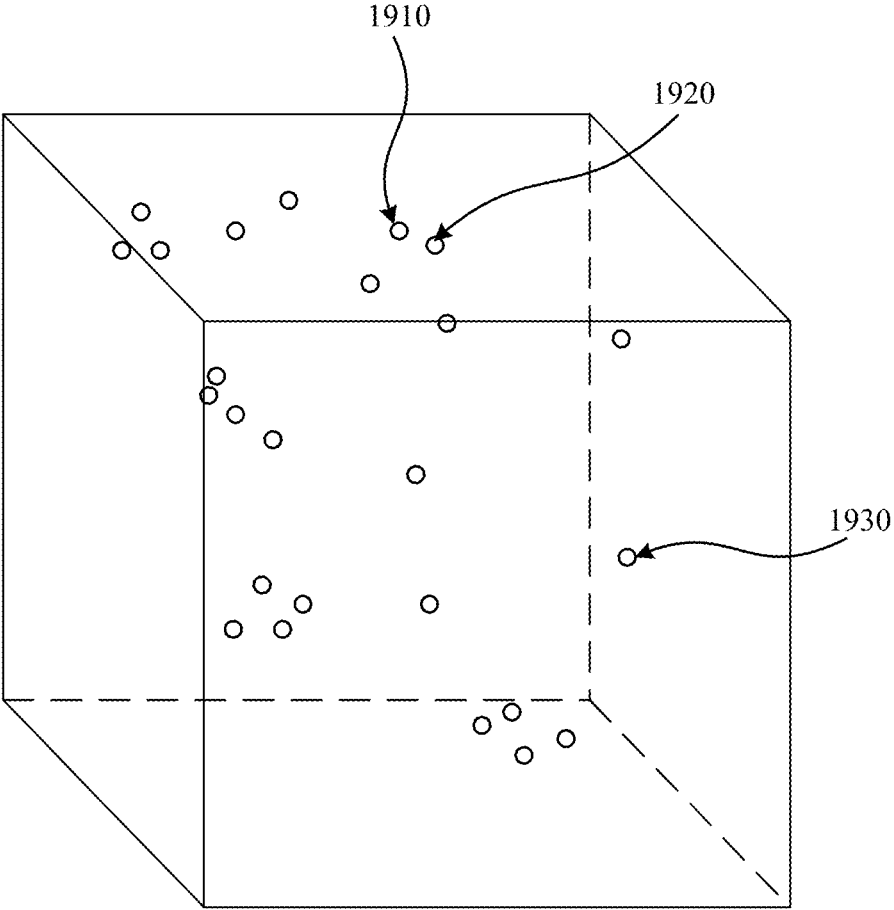


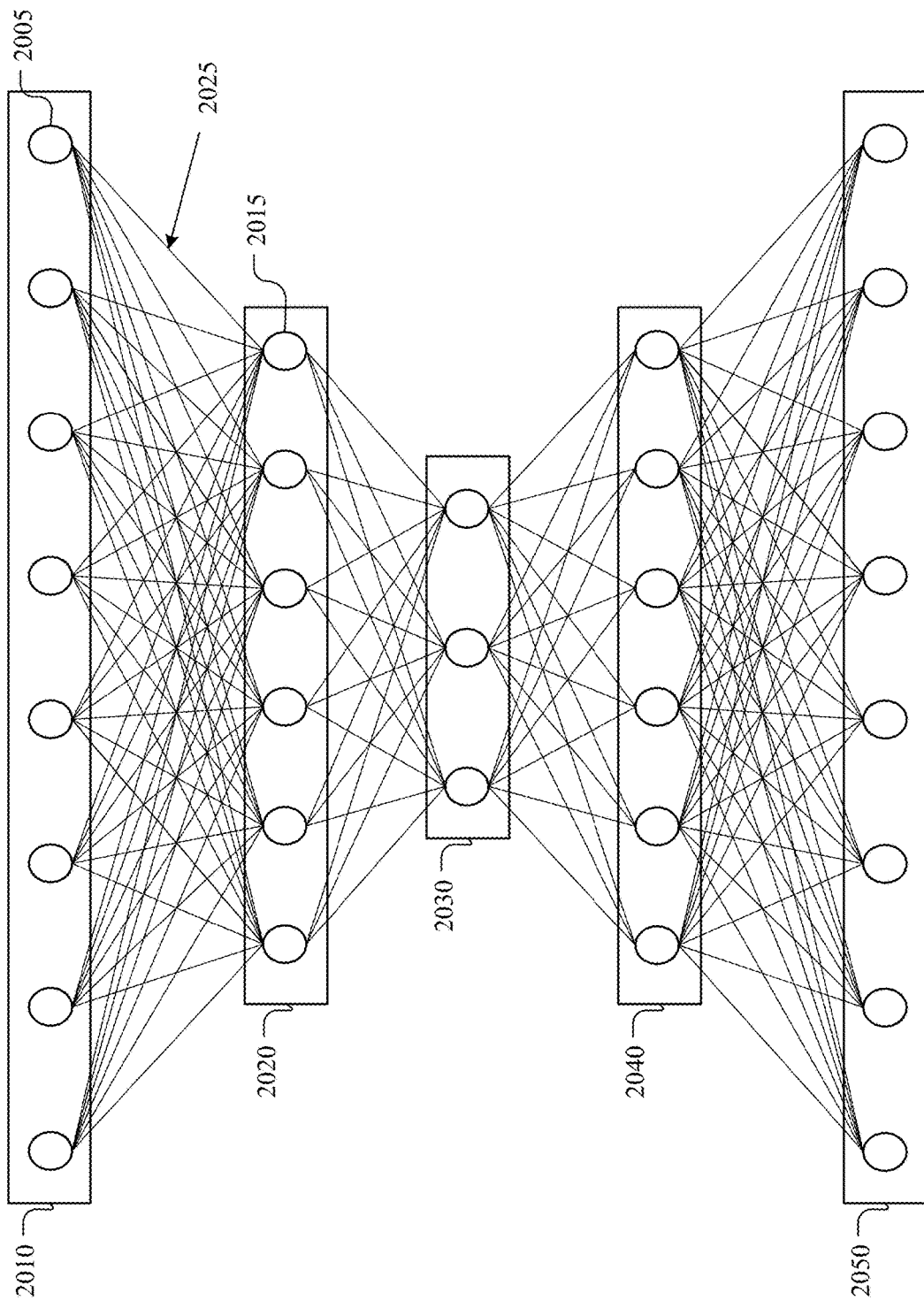
FIG. 18

1900

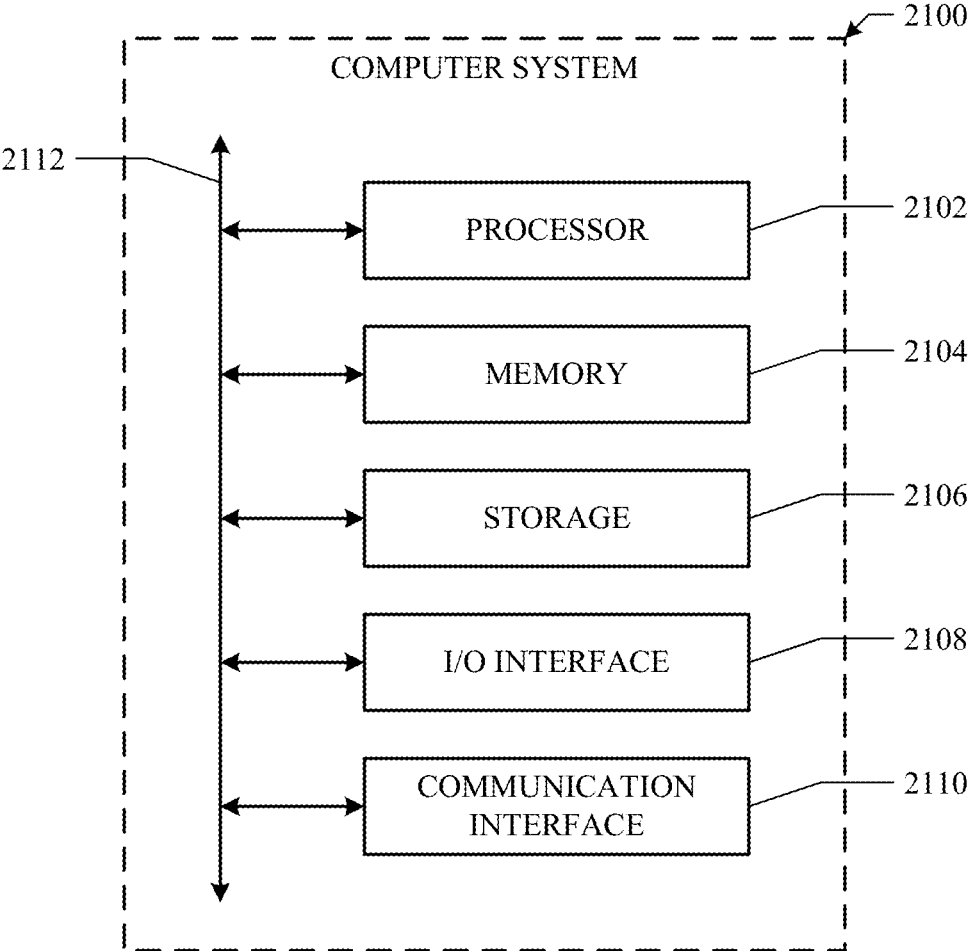


**FIG. 19**

2000



**FIG. 20**



**FIG. 21**

## SYSTEMS AND METHODS FOR IMPLEMENTING SMART ASSISTANT SYSTEMS

### PRIORITY

**[0001]** This application claims the benefit under 35 U.S.C. § 119(e) of U.S. Provisional Patent Application No. 63/376,887, filed 23 Sep. 2022, U.S. Provisional Patent Application No. 63/516,019, filed 27 Jul. 2023, and U.S. Provisional Patent Application No. 63/516,948, filed 1 Aug. 2023, each of which is incorporated herein by reference.

### TECHNICAL FIELD

**[0002]** This disclosure generally relates to databases and file management within network environments, and in particular relates to hardware and software for smart assistant systems.

### BACKGROUND

**[0003]** An assistant system can provide information or services on behalf of a user based on a combination of user input, location awareness, and the ability to access information from a variety of online sources (such as weather conditions, traffic congestion, news, stock prices, user schedules, retail prices, etc.). The user input may include text (e.g., online chat), especially in an instant messaging application or other applications, voice, images, motion, or a combination of them. The assistant system may perform concierge-type services (e.g., making dinner reservations, purchasing event tickets, making travel arrangements) or provide information based on the user input. The assistant system may also perform management or data-handling tasks based on online information and events without user initiation or interaction. Examples of those tasks that may be performed by an assistant system may include schedule management (e.g., sending an alert to a dinner date that a user is running late due to traffic conditions, update schedules for both parties, and change the restaurant reservation time). The assistant system may be enabled by the combination of computing devices, application programming interfaces (APIs), and the proliferation of applications on user devices.

**[0004]** A social-networking system, which may include a social-networking website, may enable its users (such as persons or organizations) to interact with it and with each other through it. The social-networking system may, with input from a user, create and store in the social-networking system a user profile associated with the user. The user profile may include demographic information, communication-channel information, and information on personal interests of the user. The social-networking system may also, with input from a user, create and store a record of relationships of the user with other users of the social-networking system, as well as provide services (e.g. profile/news feed posts, photo-sharing, event organization, messaging, games, or advertisements) to facilitate social interaction between or among users.

**[0005]** The social-networking system may send over one or more networks content or messages related to its services to a mobile or other computing device of a user. A user may also install software applications on a mobile or other computing device of the user for accessing a user profile of the user and other data within the social-networking system.

The social-networking system may generate a personalized set of content objects to display to a user, such as a newsfeed of aggregated stories of other users connected to the user.

### SUMMARY OF PARTICULAR EMBODIMENTS

**[0006]** In particular embodiments, the assistant system may assist a user to obtain information or services. The assistant system may enable the user to interact with the assistant system via user inputs of various modalities (e.g., audio, voice, text, image, video, gesture, motion, location, orientation) in stateful and multi-turn conversations to receive assistance from the assistant system. As an example and not by way of limitation, the assistant system may support mono-modal inputs (e.g., only voice inputs), multi-modal inputs (e.g., voice inputs and text inputs), hybrid/multi-modal inputs, or any combination thereof. User inputs provided by a user may be associated with particular assistant-related tasks, and may include, for example, user requests (e.g., verbal requests for information or performance of an action), user interactions with an assistant application associated with the assistant system (e.g., selection of UI elements via touch or gesture), or any other type of suitable user input that may be detected and understood by the assistant system (e.g., user movements detected by the client device of the user). The assistant system may create and store a user profile comprising both personal and contextual information associated with the user. In particular embodiments, the assistant system may analyze the user input using natural-language understanding (NLU). The analysis may be based on the user profile of the user for more personalized and context-aware understanding. The assistant system may resolve entities associated with the user input based on the analysis. In particular embodiments, the assistant system may interact with different agents to obtain information or services that are associated with the resolved entities. The assistant system may generate a response for the user regarding the information or services by using natural-language generation (NLG). Through the interaction with the user, the assistant system may use dialog-management techniques to manage and advance the conversation flow with the user. In particular embodiments, the assistant system may further assist the user to effectively and efficiently digest the obtained information by summarizing the information. The assistant system may also assist the user to be more engaging with an online social network by providing tools that help the user interact with the online social network (e.g., creating posts, comments, messages). The assistant system may additionally assist the user to manage different tasks such as keeping track of events. In particular embodiments, the assistant system may proactively execute, without a user input, tasks that are relevant to user interests and preferences based on the user profile, at a time relevant for the user. In particular embodiments, the assistant system may check privacy settings to ensure that accessing a user's profile or other user information and executing different tasks are permitted subject to the user's privacy settings.

**[0007]** In particular embodiments, the assistant system may assist the user via a hybrid architecture built upon both client-side processes and server-side processes. The client-side processes and the server-side processes may be two parallel workflows for processing a user input and providing assistance to the user. In particular embodiments, the client-side processes may be performed locally on a client system associated with a user. By contrast, the server-side processes

may be performed remotely on one or more computing systems. In particular embodiments, an arbitrator on the client system may coordinate receiving user input (e.g., an audio signal), determine whether to use a client-side process, a server-side process, or both, to respond to the user input, and analyze the processing results from each process. The arbitrator may instruct agents on the client-side or server-side to execute tasks associated with the user input based on the aforementioned analyses. The execution results may be further rendered as output to the client system. By leveraging both client-side and server-side processes, the assistant system can effectively assist a user with optimal usage of computing resources while at the same time protecting user privacy and enhancing security.

[0008] The embodiments disclosed herein are only examples, and the scope of this disclosure is not limited to them. Particular embodiments may include all, some, or none of the components, elements, features, functions, operations, or steps of the embodiments disclosed herein. Embodiments according to the invention are in particular disclosed in the attached claims directed to a method, a storage medium, a system and a computer program product, wherein any feature mentioned in one claim category, e.g. method, can be claimed in another claim category, e.g. system, as well. The dependencies or references back in the attached claims are chosen for formal reasons only. However any subject matter resulting from a deliberate reference back to any previous claims (in particular multiple dependencies) can be claimed as well, so that any combination of claims and the features thereof are disclosed and can be claimed regardless of the dependencies chosen in the attached claims. The subject-matter which can be claimed comprises not only the combinations of features as set out in the attached claims but also any other combination of features in the claims, wherein each feature mentioned in the claims can be combined with any other feature or combination of other features in the claims. Furthermore, any of the embodiments and features described or depicted herein can be claimed in a separate claim and/or in any combination with any embodiment or feature described or depicted herein or with any of the features of the attached claims.

#### BRIEF DESCRIPTION OF THE DRAWINGS

[0009] FIG. 1 illustrates an example network environment associated with an assistant system.

[0010] FIG. 2 illustrates an example architecture of the assistant system.

[0011] FIG. 3 illustrates an example flow diagram of the assistant system.

[0012] FIG. 4 illustrates an example task-centric flow diagram of processing a user input.

[0013] FIGS. 5A-5C illustrate an example sequence diagram for a “brightness” setting.

[0014] FIG. 6 illustrates an example sequence diagram for a “brightness” setting on the client side.

[0015] FIG. 7A illustrates an example flow diagram for NLU semantic predictions on a VR headset.

[0016] FIG. 7B illustrates an example flow diagram for NLU semantic predictions on a tablet.

[0017] FIG. 8A illustrates an example flow diagram for modeled device context.

[0018] FIG. 8B illustrates an example flow diagram for intent handler transform.

[0019] FIG. 9 illustrates an example flow diagram for providing parallel legacy and dynamic settings intents for “pump up the volume to 5.”

[0020] FIG. 10A illustrates an example artificial reality (AR) system.

[0021] FIG. 10B illustrates an example augmented reality (AR) system.

[0022] FIGS. 11A-11B illustrates an example flow diagram of processing an audio input.

[0023] FIG. 12 illustrates an example flow diagram of processing an audio input.

[0024] FIG. 13 illustrates an example architecture of a system to process a user input.

[0025] FIG. 14 illustrates example indications of the attention state of an object.

[0026] FIGS. 15A-15B illustrates an example extended reality (XR) environment containing AI voice-driven interactions.

[0027] FIGS. 16A-16B illustrates another example extended reality (XR) environment containing AI voice-driven interactions.

[0028] FIG. 17 illustrates an example method for implementing AI-driven dialog

[0029] FIG. 18 illustrates an example social graph.

[0030] FIG. 19 illustrates an example view of an embedding space.

[0031] FIG. 20 illustrates an example artificial neural network.

[0032] FIG. 21 illustrates an example computer system.

#### DESCRIPTION OF EXAMPLE EMBODIMENTS

##### System Overview

[0033] FIG. 1 illustrates an example network environment 100 associated with an assistant system. Network environment 100 includes a client system 130, an assistant system 140, a social-networking system 160, and a third-party system 170 connected to each other by a network 110. Although FIG. 1 illustrates a particular arrangement of a client system 130, an assistant system 140, a social-networking system 160, a third-party system 170, and a network 110, this disclosure contemplates any suitable arrangement of a client system 130, an assistant system 140, a social-networking system 160, a third-party system 170, and a network 110. As an example and not by way of limitation, two or more of a client system 130, a social-networking system 160, an assistant system 140, and a third-party system 170 may be connected to each other directly, bypassing a network 110. As another example, two or more of a client system 130, an assistant system 140, a social-networking system 160, and a third-party system 170 may be physically or logically co-located with each other in whole or in part. Moreover, although FIG. 1 illustrates a particular number of client systems 130, assistant systems 140, social-networking systems 160, third-party systems 170, and networks 110, this disclosure contemplates any suitable number of client systems 130, assistant systems 140, social-networking systems 160, third-party systems 170, and networks 110. As an example and not by way of limitation, network environment 100 may include multiple client systems 130, assistant systems 140, social-networking systems 160, third-party systems 170, and networks 110.

[0034] This disclosure contemplates any suitable network 110. As an example and not by way of limitation, one or

more portions of a network **110** may include an ad hoc network, an intranet, an extranet, a virtual private network (VPN), a local area network (LAN), a wireless LAN (WLAN), a wide area network (WAN), a wireless WAN (WWAN), a metropolitan area network (MAN), a portion of the Internet, a portion of the Public Switched Telephone Network (PSTN), a cellular technology-based network, a satellite communications technology-based network, another network **110**, or a combination of two or more such networks **110**.

[0035] Links **150** may connect a client system **130**, an assistant system **140**, a social-networking system **160**, and a third-party system **170** to a communication network **110** or to each other. This disclosure contemplates any suitable links **150**. In particular embodiments, one or more links **150** include one or more wireline (such as for example Digital Subscriber Line (DSL) or Data Over Cable Service Interface Specification (DOCSIS)), wireless (such as for example Wi-Fi or Worldwide Interoperability for Microwave Access (WiMAX)), or optical (such as for example Synchronous Optical Network (SONET) or Synchronous Digital Hierarchy (SDH)) links. In particular embodiments, one or more links **150** each include an ad hoc network, an intranet, an extranet, a VPN, a LAN, a WLAN, a WAN, a WWAN, a MAN, a portion of the Internet, a portion of the PSTN, a cellular technology-based network, a satellite communications technology-based network, another link **150**, or a combination of two or more such links **150**. Links **150** need not necessarily be the same throughout a network environment **100**. One or more first links **150** may differ in one or more respects from one or more second links **150**.

[0036] In particular embodiments, a client system **130** may be any suitable electronic device including hardware, software, or embedded logic components, or a combination of two or more such components, and may be capable of carrying out the functionalities implemented or supported by a client system **130**. As an example and not by way of limitation, the client system **130** may include a computer system such as a desktop computer, notebook or laptop computer, netbook, a tablet computer, e-book reader, GPS device, camera, personal digital assistant (PDA), handheld electronic device, cellular telephone, smartphone, smart speaker, smart watch, smart glasses, augmented-reality (AR) smart glasses, virtual reality (VR) headset, other suitable electronic device, or any suitable combination thereof. In particular embodiments, the client system **130** may be a smart assistant device. More information on smart assistant devices may be found in U.S. patent application Ser. No. 15/949,011, filed 9 Apr. 2018, U.S. patent application Ser. No. 16/153,574, filed 5 Oct. 2018, U.S. Design patent application Ser. No. 29/631,910, filed 3 Jan. 2018, U.S. Design patent application Ser. No. 29/631,747, filed 2 Jan. 2018, U.S. Design patent application Ser. No. 29/631,913, filed 3 Jan. 2018, and U.S. Design patent application Ser. No. 29/631,914, filed 3 Jan. 2018, each of which is incorporated by reference. This disclosure contemplates any suitable client systems **130**. In particular embodiments, a client system **130** may enable a network user at a client system **130** to access a network **110**. The client system **130** may also enable the user to communicate with other users at other client systems **130**.

[0037] In particular embodiments, a client system **130** may include a web browser **132**, and may have one or more add-ons, plug-ins, or other extensions. A user at a client

system **130** may enter a Uniform Resource Locator (URL) or other address directing a web browser **132** to a particular server (such as server **162**, or a server associated with a third-party system **170**), and the web browser **132** may generate a Hyper Text Transfer Protocol (HTTP) request and communicate the HTTP request to server. The server may accept the HTTP request and communicate to a client system **130** one or more Hyper Text Markup Language (HTML) files responsive to the HTTP request. The client system **130** may render a web interface (e.g. a webpage) based on the HTML files from the server for presentation to the user. This disclosure contemplates any suitable source files. As an example and not by way of limitation, a web interface may be rendered from HTML files, Extensible Hyper Text Markup Language (XHTML) files, or Extensible Markup Language (XML) files, according to particular needs. Such interfaces may also execute scripts, combinations of markup language and scripts, and the like. Herein, reference to a web interface encompasses one or more corresponding source files (which a browser may use to render the web interface) and vice versa, where appropriate.

[0038] In particular embodiments, a client system **130** may include a social-networking application **134** installed on the client system **130**. A user at a client system **130** may use the social-networking application **134** to access an online social network. The user at the client system **130** may use the social-networking application **134** to communicate with the user's social connections (e.g., friends, followers, followed accounts, contacts, etc.). The user at the client system **130** may also use the social-networking application **134** to interact with a plurality of content objects (e.g., posts, news articles, ephemeral content, etc.) on the online social network. As an example and not by way of limitation, the user may browse trending topics and breaking news using the social-networking application **134**.

[0039] In particular embodiments, a client system **130** may include an assistant application **136**. A user at a client system **130** may use the assistant application **136** to interact with the assistant system **140**. In particular embodiments, the assistant application **136** may include an assistant xbot functionality as a front-end interface for interacting with the user of the client system **130**, including receiving user inputs and presenting outputs. In particular embodiments, the assistant application **136** may comprise a stand-alone application. In particular embodiments, the assistant application **136** may be integrated into the social-networking application **134** or another suitable application (e.g., a messaging application). In particular embodiments, the assistant application **136** may be also integrated into the client system **130**, an assistant hardware device, or any other suitable hardware devices. In particular embodiments, the assistant application **136** may be also part of the assistant system **140**. In particular embodiments, the assistant application **136** may be accessed via the web browser **132**. In particular embodiments, the user may interact with the assistant system **140** by providing user input to the assistant application **136** via various modalities (e.g., audio, voice, text, vision, image, video, gesture, motion, activity, location, orientation). The assistant application **136** may communicate the user input to the assistant system **140** (e.g., via the assistant xbot). Based on the user input, the assistant system **140** may generate responses. The assistant system **140** may send the generated responses to the assistant application **136**. The assistant application **136** may then present the responses to the user

at the client system 130 via various modalities (e.g., audio, text, image, and video). As an example and not by way of limitation, the user may interact with the assistant system 140 by providing a user input (e.g., a verbal request for information regarding a current status of nearby vehicle traffic) to the assistant xbot via a microphone of the client system 130. The assistant application 136 may then communicate the user input to the assistant system 140 over network 110. The assistant system 140 may accordingly analyze the user input, generate a response based on the analysis of the user input (e.g., vehicle traffic information obtained from a third-party source), and communicate the generated response back to the assistant application 136. The assistant application 136 may then present the generated response to the user in any suitable manner (e.g., displaying a text-based push notification and/or image(s) illustrating a local map of nearby vehicle traffic on a display of the client system 130).

[0040] In particular embodiments, a client system 130 may implement wake-word detection techniques to allow users to conveniently activate the assistant system 140 using one or more wake-words associated with assistant system 140. As an example and not by way of limitation, the system audio API on client system 130 may continuously monitor user input comprising audio data (e.g., frames of voice data) received at the client system 130. In this example, a wake-word associated with the assistant system 140 may be the voice phrase “hey assistant.” In this example, when the system audio API on client system 130 detects the voice phrase “hey assistant” in the monitored audio data, the assistant system 140 may be activated for subsequent interaction with the user. In alternative embodiments, similar detection techniques may be implemented to activate the assistant system 140 using particular non-audio user inputs associated with the assistant system 140. For example, the non-audio user inputs may be specific visual signals detected by a low-power sensor (e.g., camera) of client system 130. As an example and not by way of limitation, the visual signals may be a static image (e.g., barcode, QR code, universal product code (UPC)), a position of the user (e.g., the user’s gaze towards client system 130), a user motion (e.g., the user pointing at an object), or any other suitable visual signal.

[0041] In particular embodiments, a client system 130 may include a rendering device 137 and, optionally, a companion device 138. The rendering device 137 may be configured to render outputs generated by the assistant system 140 to the user. The companion device 138 may be configured to perform computations associated with particular tasks (e.g., communications with the assistant system 140) locally (i.e., on-device) on the companion device 138 in particular circumstances (e.g., when the rendering device 137 is unable to perform said computations). In particular embodiments, the client system 130, the rendering device 137, and/or the companion device 138 may each be a suitable electronic device including hardware, software, or embedded logic components, or a combination of two or more such components, and may be capable of carrying out, individually or cooperatively, the functionalities implemented or supported by the client system 130 described herein. As an example and not by way of limitation, the client system 130, the rendering device 137, and/or the companion device 138 may each include a computer system such as a desktop computer, notebook or laptop computer,

netbook, a tablet computer, e-book reader, GPS device, camera, personal digital assistant (PDA), handheld electronic device, cellular telephone, smartphone, smart speaker, virtual reality (VR) headset, augmented-reality (AR) smart glasses, other suitable electronic device, or any suitable combination thereof. In particular embodiments, one or more of the client system 130, the rendering device 137, and the companion device 138 may operate as a smart assistant device. As an example and not by way of limitation, the rendering device 137 may comprise smart glasses and the companion device 138 may comprise a smart phone. As another example and not by way of limitation, the rendering device 137 may comprise a smart watch and the companion device 138 may comprise a smart phone. As yet another example and not by way of limitation, the rendering device 137 may comprise smart glasses and the companion device 138 may comprise a smart remote for the smart glasses. As yet another example and not by way of limitation, the rendering device 137 may comprise a VR/AR headset and the companion device 138 may comprise a smart phone.

[0042] In particular embodiments, a user may interact with the assistant system 140 using the rendering device 137 or the companion device 138, individually or in combination. In particular embodiments, one or more of the client system 130, the rendering device 137, and the companion device 138 may implement a multi-stage wake-word detection model to enable users to conveniently activate the assistant system 140 by continuously monitoring for one or more wake-words associated with assistant system 140. At a first stage of the wake-word detection model, the rendering device 137 may receive audio user input (e.g., frames of voice data). If a wireless connection between the rendering device 137 and the companion device 138 is available, the application on the rendering device 137 may communicate the received audio user input to the companion application on the companion device 138 via the wireless connection. At a second stage of the wake-word detection model, the companion application on the companion device 138 may process the received audio user input to detect a wake-word associated with the assistant system 140. The companion application on the companion device 138 may then communicate the detected wake-word to a server associated with the assistant system 140 via wireless network 110. At a third stage of the wake-word detection model, the server associated with the assistant system 140 may perform a keyword verification on the detected wake-word to verify whether the user intended to activate and receive assistance from the assistant system 140. In alternative embodiments, any of the processing, detection, or keyword verification may be performed by the rendering device 137 and/or the companion device 138. In particular embodiments, when the assistant system 140 has been activated by the user, an application on the rendering device 137 may be configured to receive user input from the user, and a companion application on the companion device 138 may be configured to handle user inputs (e.g., user requests) received by the application on the rendering device 137. In particular embodiments, the rendering device 137 and the companion device 138 may be associated with each other (i.e., paired) via one or more wireless communication protocols (e.g., Bluetooth).

[0043] The following example workflow illustrates how a rendering device 137 and a companion device 138 may handle a user input provided by a user. In this example, an application on the rendering device 137 may receive a user



input comprising a user request directed to the rendering device 137. The application on the rendering device 137 may then determine a status of a wireless connection (i.e., tethering status) between the rendering device 137 and the companion device 138. If a wireless connection between the rendering device 137 and the companion device 138 is not available, the application on the rendering device 137 may communicate the user request (optionally including additional data and/or contextual information available to the rendering device 137) to the assistant system 140 via the network 110. The assistant system 140 may then generate a response to the user request and communicate the generated response back to the rendering device 137. The rendering device 137 may then present the response to the user in any suitable manner. Alternatively, if a wireless connection between the rendering device 137 and the companion device 138 is available, the application on the rendering device 137 may communicate the user request (optionally including additional data and/or contextual information available to the rendering device 137) to the companion application on the companion device 138 via the wireless connection. The companion application on the companion device 138 may then communicate the user request (optionally including additional data and/or contextual information available to the companion device 138) to the assistant system 140 via the network 110. The assistant system 140 may then generate a response to the user request and communicate the generated response back to the companion device 138. The companion application on the companion device 138 may then communicate the generated response to the application on the rendering device 137. The rendering device 137 may then present the response to the user in any suitable manner. In the preceding example workflow, the rendering device 137 and the companion device 138 may each perform one or more computations and/or processes at each respective step of the workflow. In particular embodiments, performance of the computations and/or processes disclosed herein may be adaptively switched between the rendering device 137 and the companion device 138 based at least in part on a device state of the rendering device 137 and/or the companion device 138, a task associated with the user input, and/or one or more additional factors. As an example and not by way of limitation, one factor may be signal strength of the wireless connection between the rendering device 137 and the companion device 138. For example, if the signal strength of the wireless connection between the rendering device 137 and the companion device 138 is strong, the computations and processes may be adaptively switched to be substantially performed by the companion device 138 in order to, for example, benefit from the greater processing power of the CPU of the companion device 138. Alternatively, if the signal strength of the wireless connection between the rendering device 137 and the companion device 138 is weak, the computations and processes may be adaptively switched to be substantially performed by the rendering device 137 in a standalone manner. In particular embodiments, if the client system 130 does not comprise a companion device 138, the aforementioned computations and processes may be performed solely by the rendering device 137 in a standalone manner.

[0044] In particular embodiments, an assistant system 140 may assist users with various assistant-related tasks. The assistant system 140 may interact with the social-networking

system 160 and/or the third-party system 170 when executing these assistant-related tasks.

[0045] In particular embodiments, the social-networking system 160 may be a network-addressable computing system that can host an online social network. The social-networking system 160 may generate, store, receive, and send social-networking data, such as, for example, user profile data, concept-profile data, social-graph information, or other suitable data related to the online social network. The social-networking system 160 may be accessed by the other components of network environment 100 either directly or via a network 110. As an example and not by way of limitation, a client system 130 may access the social-networking system 160 using a web browser 132 or a native application associated with the social-networking system 160 (e.g., a mobile social-networking application, a messaging application, another suitable application, or any combination thereof) either directly or via a network 110. In particular embodiments, the social-networking system 160 may include one or more servers 162. Each server 162 may be a unitary server or a distributed server spanning multiple computers or multiple datacenters. As an example and not by way of limitation, each server 162 may be a web server, a news server, a mail server, a message server, an advertising server, a file server, an application server, an exchange server, a database server, a proxy server, another server suitable for performing functions or processes described herein, or any combination thereof. In particular embodiments, each server 162 may include hardware, software, or embedded logic components or a combination of two or more such components for carrying out the appropriate functionalities implemented or supported by server 162. In particular embodiments, the social-networking system 160 may include one or more data stores 164. Data stores 164 may be used to store various types of information. In particular embodiments, the information stored in data stores 164 may be organized according to specific data structures. In particular embodiments, each data store 164 may be a relational, columnar, correlation, or other suitable database. Although this disclosure describes or illustrates particular types of databases, this disclosure contemplates any suitable types of databases. Particular embodiments may provide interfaces that enable a client system 130, a social-networking system 160, an assistant system 140, or a third-party system 170 to manage, retrieve, modify, add, or delete, the information stored in data store 164.

[0046] In particular embodiments, the social-networking system 160 may store one or more social graphs in one or more data stores 164. In particular embodiments, a social graph may include multiple nodes—which may include multiple user nodes (each corresponding to a particular user) or multiple concept nodes (each corresponding to a particular concept)—and multiple edges connecting the nodes. The social-networking system 160 may provide users of the online social network the ability to communicate and interact with other users. In particular embodiments, users may join the online social network via the social-networking system 160 and then add connections (e.g., relationships) to a number of other users of the social-networking system 160 whom they want to be connected to. Herein, the term “friend” may refer to any other user of the social-networking system 160 with whom a user has formed a connection, association, or relationship via the social-networking system 160.

[0047] In particular embodiments, the social-networking system 160 may provide users with the ability to take actions on various types of items or objects, supported by the social-networking system 160. As an example and not by way of limitation, the items and objects may include groups or social networks to which users of the social-networking system 160 may belong, events or calendar entries in which a user might be interested, computer-based applications that a user may use, transactions that allow users to buy or sell items via the service, interactions with advertisements that a user may perform, or other suitable items or objects. A user may interact with anything that is capable of being represented in the social-networking system 160 or by an external system of a third-party system 170, which is separate from the social-networking system 160 and coupled to the social-networking system 160 via a network 110.

[0048] In particular embodiments, the social-networking system 160 may be capable of linking a variety of entities. As an example and not by way of limitation, the social-networking system 160 may enable users to interact with each other as well as receive content from third-party systems 170 or other entities, or to allow users to interact with these entities through an application programming interfaces (API) or other communication channels.

[0049] In particular embodiments, a third-party system 170 may include one or more types of servers, one or more data stores, one or more interfaces, including but not limited to APIs, one or more web services, one or more content sources, one or more networks, or any other suitable components, e.g., that servers may communicate with. A third-party system 170 may be operated by a different entity from an entity operating the social-networking system 160. In particular embodiments, however, the social-networking system 160 and third-party systems 170 may operate in conjunction with each other to provide social-networking services to users of the social-networking system 160 or third-party systems 170. In this sense, the social-networking system 160 may provide a platform, or backbone, which other systems, such as third-party systems 170, may use to provide social-networking services and functionality to users across the Internet.

[0050] In particular embodiments, a third-party system 170 may include a third-party content object provider. A third-party content object provider may include one or more sources of content objects, which may be communicated to a client system 130. As an example and not by way of limitation, content objects may include information regarding things or activities of interest to the user, such as, for example, movie show times, movie reviews, restaurant reviews, restaurant menus, product information and reviews, or other suitable information. As another example and not by way of limitation, content objects may include incentive content objects, such as coupons, discount tickets, gift certificates, or other suitable incentive objects. In particular embodiments, a third-party content provider may use one or more third-party agents to provide content objects and/or services. A third-party agent may be an implementation that is hosted and executing on the third-party system 170.

[0051] In particular embodiments, the social-networking system 160 also includes user-generated content objects, which may enhance a user's interactions with the social-networking system 160. User-generated content may include anything a user can add, upload, send, or "post" to the social-networking system 160. As an example and not by

way of limitation, a user communicates posts to the social-networking system 160 from a client system 130. Posts may include data such as status updates or other textual data, location information, photos, videos, links, music or other similar data or media. Content may also be added to the social-networking system 160 by a third-party through a "communication channel," such as a newsfeed or stream.

[0052] In particular embodiments, the social-networking system 160 may include a variety of servers, sub-systems, programs, modules, logs, and data stores. In particular embodiments, the social-networking system 160 may include one or more of the following: a web server, action logger, API-request server, relevance-and-ranking engine, content-object classifier, notification controller, action log, third-party-content-object-exposure log, inference module, authorization/privacy server, search module, advertisement-targeting module, user-interface module, user-profile store, connection store, third-party content store, or location store. The social-networking system 160 may also include suitable components such as network interfaces, security mechanisms, load balancers, failover servers, management-and-network-operations consoles, other suitable components, or any suitable combination thereof. In particular embodiments, the social-networking system 160 may include one or more user-profile stores for storing user profiles. A user profile may include, for example, biographic information, demographic information, behavioral information, social information, or other types of descriptive information, such as work experience, educational history, hobbies or preferences, interests, affinities, or location. Interest information may include interests related to one or more categories. Categories may be general or specific. As an example and not by way of limitation, if a user "likes" an article about a brand of shoes the category may be the brand, or the general category of "shoes" or "clothing." A connection store may be used for storing connection information about users. The connection information may indicate users who have similar or common work experience, group memberships, hobbies, educational history, or are in any way related or share common attributes. The connection information may also include user-defined connections between different users and content (both internal and external). A web server may be used for linking the social-networking system 160 to one or more client systems 130 or one or more third-party systems 170 via a network 110. The web server may include a mail server or other messaging functionality for receiving and routing messages between the social-networking system 160 and one or more client systems 130. An API-request server may allow, for example, an assistant system 140 or a third-party system 170 to access information from the social-networking system 160 by calling one or more APIs. An action logger may be used to receive communications from a web server about a user's actions on or off the social-networking system 160. In conjunction with the action log, a third-party-content-object log may be maintained of user exposures to third-party-content objects. A notification controller may provide information regarding content objects to a client system 130. Information may be pushed to a client system 130 as notifications, or information may be pulled from a client system 130 responsive to a user input comprising a user request received from a client system 130. Authorization servers may be used to enforce one or more privacy settings of the users of the social-networking system 160. A privacy setting of a user may

determine how particular information associated with a user can be shared. The authorization server may allow users to opt in to or opt out of having their actions logged by the social-networking system 160 or shared with other systems (e.g., a third-party system 170), such as, for example, by setting appropriate privacy settings. Third-party-content-object stores may be used to store content objects received from third parties, such as a third-party system 170. Location stores may be used for storing location information received from client systems 130 associated with users. Advertisement-pricing modules may combine social information, the current time, location information, or other suitable information to provide relevant advertisements, in the form of notifications, to a user.

#### Assistant Systems

[0053] FIG. 2 illustrates an example architecture 200 of the assistant system 140. In particular embodiments, the assistant system 140 may assist a user to obtain information or services. The assistant system 140 may enable the user to interact with the assistant system 140 via user inputs of various modalities (e.g., audio, voice, text, vision, image, video, gesture, motion, activity, location, orientation) in stateful and multi-turn conversations to receive assistance from the assistant system 140. As an example and not by way of limitation, a user input may comprise an audio input based on the user's voice (e.g., a verbal command), which may be processed by a system audio API (application programming interface) on client system 130. The system audio API may perform techniques including echo cancellation, noise removal, beam forming, self-user voice activation, speaker identification, voice activity detection (VAD), and/or any other suitable acoustic technique in order to generate audio data that is readily processable by the assistant system 140. In particular embodiments, the assistant system 140 may support mono-modal inputs (e.g., only voice inputs), multi-modal inputs (e.g., voice inputs and text inputs), hybrid/multi-modal inputs, or any combination thereof. In particular embodiments, a user input may be a user-generated input that is sent to the assistant system 140 in a single turn. User inputs provided by a user may be associated with particular assistant-related tasks, and may include, for example, user requests (e.g., verbal requests for information or performance of an action), user interactions with the assistant application 136 associated with the assistant system 140 (e.g., selection of UI elements via touch or gesture), or any other type of suitable user input that may be detected and understood by the assistant system 140 (e.g., user movements detected by the client device 130 of the user).

[0054] In particular embodiments, the assistant system 140 may create and store a user profile comprising both personal and contextual information associated with the user. In particular embodiments, the assistant system 140 may analyze the user input using natural-language understanding (NLU) techniques. The analysis may be based at least in part on the user profile of the user for more personalized and context-aware understanding. The assistant system 140 may resolve entities associated with the user input based on the analysis. In particular embodiments, the assistant system 140 may interact with different agents to obtain information or services that are associated with the resolved entities. The assistant system 140 may generate a response for the user regarding the information or services

by using natural-language generation (NLG). Through the interaction with the user, the assistant system 140 may use dialog management techniques to manage and forward the conversation flow with the user. In particular embodiments, the assistant system 140 may further assist the user to effectively and efficiently digest the obtained information by summarizing the information. The assistant system 140 may also assist the user to be more engaging with an online social network by providing tools that help the user interact with the online social network (e.g., creating posts, comments, messages). The assistant system 140 may additionally assist the user to manage different tasks such as keeping track of events. In particular embodiments, the assistant system 140 may proactively execute, without a user input, pre-authorized tasks that are relevant to user interests and preferences based on the user profile, at a time relevant for the user. In particular embodiments, the assistant system 140 may check privacy settings to ensure that accessing a user's profile or other user information and executing different tasks are permitted subject to the user's privacy settings. More information on assisting users subject to privacy settings may be found in U.S. patent application Ser. No. 16/182,542, filed 6 Nov. 2018, which is incorporated by reference.

[0055] In particular embodiments, the assistant system 140 may assist a user via an architecture built upon client-side processes and server-side processes which may operate in various operational modes. In FIG. 2, the client-side process is illustrated above the dashed line 202 whereas the server-side process is illustrated below the dashed line 202. A first operational mode (i.e., on-device mode) may be a workflow in which the assistant system 140 processes a user input and provides assistance to the user by primarily or exclusively performing client-side processes locally on the client system 130. For example, if the client system 130 is not connected to a network 110 (i.e., when client system 130 is offline), the assistant system 140 may handle a user input in the first operational mode utilizing only client-side processes. A second operational mode (i.e., cloud mode) may be a workflow in which the assistant system 140 processes a user input and provides assistance to the user by primarily or exclusively performing server-side processes on one or more remote servers (e.g., a server associated with assistant system 140). As illustrated in FIG. 2, a third operational mode (i.e., blended mode) may be a parallel workflow in which the assistant system 140 processes a user input and provides assistance to the user by performing client-side processes locally on the client system 130 in conjunction with server-side processes on one or more remote servers (e.g., a server associated with assistant system 140). For example, the client system 130 and the server associated with assistant system 140 may both perform automatic speech recognition (ASR) and natural-language understanding (NLU) processes, but the client system 130 may delegate dialog, agent, and natural-language generation (NLG) processes to be performed by the server associated with assistant system 140.

[0056] In particular embodiments, selection of an operational mode may be based at least in part on a device state, a task associated with a user input, and/or one or more additional factors. As an example and not by way of limitation, as described above, one factor may be a network connectivity status for client system 130. For example, if the client system 130 is not connected to a network 110 (i.e., when client system 130 is offline), the assistant system 140

may handle a user input in the first operational mode (i.e., on-device mode). As another example and not by way of limitation, another factor may be based on a measure of available battery power (i.e., battery status) for the client system 130. For example, if there is a need for client system 130 to conserve battery power (e.g., when client system 130 has minimal available battery power or the user has indicated a desire to conserve the battery power of the client system 130), the assistant system 140 may handle a user input in the second operational mode (i.e., cloud mode) or the third operational mode (i.e., blended mode) in order to perform fewer power-intensive operations on the client system 130. As yet another example and not by way of limitation, another factor may be one or more privacy constraints (e.g., specified privacy settings, applicable privacy policies). For example, if one or more privacy constraints limits or precludes particular data from being transmitted to a remote server (e.g., a server associated with the assistant system 140), the assistant system 140 may handle a user input in the first operational mode (i.e., on-device mode) in order to protect user privacy. As yet another example and not by way of limitation, another factor may be desynchronized context data between the client system 130 and a remote server (e.g., the server associated with assistant system 140). For example, the client system 130 and the server associated with assistant system 140 may be determined to have inconsistent, missing, and/or unreconciled context data, the assistant system 140 may handle a user input in the third operational mode (i.e., blended mode) to reduce the likelihood of an inadequate analysis associated with the user input. As yet another example and not by way of limitation, another factor may be a measure of latency for the connection between client system 130 and a remote server (e.g., the server associated with assistant system 140). For example, if a task associated with a user input may significantly benefit from and/or require prompt or immediate execution (e.g., photo capturing tasks), the assistant system 140 may handle the user input in the first operational mode (i.e., on-device mode) to ensure the task is performed in a timely manner. As yet another example and not by way of limitation, another factor may be, for a feature relevant to a task associated with a user input, whether the feature is only supported by a remote server (e.g., the server associated with assistant system 140). For example, if the relevant feature requires advanced technical functionality (e.g., high-powered processing capabilities, rapid update cycles) that is only supported by the server associated with assistant system 140 and is not supported by client system 130 at the time of the user input, the assistant system 140 may handle the user input in the second operational mode (i.e., cloud mode) or the third operational mode (i.e., blended mode) in order to benefit from the relevant feature.

**[0057]** In particular embodiments, an on-device orchestrator 206 on the client system 130 may coordinate receiving a user input and may determine, at one or more decision points in an example workflow, which of the operational modes described above should be used to process or continue processing the user input. As discussed above, selection of an operational mode may be based at least in part on a device state, a task associated with a user input, and/or one or more additional factors. As an example and not by way of limitation, with reference to the workflow architecture illustrated in FIG. 2, after a user input is received from a user, the on-device orchestrator 206 may determine, at decision point

(D0) 205, whether to begin processing the user input in the first operational mode (i.e., on-device mode), the second operational mode (i.e., cloud mode), or the third operational mode (i.e., blended mode). For example, at decision point (D0) 205, the on-device orchestrator 206 may select the first operational mode (i.e., on-device mode) if the client system 130 is not connected to network 110 (i.e., when client system 130 is offline), if one or more privacy constraints expressly require on-device processing (e.g., adding or removing another person to a private call between users), or if the user input is associated with a task which does not require or benefit from server-side processing (e.g., setting an alarm or calling another user). As another example, at decision point (D0) 205, the on-device orchestrator 206 may select the second operational mode (i.e., cloud mode) or the third operational mode (i.e., blended mode) if the client system 130 has a need to conserve battery power (e.g., when client system 130 has minimal available battery power or the user has indicated a desire to conserve the battery power of the client system 130) or has a need to limit additional utilization of computing resources (e.g., when other processes operating on client device 130 require high CPU utilization (e.g., SMS messaging applications)).

**[0058]** In particular embodiments, if the on-device orchestrator 206 determines at decision point (D0) 205 that the user input should be processed using the first operational mode (i.e., on-device mode) or the third operational mode (i.e., blended mode), the client-side process may continue as illustrated in FIG. 2. As an example and not by way of limitation, if the user input comprises speech data, the speech data may be received at a local automatic speech recognition (ASR) module 208a on the client system 130. The ASR module 208a may allow a user to dictate and have speech transcribed as written text, have a document synthesized as an audio stream, or issue commands that are recognized as such by the system.

**[0059]** In particular embodiments, the output of the ASR module 208a may be sent to a local natural-language understanding (NLU) module 210a. The NLU module 210a may perform named entity resolution (NER), or named entity resolution may be performed by the entity resolution module 212a, as described below. In particular embodiments, one or more of an intent, a slot, or a domain may be an output of the NLU module 210a.

**[0060]** In particular embodiments, the user input may comprise non-speech data, which may be received at a local context engine 220a. As an example and not by way of limitation, the non-speech data may comprise locations, visuals, touch, gestures, world updates, social updates, contextual information, information related to people, activity data, and/or any other suitable type of non-speech data. The non-speech data may further comprise sensory data received by client system 130 sensors (e.g., microphone, camera), which may be accessed subject to privacy constraints and further analyzed by computer vision technologies. In particular embodiments, the computer vision technologies may comprise object detection, scene recognition, hand tracking, eye tracking, and/or any other suitable computer vision technologies. In particular embodiments, the non-speech data may be subject to geometric constructions, which may comprise constructing objects surrounding a user using any suitable type of data collected by a client system 130. As an example and not by way of limitation, a user may be wearing AR glasses, and geometric constructions may be utilized to

determine spatial locations of surfaces and items (e.g., a floor, a wall, a user's hands). In particular embodiments, the non-speech data may be inertial data captured by AR glasses or a VR headset, and which may be data associated with linear and angular motions (e.g., measurements associated with a user's body movements). In particular embodiments, the context engine 220a may determine various types of events and context based on the non-speech data.

**[0061]** In particular embodiments, the outputs of the NLU module 210a and/or the context engine 220a may be sent to an entity resolution module 212a. The entity resolution module 212a may resolve entities associated with one or more slots output by NLU module 210a. In particular embodiments, each resolved entity may be associated with one or more entity identifiers. As an example and not by way of limitation, an identifier may comprise a unique user identifier (ID) corresponding to a particular user (e.g., a unique username or user ID number for the social-networking system 160). In particular embodiments, each resolved entity may also be associated with a confidence score. More information on resolving entities may be found in U.S. Pat. No. 10,803,050, filed 27 Jul. 2018, and U.S. patent application Ser. No. 16/048,072, filed 27 Jul. 2018, each of which is incorporated by reference.

**[0062]** In particular embodiments, at decision point (D0) 205, the on-device orchestrator 206 may determine that a user input should be handled in the second operational mode (i.e., cloud mode) or the third operational mode (i.e., blended mode). In these operational modes, the user input may be handled by certain server-side modules in a similar manner as the client-side process described above.

**[0063]** In particular embodiments, if the user input comprises speech data, the speech data of the user input may be received at a remote automatic speech recognition (ASR) module 208b on a remote server (e.g., the server associated with assistant system 140). The ASR module 208b may allow a user to dictate and have speech transcribed as written text, have a document synthesized as an audio stream, or issue commands that are recognized as such by the system.

**[0064]** In particular embodiments, the output of the ASR module 208b may be sent to a remote natural-language understanding (NLU) module 210b. In particular embodiments, the NLU module 210b may perform named entity resolution (NER) or named entity resolution may be performed by entity resolution module 212b of dialog manager module 216b as described below. In particular embodiments, one or more of an intent, a slot, or a domain may be an output of the NLU module 210b.

**[0065]** In particular embodiments, the user input may comprise non-speech data, which may be received at a remote context engine 220b. In particular embodiments, the remote context engine 220b may determine various types of events and context based on the non-speech data. In particular embodiments, the output of the NLU module 210b and/or the context engine 220b may be sent to a remote dialog manager 216b.

**[0066]** In particular embodiments, as discussed above, an on-device orchestrator 206 on the client system 130 may coordinate receiving a user input and may determine, at one or more decision points in an example workflow, which of the operational modes described above should be used to process or continue processing the user input. As further discussed above, selection of an operational mode may be based at least in part on a device state, a task associated with

a user input, and/or one or more additional factors. As an example and not by way of limitation, with continued reference to the workflow architecture illustrated in FIG. 2, after the entity resolution module 212a generates an output or a null output, the on-device orchestrator 206 may determine, at decision point (D1) 215, whether to continue processing the user input in the first operational mode (i.e., on-device mode), the second operational mode (i.e., cloud mode), or the third operational mode (i.e., blended mode). For example, at decision point (D1) 215, the on-device orchestrator 206 may select the first operational mode (i.e., on-device mode) if an identified intent is associated with a latency sensitive processing task (e.g., taking a photo, pausing a stopwatch). As another example and not by way of limitation, if a messaging task is not supported by on-device processing on the client system 130, the on-device orchestrator 206 may select the third operational mode (i.e., blended mode) to process the user input associated with a messaging request. As yet another example, at decision point (D1) 215, the on-device orchestrator 206 may select the second operational mode (i.e., cloud mode) or the third operational mode (i.e., blended mode) if the task being processed requires access to a social graph, a knowledge graph, or a concept graph not stored on the client system 130. Alternatively, the on-device orchestrator 206 may instead select the first operational mode (i.e., on-device mode) if a sufficient version of an informational graph including requisite information for the task exists on the client system 130 (e.g., a smaller and/or bootstrapped version of a knowledge graph).

**[0067]** In particular embodiments, if the on-device orchestrator 206 determines at decision point (D1) 215 that processing should continue using the first operational mode (i.e., on-device mode) or the third operational mode (i.e., blended mode), the client-side process may continue as illustrated in FIG. 2. As an example and not by way of limitation, the output from the entity resolution module 212a may be sent to an on-device dialog manager 216a. In particular embodiments, the on-device dialog manager 216a may comprise a dialog state tracker 218a and an action selector 222a. The on-device dialog manager 216a may have complex dialog logic and product-related business logic to manage the dialog state and flow of the conversation between the user and the assistant system 140. The on-device dialog manager 216a may include full functionality for end-to-end integration and multi-turn support (e.g., confirmation, disambiguation). The on-device dialog manager 216a may also be lightweight with respect to computing limitations and resources including memory, computation (CPU), and binary size constraints. The on-device dialog manager 216a may also be scalable to improve developer experience. In particular embodiments, the on-device dialog manager 216a may benefit the assistant system 140, for example, by providing offline support to alleviate network connectivity issues (e.g., unstable or unavailable network connections), by using client-side processes to prevent privacy-sensitive information from being transmitted off of client system 130, and by providing a stable user experience in high-latency sensitive scenarios.

**[0068]** In particular embodiments, the on-device dialog manager 216a may further conduct false trigger mitigation. Implementation of false trigger mitigation may detect and prevent false triggers from user inputs which would otherwise invoke the assistant system 140 (e.g., an unintended

wake-word) and may further prevent the assistant system **140** from generating data records based on the false trigger that may be inaccurate and/or subject to privacy constraints. As an example and not by way of limitation, if a user is in a voice call, the user's conversation during the voice call may be considered private, and the false trigger mitigation may limit detection of wake-words to audio user inputs received locally by the user's client system **130**. In particular embodiments, the on-device dialog manager **216a** may implement false trigger mitigation based on a nonsense detector. If the nonsense detector determines with a high confidence that a received wake-word is not logically and/or contextually sensible at the point in time at which it was received from the user, the on-device dialog manager **216a** may determine that the user did not intend to invoke the assistant system **140**.

[0069] In particular embodiments, due to a limited computing power of the client system **130**, the on-device dialog manager **216a** may conduct on-device learning based on learning algorithms particularly tailored for client system **130**. As an example and not by way of limitation, federated learning techniques may be implemented by the on-device dialog manager **216a**. Federated learning is a specific category of distributed machine learning techniques which may train machine-learning models using decentralized data stored on end devices (e.g., mobile phones). In particular embodiments, the on-device dialog manager **216a** may use federated user representation learning model to extend existing neural-network personalization techniques to implementation of federated learning by the on-device dialog manager **216a**. Federated user representation learning may personalize federated learning models by learning task-specific user representations (i.e., embeddings) and/or by personalizing model weights. Federated user representation learning is a simple, scalable, privacy-preserving, and resource-efficient. Federated user representation learning may divide model parameters into federated and private parameters. Private parameters, such as private user embeddings, may be trained locally on a client system **130** instead of being transferred to or averaged by a remote server (e.g., the server associated with assistant system **140**). Federated parameters, by contrast, may be trained remotely on the server. In particular embodiments, the on-device dialog manager **216a** may use an active federated learning model, which may transmit a global model trained on the remote server to client systems **130** and calculate gradients locally on the client systems **130**. Active federated learning may enable the on-device dialog manager **216a** to minimize the transmission costs associated with downloading models and uploading gradients. For active federated learning, in each round, client systems **130** may be selected in a semi-random manner based at least in part on a probability conditioned on the current model and the data on the client systems **130** in order to optimize efficiency for training the federated learning model.

[0070] In particular embodiments, the dialog state tracker **218a** may track state changes over time as a user interacts with the world and the assistant system **140** interacts with the user. As an example and not by way of limitation, the dialog state tracker **218a** may track, for example, what the user is talking about, whom the user is with, where the user is, what tasks are currently in progress, and where the user's gaze is at subject to applicable privacy policies.

[0071] In particular embodiments, at decision point (D1) **215**, the on-device orchestrator **206** may determine to forward the user input to the server for either the second operational mode (i.e., cloud mode) or the third operational mode (i.e., blended mode). As an example and not by way of limitation, if particular functionalities or processes (e.g., messaging) are not supported by on the client system **130**, the on-device orchestrator **206** may determine at decision point (D1) **215** to use the third operational mode (i.e., blended mode). In particular embodiments, the on-device orchestrator **206** may cause the outputs from the NLU module **210a**, the context engine **220a**, and the entity resolution module **212a**, via a dialog manager proxy **224**, to be forwarded to an entity resolution module **212b** of the remote dialog manager **216b** to continue the processing. The dialog manager proxy **224** may be a communication channel for information/events exchange between the client system **130** and the server. In particular embodiments, the dialog manager **216b** may additionally comprise a remote arbitrator **226b**, a remote dialog state tracker **218b**, and a remote action selector **222b**. In particular embodiments, the assistant system **140** may have started processing a user input with the second operational mode (i.e., cloud mode) at decision point (D0) **205** and the on-device orchestrator **206** may determine to continue processing the user input based on the second operational mode (i.e., cloud mode) at decision point (D1) **215**. Accordingly, the output from the NLU module **210b** and the context engine **220b** may be received at the remote entity resolution module **212b**. The remote entity resolution module **212b** may have similar functionality as the local entity resolution module **212a**, which may comprise resolving entities associated with the slots. In particular embodiments, the entity resolution module **212b** may access one or more of the social graph, the knowledge graph, or the concept graph when resolving the entities. The output from the entity resolution module **212b** may be received at the arbitrator **226b**.

[0072] In particular embodiments, the remote arbitrator **226b** may be responsible for choosing between client-side and server-side upstream results (e.g., results from the NLU module **210a/b**, results from the entity resolution module **212a/b**, and results from the context engine **220a/b**). The arbitrator **226b** may send the selected upstream results to the remote dialog state tracker **218b**. In particular embodiments, similarly to the local dialog state tracker **218a**, the remote dialog state tracker **218b** may convert the upstream results into candidate tasks using task specifications and resolve arguments with entity resolution.

[0073] In particular embodiments, at decision point (D2) **225**, the on-device orchestrator **206** may determine whether to continue processing the user input based on the first operational mode (i.e., on-device mode) or forward the user input to the server for the third operational mode (i.e., blended mode). The decision may depend on, for example, whether the client-side process is able to resolve the task and slots successfully, whether there is a valid task policy with a specific feature support, and/or the context differences between the client-side process and the server-side process. In particular embodiments, decisions made at decision point (D2) **225** may be for multi-turn scenarios. In particular embodiments, there may be at least two possible scenarios. In a first scenario, the assistant system **140** may have started processing a user input in the first operational mode (i.e., on-device mode) using client-side dialog state. If at some

point the assistant system **140** decides to switch to having the remote server process the user input, the assistant system **140** may create a programmatic/predefined task with the current task state and forward it to the remote server. For subsequent turns, the assistant system **140** may continue processing in the third operational mode (i.e., blended mode) using the server-side dialog state. In another scenario, the assistant system **140** may have started processing the user input in either the second operational mode (i.e., cloud mode) or the third operational mode (i.e., blended mode) and may substantially rely on server-side dialog state for all subsequent turns. If the on-device orchestrator **206** determines to continue processing the user input based on the first operational mode (i.e., on-device mode), the output from the dialog state tracker **218a** may be received at the action selector **222a**.

**[0074]** In particular embodiments, at decision point (D2) **225**, the on-device orchestrator **206** may determine to forward the user input to the remote server and continue processing the user input in either the second operational mode (i.e., cloud mode) or the third operational mode (i.e., blended mode). The assistant system **140** may create a programmatic/predefined task with the current task state and forward it to the server, which may be received at the action selector **222b**. In particular embodiments, the assistant system **140** may have started processing the user input in the second operational mode (i.e., cloud mode), and the on-device orchestrator **206** may determine to continue processing the user input in the second operational mode (i.e., cloud mode) at decision point (D2) **225**. Accordingly, the output from the dialog state tracker **218b** may be received at the action selector **222b**.

**[0075]** In particular embodiments, the action selector **222a/b** may perform interaction management. The action selector **222a/b** may determine and trigger a set of general executable actions. The actions may be executed either on the client system **130** or at the remote server. As an example and not by way of limitation, these actions may include providing information or suggestions to the user. In particular embodiments, the actions may interact with agents **228a/b**, users, and/or the assistant system **140** itself. These actions may comprise actions including one or more of a slot request, a confirmation, a disambiguation, or an agent execution. The actions may be independent of the underlying implementation of the action selector **222a/b**. For more complicated scenarios such as, for example, multi-turn tasks or tasks with complex business logic, the local action selector **222a** may call one or more local agents **228a**, and the remote action selector **222b** may call one or more remote agents **228b** to execute the actions. Agents **228a/b** may be invoked via task ID, and any actions may be routed to the correct agent **228a/b** using that task ID. In particular embodiments, an agent **228a/b** may be configured to serve as a broker across a plurality of content providers for one domain. A content provider may be an entity responsible for carrying out an action associated with an intent or completing a task associated with the intent. In particular embodiments, agents **228a/b** may provide several functionalities for the assistant system **140** including, for example, native template generation, task specific business logic, and querying external APIs. When executing actions for a task, agents **228a/b** may use context from the dialog state tracker **218a/b**, and may also update the dialog state tracker **218a/b**.

In particular embodiments, agents **228a/b** may also generate partial payloads from a dialog act.

**[0076]** In particular embodiments, the local agents **228a** may have different implementations to be compiled/registered for different platforms (e.g., smart glasses versus a VR headset). In particular embodiments, multiple device-specific implementations (e.g., real-time calls for a client system **130** or a messaging application on the client system **130**) may be handled internally by a single agent **228a**. Alternatively, device-specific implementations may be handled by multiple agents **228a** associated with multiple domains. As an example and not by way of limitation, calling an agent **228a** on smart glasses may be implemented in a different manner than calling an agent **228a** on a smart phone. Different platforms may also utilize varying numbers of agents **228a**. The agents **228a** may also be cross-platform (i.e., different operating systems on the client system **130**). In addition, the agents **228a** may have minimized startup time or binary size impact. Local agents **228a** may be suitable for particular use cases. As an example and not by way of limitation, one use case may be emergency calling on the client system **130**. As another example and not by way of limitation, another use case may be responding to a user input without network connectivity. As yet another example and not by way of limitation, another use case may be that particular domains/tasks may be privacy sensitive and may prohibit user inputs being sent to the remote server.

**[0077]** In particular embodiments, the local action selector **222a** may call a local delivery system **230a** for executing the actions, and the remote action selector **222b** may call a remote delivery system **230b** for executing the actions. The delivery system **230a/b** may deliver a predefined event upon receiving triggering signals from the dialog state tracker **218a/b** by executing corresponding actions. The delivery system **230a/b** may ensure that events get delivered to a host with a living connection. As an example and not by way of limitation, the delivery system **230a/b** may broadcast to all online devices that belong to one user. As another example and not by way of limitation, the delivery system **230a/b** may deliver events to target-specific devices. The delivery system **230a/b** may further render a payload using up-to-date device context.

**[0078]** In particular embodiments, the on-device dialog manager **216a** may additionally comprise a separate local action execution module, and the remote dialog manager **216b** may additionally comprise a separate remote action execution module. The local execution module and the remote action execution module may have similar functionality. In particular embodiments, the action execution module may call the agents **228a/b** to execute tasks. The action execution module may additionally perform a set of general executable actions determined by the action selector **222a/b**. The set of executable actions may interact with agents **228a/b**, users, and the assistant system **140** itself via the delivery system **230a/b**.

**[0079]** In particular embodiments, if the user input is handled using the first operational mode (i.e., on-device mode), results from the agents **228a** and/or the delivery system **230a** may be returned to the on-device dialog manager **216a**. The on-device dialog manager **216a** may then instruct a local arbitrator **226a** to generate a final response based on these results. The arbitrator **226a** may aggregate the results and evaluate them. As an example and not by way of limitation, the arbitrator **226a** may rank and



select a best result for responding to the user input. If the user request is handled in the second operational mode (i.e., cloud mode), the results from the agents **228b** and/or the delivery system **230b** may be returned to the remote dialog manager **216b**. The remote dialog manager **216b** may instruct, via the dialog manager proxy **224**, the arbitrator **226a** to generate the final response based on these results. Similarly, the arbitrator **226a** may analyze the results and select the best result to provide to the user. If the user input is handled based on the third operational mode (i.e., blended mode), the client-side results and server-side results (e.g., from agents **228a/b** and/or delivery system **230a/b**) may both be provided to the arbitrator **226a** by the on-device dialog manager **216a** and remote dialog manager **216b**, respectively. The arbitrator **226** may then choose between the client-side and server-side side results to determine the final result to be presented to the user. In particular embodiments, the logic to decide between these results may depend on the specific use-case.

**[0080]** In particular embodiments, the local arbitrator **226a** may generate a response based on the final result and send it to a render output module **232**. The render output module **232** may determine how to render the output in a way that is suitable for the client system **130**. As an example and not by way of limitation, for a VR headset or AR smart glasses, the render output module **232** may determine to render the output using a visual-based modality (e.g., an image or a video clip) that may be displayed via the VR headset or AR smart glasses. As another example, the response may be rendered as audio signals that may be played by the user via a VR headset or AR smart glasses. As yet another example, the response may be rendered as augmented-reality data for enhancing user experience.

**[0081]** In particular embodiments, in addition to determining an operational mode to process the user input, the on-device orchestrator **206** may also determine whether to process the user input on the rendering device **137**, process the user input on the companion device **138**, or process the user request on the remote server. The rendering device **137** and/or the companion device **138** may each use the assistant stack in a similar manner as disclosed above to process the user input. As an example and not by, the on-device orchestrator **206** may determine that part of the processing should be done on the rendering device **137**, part of the processing should be done on the companion device **138**, and the remaining processing should be done on the remote server.

**[0082]** In particular embodiments, the assistant system **140** may have a variety of capabilities including audio cognition, visual cognition, signals intelligence, reasoning, and memories. In particular embodiments, the capability of audio cognition may enable the assistant system **140** to, for example, understand a user's input associated with various domains in different languages, understand and summarize a conversation, perform on-device audio cognition for complex commands, identify a user by voice, extract topics from a conversation and auto-tag sections of the conversation, enable audio interaction without a wake-word, filter and amplify user voice from ambient noise and conversations, and/or understand which client system **130** a user is talking to if multiple client systems **130** are in vicinity.

**[0083]** In particular embodiments, the capability of visual cognition may enable the assistant system **140** to, for example, recognize interesting objects in the world through a combination of existing machine-learning models and

one-shot learning, recognize an interesting moment and auto-capture it, achieve semantic understanding over multiple visual frames across different episodes of time, provide platform support for additional capabilities in places or objects recognition, recognize a full set of settings and micro-locations including personalized locations, recognize complex activities, recognize complex gestures to control a client system **130**, handle images/videos from egocentric cameras (e.g., with motion, capture angles, resolution), accomplish similar levels of accuracy and speed regarding images with lower resolution, conduct one-shot registration and recognition of places and objects, and/or perform visual recognition on a client system **130**.

**[0084]** In particular embodiments, the assistant system **140** may leverage computer vision techniques to achieve visual cognition. Besides computer vision techniques, the assistant system **140** may explore options that may supplement these techniques to scale up the recognition of objects. In particular embodiments, the assistant system **140** may use supplemental signals such as, for example, optical character recognition (OCR) of an object's labels, GPS signals for places recognition, and/or signals from a user's client system **130** to identify the user. In particular embodiments, the assistant system **140** may perform general scene recognition (e.g., home, work, public spaces) to set a context for the user and reduce the computer-vision search space to identify likely objects or people. In particular embodiments, the assistant system **140** may guide users to train the assistant system **140**. For example, crowdsourcing may be used to get users to tag objects and help the assistant system **140** recognize more objects over time. As another example, users may register their personal objects as part of an initial setup when using the assistant system **140**. The assistant system **140** may further allow users to provide positive/negative signals for objects they interact with to train and improve personalized models for them.

**[0085]** In particular embodiments, the capability of signals intelligence may enable the assistant system **140** to, for example, determine user location, understand date/time, determine family locations, understand users' calendars and future desired locations, integrate richer sound understanding to identify setting/context through sound alone, and/or build signals intelligence models at runtime which may be personalized to a user's individual routines.

**[0086]** In particular embodiments, the capability of reasoning may enable the assistant system **140** to, for example, pick up previous conversation threads at any point in the future, synthesize all signals to understand micro and personalized context, learn interaction patterns and preferences from users' historical behavior and accurately suggest interactions that they may value, generate highly predictive proactive suggestions based on micro-context understanding, understand what content a user may want to see at what time of a day, and/or understand the changes in a scene and how that may impact the user's desired content.

**[0087]** In particular embodiments, the capabilities of memories may enable the assistant system **140** to, for example, remember which social connections a user previously called or interacted with, write into memory and query memory at will (i.e., open dictation and auto tags), extract richer preferences based on prior interactions and long-term learning, remember a user's life history, extract rich information from egocentric streams of data and auto catalog,



and/or write to memory in structured form to form rich short, episodic and long-term memories.

**[0088]** FIG. 3 illustrates an example flow diagram 300 of the assistant system 140. In particular embodiments, an assistant service module 305 may access a request manager 310 upon receiving a user input. In particular embodiments, the request manager 310 may comprise a context extractor 312 and a conversational understanding object generator (CU object generator) 314. The context extractor 312 may extract contextual information associated with the user input. The context extractor 312 may also update contextual information based on the assistant application 136 executing on the client system 130. As an example and not by way of limitation, the update of contextual information may comprise content items are displayed on the client system 130. As another example and not by way of limitation, the update of contextual information may comprise whether an alarm is set on the client system 130. As another example and not by way of limitation, the update of contextual information may comprise whether a song is playing on the client system 130. The CU object generator 314 may generate particular CU objects relevant to the user input. The CU objects may comprise dialog-session data and features associated with the user input, which may be shared with all the modules of the assistant system 140. In particular embodiments, the request manager 310 may store the contextual information and the generated CU objects in a data store 320 which is a particular data store implemented in the assistant system 140.

**[0089]** In particular embodiments, the request manager 310 may send the generated CU objects to the NLU module 210. The NLU module 210 may perform a plurality of steps to process the CU objects. The NLU module 210 may first run the CU objects through an allowlist/blocklist 330. In particular embodiments, the allowlist/blocklist 330 may comprise interpretation data matching the user input. The NLU module 210 may then perform a featurization 332 of the CU objects. The NLU module 210 may then perform domain classification/selection 334 on user input based on the features resulted from the featurization 332 to classify the user input into predefined domains. In particular embodiments, a domain may denote a social context of interaction (e.g., education), or a namespace for a set of intents (e.g., music). The domain classification/selection results may be further processed based on two related procedures. In one procedure, the NLU module 210 may process the domain classification/selection results using a meta-intent classifier 336a. The meta-intent classifier 336a may determine categories that describe the user's intent. An intent may be an element in a pre-defined taxonomy of semantic intentions, which may indicate a purpose of a user interaction with the assistant system 140. The NLU module 210a may classify a user input into a member of the pre-defined taxonomy. For example, the user input may be "Play Beethoven's 5th," and the NLU module 210a may classify the input as having the intent [IN:play\_music]. In particular embodiments, intents that are common to multiple domains may be processed by the meta-intent classifier 336a. As an example and not by way of limitation, the meta-intent classifier 336a may be based on a machine-learning model that may take the domain classification/selection results as input and calculate a probability of the input being associated with a particular predefined meta-intent. The NLU module 210 may then use a meta slot tagger 338a to annotate one or more meta slots

for the classification result from the meta-intent classifier 336a. A slot may be a named sub-string corresponding to a character string within the user input representing a basic semantic entity. For example, a slot for "pizza" may be [SL:dish]. In particular embodiments, a set of valid or expected named slots may be conditioned on the classified intent. As an example and not by way of limitation, for the intent [IN:play\_music], a valid slot may be [SL:song\_name]. In particular embodiments, the meta slot tagger 338a may tag generic slots such as references to items (e.g., the first), the type of slot, the value of the slot, etc. In particular embodiments, the NLU module 210 may process the domain classification/selection results using an intent classifier 336b. The intent classifier 336b may determine the user's intent associated with the user input. In particular embodiments, there may be one intent classifier 336b for each domain to determine the most possible intents in a given domain. As an example and not by way of limitation, the intent classifier 336b may be based on a machine-learning model that may take the domain classification/selection results as input and calculate a probability of the input being associated with a particular predefined intent. The NLU module 210 may then use a slot tagger 338b to annotate one or more slots associated with the user input. In particular embodiments, the slot tagger 338b may annotate the one or more slots for the n-grams of the user input. As an example and not by way of limitation, a user input may comprise "change 500 dollars in my account to Japanese yen." The intent classifier 336b may take the user input as input and formulate it into a vector. The intent classifier 336b may then calculate probabilities of the user input being associated with different predefined intents based on a vector comparison between the vector representing the user input and the vectors representing different predefined intents. In a similar manner, the slot tagger 338b may take the user input as input and formulate each word into a vector. The slot tagger 338b may then calculate probabilities of each word being associated with different predefined slots based on a vector comparison between the vector representing the word and the vectors representing different predefined slots. The intent of the user may be classified as "changing money". The slots of the user input may comprise "500", "dollars", "account", and "Japanese yen". The meta-intent of the user may be classified as "financial service". The meta slot may comprise "finance".

**[0090]** In particular embodiments, the natural-language understanding (NLU) module 210 may additionally extract information from one or more of a social graph, a knowledge graph, or a concept graph, and may retrieve a user's profile stored locally on the client system 130. The NLU module 210 may additionally consider contextual information when analyzing the user input. The NLU module 210 may further process information from these different sources by identifying and aggregating information, annotating n-grams of the user input, ranking the n-grams with confidence scores based on the aggregated information, and formulating the ranked n-grams into features that may be used by the NLU module 210 for understanding the user input. In particular embodiments, the NLU module 210 may identify one or more of a domain, an intent, or a slot from the user input in a personalized and context-aware manner. As an example and not by way of limitation, a user input may comprise "show me how to get to the coffee shop." The NLU module 210 may identify a particular coffee shop that the user wants

to go to based on the user's personal information and the associated contextual information. In particular embodiments, the NLU module 210 may comprise a lexicon of a particular language, a parser, and grammar rules to partition sentences into an internal representation. The NLU module 210 may also comprise one or more programs that perform naive semantics or stochastic semantic analysis, and may further use pragmatics to understand a user input. In particular embodiments, the parser may be based on a deep learning architecture comprising multiple long-short term memory (LSTM) networks. As an example and not by way of limitation, the parser may be based on a recurrent neural network grammar (RNNNG) model, which is a type of recurrent and recursive LSTM algorithm. More information on natural-language understanding (NLU) may be found in U.S. patent application Ser. No. 16/011,062, filed 18 Jun. 2018, U.S. patent application Ser. No. 16/025,317, filed 2 Jul. 2018, and U.S. patent application Ser. No. 16/038,120, filed 17 Jul. 2018, each of which is incorporated by reference.

[0091] In particular embodiments, the output of the NLU module 210 may be sent to the entity resolution module 212 to resolve relevant entities. Entities may include, for example, unique users or concepts, each of which may have a unique identifier (ID). The entities may include one or more of a real-world entity (from general knowledge base), a user entity (from user memory), a contextual entity (device context/dialog context), or a value resolution (numbers, datetime, etc.). In particular embodiments, the entity resolution module 212 may comprise domain entity resolution 340 and generic entity resolution 342. The entity resolution module 212 may execute generic and domain-specific entity resolution. The generic entity resolution 342 may resolve the entities by categorizing the slots and meta slots into different generic topics. The domain entity resolution 340 may resolve the entities by categorizing the slots and meta slots into different domains. As an example and not by way of limitation, in response to the input of an inquiry of the advantages of a particular brand of electric car, the generic entity resolution 342 may resolve the referenced brand of electric car as vehicle and the domain entity resolution 340 may resolve the referenced brand of electric car as electric car.

[0092] In particular embodiments, entities may be resolved based on knowledge 350 about the world and the user. The assistant system 140 may extract ontology data from the graphs 352. As an example and not by way of limitation, the graphs 352 may comprise one or more of a knowledge graph, a social graph, or a concept graph. The ontology data may comprise the structural relationship between different slots/meta-slots and domains. The ontology data may also comprise information of how the slots/meta-slots may be grouped, related within a hierarchy where the higher level comprises the domain, and subdivided according to similarities and differences. For example, the knowledge graph may comprise a plurality of entities. Each entity may comprise a single record associated with one or more attribute values. The particular record may be associated with a unique entity identifier. Each record may have diverse values for an attribute of the entity. Each attribute value may be associated with a confidence probability and/or a semantic weight. A confidence probability for an attribute value represents a probability that the value is accurate for the given attribute. A semantic weight for an

attribute value may represent how the value semantically appropriate for the given attribute considering all the available information. For example, the knowledge graph may comprise an entity of a book titled "BookName", which may include information extracted from multiple content sources (e.g., an online social network, online encyclopedias, book review sources, media databases, and entertainment content sources), which may be deduped, resolved, and fused to generate the single unique record for the knowledge graph. In this example, the entity titled "BookName" may be associated with a "fantasy" attribute value for a "genre" entity attribute. More information on the knowledge graph may be found in U.S. patent application Ser. No. 16/048,049, filed 27 Jul. 2018, and U.S. patent application Ser. No. 16/048,101, filed 27 Jul. 2018, each of which is incorporated by reference.

[0093] In particular embodiments, the assistant user memory (AUM) 354 may comprise user episodic memories which help determine how to assist a user more effectively. The AUM 354 may be the central place for storing, retrieving, indexing, and searching over user data. As an example and not by way of limitation, the AUM 354 may store information such as contacts, photos, reminders, etc. Additionally, the AUM 354 may automatically synchronize data to the server and other devices (only for non-sensitive data). As an example and not by way of limitation, if the user sets a nickname for a contact on one device, all devices may synchronize and get that nickname based on the AUM 354. In particular embodiments, the AUM 354 may first prepare events, user state, reminder, and trigger state for storing in a data store. Memory node identifiers (ID) may be created to store entry objects in the AUM 354, where an entry may be some piece of information about the user (e.g., photo, reminder, etc.) As an example and not by way of limitation, the first few bits of the memory node ID may indicate that this is a memory node ID type, the next bits may be the user ID, and the next bits may be the time of creation. The AUM 354 may then index these data for retrieval as needed. Index ID may be created for such purpose. In particular embodiments, given an "index key" (e.g., PHOTO\_LOCATION) and "index value" (e.g., "San Francisco"), the AUM 354 may get a list of memory IDs that have that attribute (e.g., photos in San Francisco). As an example and not by way of limitation, the first few bits may indicate this is an index ID type, the next bits may be the user ID, and the next bits may encode an "index key" and "index value". The AUM 354 may further conduct information retrieval with a flexible query language. Relation index ID may be created for such purpose. In particular embodiments, given a source memory node and an edge type, the AUM 354 may get memory IDs of all target nodes with that type of outgoing edge from the source. As an example and not by way of limitation, the first few bits may indicate this is a relation index ID type, the next bits may be the user ID, and the next bits may be a source node ID and edge type. In particular embodiments, the AUM 354 may help detect concurrent updates of different events. More information on episodic memories may be found in U.S. patent application Ser. No. 16/552,559, filed 27 Aug. 2019, which is incorporated by reference.

[0094] In particular embodiments, the entity resolution module 212 may use different techniques to resolve different types of entities. For real-world entities, the entity resolution module 212 may use a knowledge graph to resolve the span to the entities, such as "music track", "movie", etc. For user

entities, the entity resolution module **212** may use user memory or some agents to resolve the span to user-specific entities, such as “contact”, “reminders”, or “relationship”. For contextual entities, the entity resolution module **212** may perform coreference based on information from the context engine **220** to resolve the references to entities in the context, such as “him”, “her”, “the first one”, or “the last one”. In particular embodiments, for coreference, the entity resolution module **212** may create references for entities determined by the NLU module **210**. The entity resolution module **212** may then resolve these references accurately. As an example and not by way of limitation, a user input may comprise “find me the nearest grocery store and direct me there”. Based on coreference, the entity resolution module **212** may interpret “there” as “the nearest grocery store”. In particular embodiments, coreference may depend on the information from the context engine **220** and the dialog manager **216** so as to interpret references with improved accuracy. In particular embodiments, the entity resolution module **212** may additionally resolve an entity under the context (device context or dialog context), such as, for example, the entity shown on the screen or an entity from the last conversation history. For value resolutions, the entity resolution module **212** may resolve the mention to exact value in standardized form, such as numerical value, date time, address, etc.

[0095] In particular embodiments, the entity resolution module **212** may first perform a check on applicable privacy constraints in order to guarantee that performing entity resolution does not violate any applicable privacy policies. As an example and not by way of limitation, an entity to be resolved may be another user who specifies in their privacy settings that their identity should not be searchable on the online social network. In this case, the entity resolution module **212** may refrain from returning that user’s entity identifier in response to a user input. By utilizing the described information obtained from the social graph, the knowledge graph, the concept graph, and the user profile, and by complying with any applicable privacy policies, the entity resolution module **212** may resolve entities associated with a user input in a personalized, context-aware, and privacy-protected manner.

[0096] In particular embodiments, the entity resolution module **212** may work with the ASR module **208** to perform entity resolution. The following example illustrates how the entity resolution module **212** may resolve an entity name. The entity resolution module **212** may first expand names associated with a user into their respective normalized text forms as phonetic consonant representations which may be phonetically transcribed using a double metaphone algorithm. The entity resolution module **212** may then determine an n-best set of candidate transcriptions and perform a parallel comprehension process on all of the phonetic transcriptions in the n-best set of candidate transcriptions. In particular embodiments, each transcription that resolves to the same intent may then be collapsed into a single intent. Each intent may then be assigned a score corresponding to the highest scoring candidate transcription for that intent. During the collapse, the entity resolution module **212** may identify various possible text transcriptions associated with each slot, correlated by boundary timing offsets associated with the slot’s transcription. The entity resolution module **212** may then extract a subset of possible candidate transcriptions for each slot from a plurality (e.g., **1000**) of

candidate transcriptions, regardless of whether they are classified to the same intent. In this manner, the slots and intents may be scored lists of phrases. In particular embodiments, a new or running task capable of handling the intent may be identified and provided with the intent (e.g., a message composition task for an intent to send a message to another user). The identified task may then trigger the entity resolution module **212** by providing it with the scored lists of phrases associated with one of its slots and the categories against which it should be resolved. As an example and not by way of limitation, if an entity attribute is specified as “friend,” the entity resolution module **212** may run every candidate list of terms through the same expansion that may be run at matcher compilation time. Each candidate expansion of the terms may be matched in the precompiled trie matching structure. Matches may be scored using a function based at least in part on the transcribed input, matched form, and friend name. As another example and not by way of limitation, if an entity attribute is specified as “celebrity/notable person,” the entity resolution module **212** may perform parallel searches against the knowledge graph for each candidate set of terms for the slot output from the ASR module **208**. The entity resolution module **212** may score matches based on matched person popularity and ASR-provided score signal. In particular embodiments, when the memory category is specified, the entity resolution module **212** may perform the same search against user memory. The entity resolution module **212** may crawl backward through user memory and attempt to match each memory (e.g., person recently mentioned in conversation, or seen and recognized via visual signals, etc.). For each entity, the entity resolution module **212** may employ matching similarly to how friends are matched (i.e., phonetic). In particular embodiments, scoring may comprise a temporal decay factor associated with a recency with which the name was previously mentioned. The entity resolution module **212** may further combine, sort, and dedupe all matches. In particular embodiments, the task may receive the set of candidates. When multiple high scoring candidates are present, the entity resolution module **212** may perform user-facilitated disambiguation (e.g., getting real-time user feedback from users on these candidates).

[0097] In particular embodiments, the context engine **220** may help the entity resolution module **212** improve entity resolution. The context engine **220** may comprise offline aggregators and an online inference service. The offline aggregators may process a plurality of data associated with the user that are collected from a prior time window. As an example and not by way of limitation, the data may include news feed posts/comments, interactions with news feed posts/comments, search history, etc., that are collected during a predetermined timeframe (e.g., from a prior 90-day window). The processing result may be stored in the context engine **220** as part of the user profile. The user profile of the user may comprise user profile data including demographic information, social information, and contextual information associated with the user. The user profile data may also include user interests and preferences on a plurality of topics, aggregated through conversations on news feed, search logs, messaging platforms, etc. The usage of a user profile may be subject to privacy constraints to ensure that a user’s information can be used only for his/her benefit, and not shared with anyone else. More information on user profiles may be found in U.S. patent application Ser. No.

15/967,239, filed 30 Apr. 2018, which is incorporated by reference. In particular embodiments, the online inference service may analyze the conversational data associated with the user that are received by the assistant system 140 at a current time. The analysis result may be stored in the context engine 220 also as part of the user profile. In particular embodiments, both the offline aggregators and online inference service may extract personalization features from the plurality of data. The extracted personalization features may be used by other modules of the assistant system 140 to better understand user input. In particular embodiments, the entity resolution module 212 may process the information from the context engine 220 (e.g., a user profile) in the following steps based on natural-language processing (NLP). In particular embodiments, the entity resolution module 212 may tokenize text by text normalization, extract syntax features from text, and extract semantic features from text based on NLP. The entity resolution module 212 may additionally extract features from contextual information, which is accessed from dialog history between a user and the assistant system 140. The entity resolution module 212 may further conduct global word embedding, domain-specific embedding, and/or dynamic embedding based on the contextual information. The processing result may be annotated with entities by an entity tagger. Based on the annotations, the entity resolution module 212 may generate dictionaries. In particular embodiments, the dictionaries may comprise global dictionary features which can be updated dynamically offline. The entity resolution module 212 may rank the entities tagged by the entity tagger. In particular embodiments, the entity resolution module 212 may communicate with different graphs 352 including one or more of the social graph, the knowledge graph, or the concept graph to extract ontology data that is relevant to the retrieved information from the context engine 220. In particular embodiments, the entity resolution module 212 may further resolve entities based on the user profile, the ranked entities, and the information from the graphs 352.

[0098] In particular embodiments, the entity resolution module 212 may be driven by the task (corresponding to an agent 228). This inversion of processing order may make it possible for domain knowledge present in a task to be applied to pre-filter or bias the set of resolution targets when it is obvious and appropriate to do so. As an example and not by way of limitation, for the utterance “who is John?” no clear category is implied in the utterance. Therefore, the entity resolution module 212 may resolve “John” against everything. As another example and not by way of limitation, for the utterance “send a message to John”, the entity resolution module 212 may easily determine “John” refers to a person that one can message. As a result, the entity resolution module 212 may bias the resolution to a friend. As another example and not by way of limitation, for the utterance “what is John’s most famous album?” To resolve “John”, the entity resolution module 212 may first determine the task corresponding to the utterance, which is finding a music album. The entity resolution module 212 may determine that entities related to music albums include singers, producers, and recording studios. Therefore, the entity resolution module 212 may search among these types of entities in a music domain to resolve “John.”

[0099] In particular embodiments, the output of the entity resolution module 212 may be sent to the dialog manager 216 to advance the flow of the conversation with the user.

The dialog manager 216 may be an asynchronous state machine that repeatedly updates the state and selects actions based on the new state. The dialog manager 216 may additionally store previous conversations between the user and the assistant system 140. In particular embodiments, the dialog manager 216 may conduct dialog optimization. Dialog optimization relates to the challenge of understanding and identifying the most likely branching options in a dialog with a user. As an example and not by way of limitation, the assistant system 140 may implement dialog optimization techniques to obviate the need to confirm who a user wants to call because the assistant system 140 may determine a high confidence that a person inferred based on context and available data is the intended recipient. In particular embodiments, the dialog manager 216 may implement reinforcement learning frameworks to improve the dialog optimization. The dialog manager 216 may comprise dialog intent resolution 356, the dialog state tracker 218, and the action selector 222. In particular embodiments, the dialog manager 216 may execute the selected actions and then call the dialog state tracker 218 again until the action selected requires a user response, or there are no more actions to execute. Each action selected may depend on the execution result from previous actions. In particular embodiments, the dialog intent resolution 356 may resolve the user intent associated with the current dialog session based on dialog history between the user and the assistant system 140. The dialog intent resolution 356 may map intents determined by the NLU module 210 to different dialog intents. The dialog intent resolution 356 may further rank dialog intents based on signals from the NLU module 210, the entity resolution module 212, and dialog history between the user and the assistant system 140.

[0100] In particular embodiments, the dialog state tracker 218 may use a set of operators to track the dialog state. The operators may comprise necessary data and logic to update the dialog state. Each operator may act as delta of the dialog state after processing an incoming user input. In particular embodiments, the dialog state tracker 218 may comprise a task tracker, which may be based on task specifications and different rules. The dialog state tracker 218 may also comprise a slot tracker and coreference component, which may be rule based and/or recency based. The coreference component may help the entity resolution module 212 to resolve entities. In alternative embodiments, with the coreference component, the dialog state tracker 218 may replace the entity resolution module 212 and may resolve any references/mentions and keep track of the state. In particular embodiments, the dialog state tracker 218 may convert the upstream results into candidate tasks using task specifications and resolve arguments with entity resolution. Both user state (e.g., user’s current activity) and task state (e.g., triggering conditions) may be tracked. Given the current state, the dialog state tracker 218 may generate candidate tasks the assistant system 140 may process and perform for the user. As an example and not by way of limitation, candidate tasks may include “show suggestion,” “get weather information,” or “take photo.” In particular embodiments, the dialog state tracker 218 may generate candidate tasks based on available data from, for example, a knowledge graph, a user memory, and a user task history. In particular embodiments, the dialog state tracker 218 may then resolve the triggers object using the resolved arguments. As an example and not by way of limitation, a user

input “remind me to call mom when she’s online and I’m home tonight” may perform the conversion from the NLU output to the triggers representation by the dialog state tracker **218** as illustrated in Table 1 below:

TABLE 1

Example Conversion from NLU Output to Triggers Representation	
NLU Ontology Representation:	Triggers Representation:
[IN:CREATE_SMART_REMINDER Remind me to [SL:TODO call mom] when [SL:TRIGGER_CONJUNCTION [IN:GET_TRIGGER [SL:TRIGGER_SOCIAL_UPDATE she’s online] and I’m [SL:TRIGGER_LOCATION home] [SL:DATE_TIME tonight] ] ] ]	Triggers: { andTriggers: [ condition: {Contextual Event(mom is online)}, condition: {Contextual Event(location is home)}, condition: {Contextual Event(time is tonight)}})]}]

In the above example, “mom,” “home,” and “tonight” are represented by their respective entities: personEntity, locationEntity, datetimeEntity.

**[0101]** In particular embodiments, the dialog manager **216** may map events determined by the context engine **220** to actions. As an example and not by way of limitation, an action may be a natural-language generation (NLG) action, a display or overlay, a device action, or a retrieval action. The dialog manager **216** may also perform context tracking and interaction management. Context tracking may comprise aggregating real-time stream of events into a unified user state. Interaction management may comprise selecting optimal action in each state. In particular embodiments, the dialog state tracker **218** may perform context tracking (i.e., tracking events related to the user). To support processing of event streams, the dialog state tracker **218a** may use an event handler (e.g., for disambiguation, confirmation, request) that may consume various types of events and update an internal assistant state. Each event type may have one or more handlers. Each event handler may be modifying a certain slice of the assistant state. In particular embodiments, the event handlers may be operating on disjoint subsets of the state (i.e., only one handler may have write-access to a particular field in the state). In particular embodiments, all event handlers may have an opportunity to process a given event. As an example and not by way of limitation, the dialog state tracker **218** may run all event handlers in parallel on every event, and then may merge the state updates proposed by each event handler (e.g., for each event, most handlers may return a NULL update).

**[0102]** In particular embodiments, the dialog state tracker **218** may work as any programmatic handler (logic) that requires versioning. In particular embodiments, instead of directly altering the dialog state, the dialog state tracker **218** may be a side-effect free component and generate n-best candidates of dialog state update operators that propose updates to the dialog state. The dialog state tracker **218** may comprise intent resolvers containing logic to handle different types of NLU intent based on the dialog state and generate the operators. In particular embodiments, the logic may be organized by intent handler, such as a disambiguation intent handler to handle the intents when the assistant system **140** asks for disambiguation, a confirmation intent handler that

comprises the logic to handle confirmations, etc. Intent resolvers may combine the turn intent together with the dialog state to generate the contextual updates for a conversation with the user. A slot resolution component may then recursively resolve the slots in the update operators with resolution providers including the knowledge graph and domain agents. In particular embodiments, the dialog state tracker **218** may update/rank the dialog state of the current dialog session. As an example and not by way of limitation, the dialog state tracker **218** may update the dialog state as “completed” if the dialog session is over. As another example and not by way of limitation, the dialog state tracker **218** may rank the dialog state based on a priority associated with it.

**[0103]** In particular embodiments, the dialog state tracker **218** may communicate with the action selector **222** about the dialog intents and associated content objects. In particular embodiments, the action selector **222** may rank different dialog hypotheses for different dialog intents. The action selector **222** may take candidate operators of dialog state and consult the dialog policies **360** to decide what actions should be executed. In particular embodiments, a dialog policy **360** may be a tree-based policy, which is a pre-constructed dialog plan. Based on the current dialog state, a dialog policy **360** may choose a node to execute and generate the corresponding actions. As an example and not by way of limitation, the tree-based policy may comprise topic grouping nodes and dialog action (leaf) nodes. In particular embodiments, a dialog policy **360** may also comprise a data structure that describes an execution plan of an action by an agent **228**. A dialog policy **360** may further comprise multiple goals related to each other through logical operators. In particular embodiments, a goal may be an outcome of a portion of the dialog policy and it may be constructed by the dialog manager **216**. A goal may be represented by an identifier (e.g., string) with one or more named arguments, which parameterize the goal. As an example and not by way of limitation, a goal with its associated goal argument may be represented as {confirm\_artist, args:{artist: “Madonna” }}. In particular embodiments, goals may be mapped to leaves of the tree of the tree-structured representation of the dialog policy **360**.

**[0104]** In particular embodiments, the assistant system **140** may use hierarchical dialog policies **360** with general policy **362** handling the cross-domain business logic and task policies **364** handling the task/domain specific logic. The general policy **362** may be used for actions that are not specific to individual tasks. The general policy **362** may be used to determine task stacking and switching, proactive tasks, notifications, etc. The general policy **362** may comprise handling low-confidence intents, internal errors, unacceptable user response with retries, and/or skipping or inserting confirmation based on ASR or NLU confidence scores. The general policy **362** may also comprise the logic of ranking dialog state update candidates from the dialog state tracker **218** output and pick the one to update (such as picking the top ranked task intent). In particular embodiments, the assistant system **140** may have a particular interface for the general policy **362**, which allows for consolidating scattered cross-domain policy/business-rules, especial those found in the dialog state tracker **218**, into a function of the action selector **222**. The interface for the general policy **362** may also allow for authoring of self-contained sub-policy units that may be tied to specific

situations or clients (e.g., policy functions that may be easily switched on or off based on clients, situation). The interface for the general policy 362 may also allow for providing a layering of policies with back-off, i.e., multiple policy units, with highly specialized policy units that deal with specific situations being backed up by more general policies 362 that apply in wider circumstances. In this context the general policy 362 may alternatively comprise intent or task specific policy.

[0105] In particular embodiments, a task policy 364 may comprise the logic for action selector 222 based on the task and current state. The task policy 364 may be dynamic and ad-hoc. In particular embodiments, the types of task policies 364 may include one or more of the following types: (1) manually crafted tree-based dialog plans; (2) coded policy that directly implements the interface for generating actions; (3) configurator-specified slot-filling tasks; or (4) machine-learning model based policy learned from data. In particular embodiments, the assistant system 140 may bootstrap new domains with rule-based logic and later refine the task policies 364 with machine-learning models. In particular embodiments, the general policy 362 may pick one operator from the candidate operators to update the dialog state, followed by the selection of a user facing action by a task policy 364. Once a task is active in the dialog state, the corresponding task policy 364 may be consulted to select right actions.

[0106] In particular embodiments, the action selector 222 may select an action based on one or more of the event determined by the context engine 220, the dialog intent and state, the associated content objects, and the guidance from dialog policies 360. Each dialog policy 360 may be subscribed to specific conditions over the fields of the state. After an event is processed and the state is updated, the action selector 222 may run a fast search algorithm (e.g., similarly to the Boolean satisfiability) to identify which policies should be triggered based on the current state. In particular embodiments, if multiple policies are triggered, the action selector 222 may use a tie-breaking mechanism to pick a particular policy. Alternatively, the action selector 222 may use a more sophisticated approach which may dry-run each policy and then pick a particular policy which may be determined to have a high likelihood of success. In particular embodiments, mapping events to actions may result in several technical advantages for the assistant system 140. One technical advantage may include that each event may be a state update from the user or the user's physical/digital environment, which may or may not trigger an action from assistant system 140. Another technical advantage may include possibilities to handle rapid bursts of events (e.g., user enters a new building and sees many people) by first consuming all events to update state, and then triggering action(s) from the final state. Another technical advantage may include consuming all events into a single global assistant state.

[0107] In particular embodiments, the action selector 222 may take the dialog state update operators as part of the input to select the dialog action. The execution of the dialog action may generate a set of expectations to instruct the dialog state tracker 218 to handle future turns. In particular embodiments, an expectation may be used to provide context to the dialog state tracker 218 when handling the user input from next turn. As an example and not by way of limitation, slot request dialog action may have the expecta-

tion of providing a value for the requested slot. In particular embodiments, both the dialog state tracker 218 and the action selector 222 may not change the dialog state until the selected action is executed. This may allow the assistant system 140 to execute the dialog state tracker 218 and the action selector 222 for processing speculative ASR results and to do n-best ranking with dry runs.

[0108] In particular embodiments, the action selector 222 may call different agents 228 for task execution. Meanwhile, the dialog manager 216 may receive an instruction to update the dialog state. As an example and not by way of limitation, the update may comprise awaiting agents' 228 response. An agent 228 may select among registered content providers to complete the action. The data structure may be constructed by the dialog manager 216 based on an intent and one or more slots associated with the intent. In particular embodiments, the agents 228 may comprise first-party agents and third-party agents. In particular embodiments, first-party agents may comprise internal agents that are accessible and controllable by the assistant system 140 (e.g. agents associated with services provided by the online social network, such as messaging services or photo-share services). In particular embodiments, third-party agents may comprise external agents that the assistant system 140 has no control over (e.g., third-party online music application agents, ticket sales agents). The first-party agents may be associated with first-party providers that provide content objects and/or services hosted by the social-networking system 160. The third-party agents may be associated with third-party providers that provide content objects and/or services hosted by the third-party system 170. In particular embodiments, each of the first-party agents or third-party agents may be designated for a particular domain. As an example and not by way of limitation, the domain may comprise weather, transportation, music, shopping, social, videos, photos, events, locations, and/or work. In particular embodiments, the assistant system 140 may use a plurality of agents 228 collaboratively to respond to a user input. As an example and not by way of limitation, the user input may comprise "direct me to my next meeting." The assistant system 140 may use a calendar agent to retrieve the location of the next meeting. The assistant system 140 may then use a navigation agent to direct the user to the next meeting.

[0109] In particular embodiments, the dialog manager 216 may support multi-turn compositional resolution of slot mentions. For a compositional parse from the NLU module 210, the resolver may recursively resolve the nested slots. The dialog manager 216 may additionally support disambiguation for the nested slots. As an example and not by way of limitation, the user input may be "remind me to call Alex". The resolver may need to know which Alex to call before creating an actionable reminder to-do entity. The resolver may halt the resolution and set the resolution state when further user clarification is necessary for a particular slot. The general policy 362 may examine the resolution state and create corresponding dialog action for user clarification. In dialog state tracker 218, based on the user input and the last dialog action, the dialog manager 216 may update the nested slot. This capability may allow the assistant system 140 to interact with the user not only to collect missing slot values but also to reduce ambiguity of more complex/ambiguous utterances to complete the task. In particular embodiments, the dialog manager 216 may further support requesting missing slots in a nested intent and

multi-intent user inputs (e.g., “take this photo and send it to Dad”). In particular embodiments, the dialog manager **216** may support machine-learning models for more robust dialog experience. As an example and not by way of limitation, the dialog state tracker **218** may use neural network based models (or any other suitable machine-learning models) to model belief over task hypotheses. As another example and not by way of limitation, for action selector **222**, highest priority policy units may comprise white-list/black-list overrides, which may have to occur by design; middle priority units may comprise machine-learning models designed for action selection; and lower priority units may comprise rule-based fallbacks when the machine-learning models elect not to handle a situation. In particular embodiments, machine-learning model based general policy unit may help the assistant system **140** reduce redundant disambiguation or confirmation steps, thereby reducing the number of turns to execute the user input.

**[0110]** In particular embodiments, the determined actions by the action selector **222** may be sent to the delivery system **230**. The delivery system **230** may comprise a CU composer **370**, a response generation component **380**, a dialog state writing component **382**, and a text-to-speech (TTS) component **390**. Specifically, the output of the action selector **222** may be received at the CU composer **370**. In particular embodiments, the output from the action selector **222** may be formulated as a <k,c,u,d> tuple, in which k indicates a knowledge source, c indicates a communicative goal, u indicates a user model, and d indicates a discourse model.

**[0111]** In particular embodiments, the CU composer **370** may generate a communication content for the user using a natural-language generation (NLG) component **372**. In particular embodiments, the NLG component **372** may use different language models and/or language templates to generate natural-language outputs. The generation of natural-language outputs may be application specific. The generation of natural-language outputs may be also personalized for each user. In particular embodiments, the NLG component **372** may comprise a content determination component, a sentence planner, and a surface realization component. The content determination component may determine the communication content based on the knowledge source, communicative goal, and the user’s expectations. As an example and not by way of limitation, the determining may be based on a description logic. The description logic may comprise, for example, three fundamental notions which are individuals (representing objects in the domain), concepts (describing sets of individuals), and roles (representing binary relations between individuals or concepts). The description logic may be characterized by a set of constructors that allow the natural-language generator to build complex concepts/roles from atomic ones. In particular embodiments, the content determination component may perform the following tasks to determine the communication content. The first task may comprise a translation task, in which the input to the NLG component **372** may be translated to concepts. The second task may comprise a selection task, in which relevant concepts may be selected among those resulted from the translation task based on the user model. The third task may comprise a verification task, in which the coherence of the selected concepts may be verified. The fourth task may comprise an instantiation task, in which the verified concepts may be instantiated as an executable file that can be processed by the NLG component **372**. The sentence planner

may determine the organization of the communication content to make it human understandable. The surface realization component may determine specific words to use, the sequence of the sentences, and the style of the communication content.

**[0112]** In particular embodiments, the CU composer **370** may also determine a modality of the generated communication content using the UI payload generator **374**. Since the generated communication content may be considered as a response to the user input, the CU composer **370** may additionally rank the generated communication content using a response ranker **376**. As an example and not by way of limitation, the ranking may indicate the priority of the response. In particular embodiments, the CU composer **370** may comprise a natural-language synthesis (NLS) component that may be separate from the NLG component **372**. The NLS component may specify attributes of the synthesized speech generated by the CU composer **370**, including gender, volume, pace, style, or register, in order to customize the response for a particular user, task, or agent. The NLS component may tune language synthesis without engaging the implementation of associated tasks. In particular embodiments, the CU composer **370** may check privacy constraints associated with the user to make sure the generation of the communication content follows the privacy policies. More information on customizing natural-language generation (NLG) may be found in U.S. patent application Ser. No. 15/967,279, filed 30 Apr. 2018, and U.S. patent application Ser. No. 15/966,455, filed 30 Apr. 2018, which is incorporated by reference.

**[0113]** In particular embodiments, the delivery system **230** may perform different tasks based on the output of the CU composer **370**. These tasks may include writing (i.e., storing/updating) the dialog state into the data store **330** using the dialog state writing component **382** and generating responses using the response generation component **380**. In particular embodiments, the output of the CU composer **370** may be additionally sent to the TTS component **390** if the determined modality of the communication content is audio. In particular embodiments, the output from the delivery system **230** comprising one or more of the generated responses, the communication content, or the speech generated by the TTS component **390** may be then sent back to the dialog manager **216**.

**[0114]** In particular embodiments, the orchestrator **206** may determine, based on the output of the entity resolution module **212**, whether to processing a user input on the client system **130** or on the server, or in the third operational mode (i.e., blended mode) using both. Besides determining how to process the user input, the orchestrator **206** may receive the results from the agents **228** and/or the results from the delivery system **230** provided by the dialog manager **216**. The orchestrator **206** may then forward these results to the arbitrator **226**. The arbitrator **226** may aggregate these results, analyze them, select the best result, and provide the selected result to the render output module **232**. In particular embodiments, the arbitrator **226** may consult with dialog policies **360** to obtain the guidance when analyzing these results. In particular embodiments, the render output module **232** may generate a response that is suitable for the client system **130**.

**[0115]** FIG. 4 illustrates an example task-centric flow diagram **400** of processing a user input. In particular embodiments, the assistant system **140** may assist users not

only with voice-initiated experiences but also more proactive, multi-modal experiences that are initiated on understanding user context. In particular embodiments, the assistant system **140** may rely on assistant tasks for such purpose. An assistant task may be a central concept that is shared across the whole assistant stack to understand user intention, interact with the user and the world to complete the right task for the user. In particular embodiments, an assistant task may be the primitive unit of assistant capability. It may comprise data fetching, updating some state, executing some command, or complex tasks composed of a smaller set of tasks. Completing a task correctly and successfully to deliver the value to the user may be the goal that the assistant system **140** is optimized for. In particular embodiments, an assistant task may be defined as a capability or a feature. The assistant task may be shared across multiple product surfaces if they have exactly the same requirements so it may be easily tracked. It may also be passed from device to device, and easily picked up mid-task by another device since the primitive unit is consistent. In addition, the consistent format of the assistant task may allow developers working on different modules in the assistant stack to more easily design around it. Furthermore, it may allow for task sharing. As an example and not by way of limitation, if a user is listening to music on smart glasses, the user may say “play this music on my phone.” In the event that the phone hasn’t been woken or has a task to execute, the smart glasses may formulate a task that is provided to the phone, which may then be executed by the phone to start playing music. In particular embodiments, the assistant task may be retained by each surface separately if they have different expected behaviors. In particular embodiments, the assistant system **140** may identify the right task based on user inputs in different modality or other signals, conduct conversation to collect all necessary information, and complete that task with action selector **222** implemented internally or externally, on server or locally product surfaces. In particular embodiments, the assistant stack may comprise a set of processing components from wake-up, recognizing user inputs, understanding user intention, reasoning about the tasks, fulfilling a task to generate natural-language response with voices.

**[0116]** In particular embodiments, the user input may comprise speech input. The speech input may be received at the ASR module **208** for extracting the text transcription from the speech input. The ASR module **208** may use statistical models to determine the most likely sequences of words that correspond to a given portion of speech received by the assistant system **140** as audio input. The models may include one or more of hidden Markov models, neural networks, deep learning models, or any combination thereof. The received audio input may be encoded into digital data at a particular sampling rate (e.g., 16, 44.1, or 96 kHz) and with a particular number of bits representing each sample (e.g., 8, 16, or 24 bits).

**[0117]** In particular embodiments, the ASR module **208** may comprise one or more of a grapheme-to-phoneme (G2P) model, a pronunciation learning model, a personalized acoustic model, a personalized language model (PLM), or an end-pointing model. In particular embodiments, the grapheme-to-phoneme (G2P) model may be used to determine a user’s grapheme-to-phoneme style (i.e., what it may sound like when a particular user speaks a particular word). In particular embodiments, the personalized acoustic model

may be a model of the relationship between audio signals and the sounds of phonetic units in the language. Therefore, such personalized acoustic model may identify how a user’s voice sounds. The personalized acoustical model may be generated using training data such as training speech received as audio input and the corresponding phonetic units that correspond to the speech. The personalized acoustical model may be trained or refined using the voice of a particular user to recognize that user’s speech. In particular embodiments, the personalized language model may then determine the most likely phrase that corresponds to the identified phonetic units for a particular audio input. The personalized language model may be a model of the probabilities that various word sequences may occur in the language. The sounds of the phonetic units in the audio input may be matched with word sequences using the personalized language model, and greater weights may be assigned to the word sequences that are more likely to be phrases in the language. The word sequence having the highest weight may be then selected as the text that corresponds to the audio input. In particular embodiments, the personalized language model may also be used to predict what words a user is most likely to say given a context. In particular embodiments, the end-pointing model may detect when the end of an utterance is reached. In particular embodiments, based at least in part on a limited computing power of the client system **130**, the assistant system **140** may optimize the personalized language model at runtime during the client-side process. As an example and not by way of limitation, the assistant system **140** may pre-compute a plurality of personalized language models for a plurality of possible subjects a user may talk about. When a user input is associated with a request for assistance, the assistant system **140** may promptly switch between and locally optimize the pre-computed language models at runtime based on user activities. As a result, the assistant system **140** may preserve computational resources while efficiently identifying a subject matter associated with the user input. In particular embodiments, the assistant system **140** may also dynamically re-learn user pronunciations at runtime.

**[0118]** In particular embodiments, the user input may comprise non-speech input. The non-speech input may be received at the context engine **220** for determining events and context from the non-speech input. The context engine **220** may determine multi-modal events comprising voice/text intents, location updates, visual events, touch, gaze, gestures, activities, device/application events, and/or any other suitable type of events. The voice/text intents may depend on the ASR module **208** and the NLU module **210**. The location updates may be consumed by the dialog manager **216** to support various proactive/reactive scenarios. The visual events may be based on person or object appearing in the user’s field of view. These events may be consumed by the dialog manager **216** and recorded in transient user state to support visual co-reference (e.g., resolving “that” in “how much is that shirt?” and resolving “him” in “send him my contact”). The gaze, gesture, and activity may result in flags being set in the transient user state (e.g., user is running) which may condition the action selector **222**. For the device/application events, if an application makes an update to the device state, this may be published to the assistant system **140** so that the dialog manager **216** may use this context (what is currently displayed to the user) to handle reactive and proactive scenarios. As an example and



not by way of limitation, the context engine 220 may cause a push notification message to be displayed on a display screen of the user's client system 130. The user may interact with the push notification message, which may initiate a multi-modal event (e.g., an event workflow for replying to a message received from another user). Other example multi-modal events may include seeing a friend, seeing a landmark, being at home, running, starting a call with touch, taking a photo with touch, opening an application, etc. In particular embodiments, the context engine 220 may also determine world/social events based on world/social updates (e.g., weather changes, a friend getting online). The social updates may comprise events that a user is subscribed to, (e.g., friend's birthday, posts, comments, other notifications). These updates may be consumed by the dialog manager 216 to trigger proactive actions based on context (e.g., suggesting a user call a friend on their birthday, but only if the user is not focused on something else). As an example and not by way of limitation, receiving a message may be a social event, which may trigger the task of reading the message to the user.

[0119] In particular embodiments, the text transcription from the ASR module 208 may be sent to the NLU module 210. The NLU module 210 may process the text transcription and extract the user intention (i.e., intents) and parse the slots or parsing result based on the linguistic ontology. In particular embodiments, the intents and slots from the NLU module 210 and/or the events and contexts from the context engine 220 may be sent to the entity resolution module 212. In particular embodiments, the entity resolution module 212 may resolve entities associated with the user input based on the output from the NLU module 210 and/or the context engine 220. The entity resolution module 212 may use different techniques to resolve the entities, including accessing user memory from the assistant user memory (AUM) 354. In particular embodiments, the AUM 354 may comprise user episodic memories helpful for resolving the entities by the entity resolution module 212. The AUM 354 may be the central place for storing, retrieving, indexing, and searching over user data.

[0120] In particular embodiments, the entity resolution module 212 may provide one or more of the intents, slots, entities, events, context, or user memory to the dialog state tracker 218. The dialog state tracker 218 may identify a set of state candidates for a task accordingly, conduct interaction with the user to collect necessary information to fill the state, and call the action selector 222 to fulfill the task. In particular embodiments, the dialog state tracker 218 may comprise a task tracker 410. The task tracker 410 may track the task state associated with an assistant task. In particular embodiments, a task state may be a data structure persistent cross interaction turns and updates in real time to capture the state of the task during the whole interaction. The task state may comprise all the current information about a task execution status, such as arguments, confirmation status, confidence score, etc. Any incorrect or outdated information in the task state may lead to failure or incorrect task execution. The task state may also serve as a set of contextual information for many other components such as the ASR module 208, the NLU module 210, etc.

[0121] In particular embodiments, the task tracker 410 may comprise intent handlers 411, task candidate ranking module 414, task candidate generation module 416, and merging layer 419. In particular embodiments, a task may be

identified by its ID name. The task ID may be used to associate corresponding component assets if it is not explicitly set in the task specification, such as dialog policy 360, agent execution, NLG dialog act, etc. Therefore, the output from the entity resolution module 212 may be received by a task ID resolution component 417 of the task candidate generation module 416 to resolve the task ID of the corresponding task. In particular embodiments, the task ID resolution component 417 may call a task specification manager API 430 to access the triggering specifications and deployment specifications for resolving the task ID. Given these specifications, the task ID resolution component 417 may resolve the task ID using intents, slots, dialog state, context, and user memory.

[0122] In particular embodiments, the technical specification of a task may be defined by a task specification. The task specification may be used by the assistant system 140 to trigger a task, conduct dialog conversation, and find a right execution module (e.g., agents 228) to execute the task. The task specification may be an implementation of the product requirement document. It may serve as the general contract and requirements that all the components agreed on. It may be considered as an assembly specification for a product, while all development partners deliver the modules based on the specification. In particular embodiments, an assistant task may be defined in the implementation by a specification. As an example and not by way of limitation, the task specification may be defined as the following categories. One category may be a basic task schema which comprises the basic identification information such as ID, name, and the schema of the input arguments. Another category may be a triggering specification, which is about how a task can be triggered, such as intents, event message ID, etc. Another category may be a conversational specification, which is for dialog manager 216 to conduct the conversation with users and systems. Another category may be an execution specification, which is about how the task will be executed and fulfilled. Another category may be a deployment specification, which is about how a feature will be deployed to certain surfaces, local, and group of users.

[0123] In particular embodiments, the task specification manager API 430 may be an API for accessing a task specification manager. The task specification manager may be a module in the runtime stack for loading the specifications from all the tasks and providing interfaces to access all the tasks specifications for detailed information or generating task candidates. In particular embodiments, the task specification manager may be accessible for all components in the runtime stack via the task specification manager API 430. The task specification manager may comprise a set of static utility functions to manage tasks with the task specification manager, such as filtering task candidates by platform. Before landing the task specification, the assistant system 140 may also dynamically load the task specifications to support end-to-end development on the development stage.

[0124] In particular embodiments, the task specifications may be grouped by domains and stored in runtime configurations 435. The runtime stack may load all the task specifications from the runtime configurations 435 during the building time. In particular embodiments, in the runtime configurations 435, for a domain, there may be a cconf file and a cinc file (e.g., sidechef\_task.conf and sidechef\_task.inc). As an example and not by way of limitation,

<domain>\_tasks.cconf may comprise all the details of the task specifications. As another example and not by way of limitation, <domain>\_tasks.cinc may provide a way to override the generated specification if there is no support for that feature yet.

[0125] In particular embodiments, a task execution may require a set of arguments to execute. Therefore, an argument resolution component 418 may resolve the argument names using the argument specifications for the resolved task ID. These arguments may be resolved based on NLU outputs (e.g., slot [SL:contact]), dialog state (e.g., short-term calling history), user memory (such as user preferences, location, long-term calling history, etc.), or device context (such as timer states, screen content, etc.). In particular embodiments, the argument modality may be text, audio, images or other structured data. The slot to argument mapping may be defined by a filling strategy and/or language ontology. In particular embodiments, given the task triggering specifications, the task candidate generation module 416 may look for the list of tasks to be triggered as task candidates based on the resolved task ID and arguments.

[0126] In particular embodiments, the generated task candidates may be sent to the task candidate ranking module 414 to be further ranked. The task candidate ranking module 414 may use a rule-based ranker 415 to rank them. In particular embodiments, the rule-based ranker 415 may comprise a set of heuristics to bias certain domain tasks. The ranking logic may be described as below with principles of context priority. In particular embodiments, the priority of a user specified task may be higher than an on-foreground task. The priority of the on-foreground task may be higher than a device-domain task when the intent is a meta intent. The priority of the device-domain task may be higher than a task of a triggering intent domain. As an example and not by way of limitation, the ranking may pick the task if the task domain is mentioned or specified in the utterance, such as “create a timer in TIMER app”. As another example and not by way of imitation, the ranking may pick the task if the task domain is on foreground or active state, such as “stop the timer” to stop the timer while the TIMER app is on foreground and there is an active timer. As yet another example and not by way of imitation, the ranking may pick the task if the intent is general meta intent, and the task is device control while there is no other active application or active state. As yet another example and not by way of imitation, the ranking may pick the task if the task is the same as the intent domain. In particular embodiments, the task candidate ranking module 414 may customize some more logic to check the match of intent/slot/entity types. The ranked task candidates may be sent to the merging layer 419.

[0127] In particular embodiments, the output from the entity resolution module 212 may also sent to a task ID resolution component 412 of the intent handlers 411. The task ID resolution component 412 may resolve the task ID of the corresponding task similarly to the task ID resolution component 417. In particular embodiments, the intent handlers 411 may additionally comprise an argument resolution component 413. The argument resolution component 413 may resolve the argument names using the argument specifications for the resolved task ID similarly to the argument resolution component 418. In particular embodiments, intent handlers 411 may deal with task agnostic features and may not be expressed within the task specifications which are task specific. Intent handlers 411 may output state candi-

dates other than task candidates such as argument update, confirmation update, disambiguation update, etc. In particular embodiments, some tasks may require very complex triggering conditions or very complex argument filling logic that may not be reusable by other tasks even if they were supported in the task specifications (e.g., in-call voice commands, media tasks via [IN:PLAY\_MEDIA], etc.). Intent handlers 411 may be also suitable for such type of tasks. In particular embodiments, the results from the intent handlers 411 may take precedence over the results from the task candidate ranking module 414. The results from the intent handlers 411 may be also sent to the merging layer 419.

[0128] In particular embodiments, the merging layer 419 may combine the results from the intent handlers 411 and the results from the task candidate ranking module 414. The dialog state tracker 218 may suggest each task as a new state for the dialog policies 360 to select from, thereby generating a list of state candidates. The merged results may be further sent to a conversational understanding reinforcement engine (CURE) tracker 420. In particular embodiments, the CURE tracker 420 may be a personalized learning process to improve the determination of the state candidates by the dialog state tracker 218 under different contexts using real-time user feedback. More information on conversational understanding reinforcement engine may be found in U.S. patent application Ser. No. 17/186,459, filed 26 Feb. 2021, which is incorporated by reference.

[0129] In particular embodiments, the state candidates generated by the CURE tracker 420 may be sent to the action selector 222. The action selector 222 may consult with the task policies 364, which may be generated from execution specifications accessed via the task specification manager API 430. In particular embodiments, the execution specifications may describe how a task should be executed and what actions the action selector 222 may need to take to complete the task.

[0130] In particular embodiments, the action selector 222 may determine actions associated with the system. Such actions may involve the agents 228 to execute. As a result, the action selector 222 may send the system actions to the agents 228 and the agents 228 may return the execution results of these actions. In particular embodiments, the action selector may determine actions associated with the user or device. Such actions may need to be executed by the delivery system 230. As a result, the action selector 222 may send the user/device actions to the delivery system 230 and the delivery system 230 may return the execution results of these actions.

[0131] The embodiments disclosed herein may include or be implemented in conjunction with an artificial reality system. Artificial reality is a form of reality that has been adjusted in some manner before presentation to a user, which may include, e.g., a virtual reality (VR), an augmented reality (AR), a mixed reality (MR), a hybrid reality, or some combination and/or derivatives thereof. Artificial reality content may include completely generated content or generated content combined with captured content (e.g., real-world photographs). The artificial reality content may include video, audio, haptic feedback, or some combination thereof, and any of which may be presented in a single channel or in multiple channels (such as stereo video that produces a three-dimensional effect to the viewer). Additionally, in some embodiments, artificial reality may be associated with applications, products, accessories, services,

or some combination thereof, that are, e.g., used to create content in an artificial reality and/or used in (e.g., perform activities in) an artificial reality. The artificial reality system that provides the artificial reality content may be implemented on various platforms, including a head-mounted display (HMD) connected to a host computer system, a standalone HMD, a mobile device or computing system, or any other hardware platform capable of providing artificial reality content to one or more viewers.

#### Enabling Voice Commands for Dynamic Settings

**[0132]** In particular embodiments, the assistant system **140** may enable voice commands for settings on different assistant-enabled devices (e.g., a VR headset, a smart tablet, a smart watch, smart glasses, etc.). As an example and not by way of limitation, the voice commands may be “enable night display,” “navigate to Bluetooth settings,” “set Wi-fi,” “turn off Bluetooth,” or “pair keyboard.” Available settings may vary device to device and enabling voice support via assistant system **140** for each of these settings may be difficult. Supporting each of these settings manually may be tedious and maintaining customized code per setting may be costly. Furthermore, deploying client-side code may be a very slow process, and settings may change even on the same device (e.g., as the user’s firmware updates; as the user changes to different apps, etc.). To address such problems, the assistant system **140** may have a dynamic settings feature, where client devices may register their capabilities to a registry with the assistant system **140** in real-time, which the assistant system **140** may then access to enable voice commands for those settings. As a result, the assistant system **140** may allow users to set, query, and navigate to any device setting, reduce gaps in voice command support, scale gracefully, eliminate the need for backend changes when settings change, and maintain the consistency of the codebase. Although this disclosure describes enabling voice commands for particular settings by particular systems in a particular manner, this disclosure contemplates enabling voice commands for any suitable setting by any suitable system in any suitable manner.

**[0133]** In many scenarios, the assistant system **140** may have things on various client devices that the assistant system **140** may allow users to control using voice. As an example and not by way of limitation, the assistant system **140** may have a number of settings on various client devices that the assistant system **140** may want to allow users to control using voice. Traditional approach for enabling setting control using voice may be to create a separate pair of specialized intents for each setting needed to toggle (for toggleable settings) and in general be as specific and granular as possible. The reason may be to improve the accuracy of NLU module **210** by providing sufficient signals. The issue is that the entire process of the traditional approach may be very slow and tedious. This may restrict the ability of the assistant system **140** to quickly enable small features that more or less behave the same way and thus limits the scaling significantly. It may involve a lot of boilerplate code, hard-coded strings and configurations, and delays as the assistant system **140** waits for code to deploy, teams to align, client code to ship, etc. The whole process may take many weeks to enable support for switching a single flag via voice. This may restrict scaling. Furthermore, as the local configuration

changes on the client device, certain options may become inaccessible or new ones revealed (e.g., toggling simple guardian).

**[0134]** To address the aforementioned issues, the embodiments disclosed herein may deal with all the settings associated with the assistant system **140** in a consistent way. The assistant system **140** may allow the user to set any setting on any client device with minimal engineering overhead.

**[0135]** In particular embodiments, the high-level approach for enabling voice control of settings may be illustrated as follows. A client device may first register its capabilities to a registry. The assistant system **140** may then communicate with the registry. A payload may be then generated from the registry and put in device context. An assistant server associated with the assistant system **140** may then invoke actions without knowing details. The client device may then use the registry to map to a setting handler. The NLU module **210** or entity resolution module **212** may then ingest new settings on a known cadence.

**[0136]** In particular embodiments, the components of the dynamic settings feature may include backend components and client-side components. For backend components, the assistant system **140** may use a few meta-agents to handle a whole class of settings at once. This may have the benefit of being much easier to maintain and to scale dynamically. In particular embodiments, the backend components may use generic intents such as [IN:OPEN\_RESOURCE] for navigating to a certain section or setting, [IN:TURN\_ON/TURN\_OFF] for toggling a setting, [IN:OPEN\_RESOURCE] for navigating to a certain setting and [IN:UPDATE\_SETTING] for setting a setting to a value or choice. The backend components may also use additional intents with intent handlers that can be added for customized utterances. In particular embodiments, the backend components may allow the client device to override settings, but by default it may be the assistant server that will decide what to do. Besides the aforementioned intents, the assistant system **140** may have the following intents/agents: navigate to {setting}, set {setting} to {value}, enable {setting}, disable {setting}, trigger/invoke {setting}, modify {setting} by {delta}, and increase/decrease {setting}. Each agent may act on all relevant settings and apply the same operation to whichever one is selected.

**[0137]** The following describes example actions by a navigate-to-setting agent. This agent may send a payload that include the following. “Path” may be the section of the settings to navigate to. “Setting ID” may be the ID of the setting to navigate to. This may be used for quick fixes as well as a fallback. This may be also used as the normal handler for the “navigate to settings” intent. If a setting is too new for the client device to know how to address, and the assistant system **140** needs to ship it quickly, this agent may use this route until the change rolls out to clients. “Route priority” (optional) may be a Boolean that specifies how the client device should deal with the supplied route. The default value may prioritize the client-side route over the server-provided one. However, in some cases, such as during development or hot-fixes, the assistant system **140** may need to prioritize the server-provided route even if the client device has its own route already.

**[0138]** The following describes example actions by an update-setting agent. This agent may modify a setting to a specific value. For an example utterance “set volume to 5,” the payload may include “setting ID” which is the ID of the

setting to set and “value” which is the exact value this agent is setting. For the above example, ID is “volume” and “value” is “5”.

[0139] In particular embodiments, the client-side components may comprise one or more of a settings registry, a setting class per setting, or a family of settings. The client devices may maintain a list of all the available settings as well as their type and path. They may send a full list of all available options to the assistant server in the device context, e.g., in JSON format. When the client device starts up, it may register all of its known settings into a settings registry. Individual settings may have their own classes that wrap around the setting logic itself. In particular embodiments, that may delegate to an operating-system call.

[0140] Given that the assistant system 140 may have multiple processes dealing with settings, the assistant system 140 may use a form of inter-process communication (IPC) to communicate across the processes. The assistant system 140 may abstract that logic behind the different instances of the assistant settings registry class. A given instance of the registry class may have zero or more setting classes registered to it. Whenever it gets a request to set or get a value, it may broadcast the request to all instances (including itself) and wait for one instance to respond back confirming that the request has been handled.

[0141] FIGS. 5A-5C illustrate an example sequence diagram for a “brightness” setting. FIG. 5A may happen on system boot-up. FIGS. 5B-5C may happen with each request, so the settings registry gets updated every time a request is made. FIG. 6 illustrates an example sequence diagram for a “brightness” setting on the client side.

[0142] In particular embodiments, the components on a client device may comprise a user interface and routes. When a route corresponding to a specific setting is launched, the settings app may open the corresponding section with the setting visible. The setting may be then highlighted for easy identification. To route to specific settings, a query parameter may be added to the route used to open the settings app. For example, the route `‘/device?setting=change-language’` may correspond to the language setting in the device section.

[0143] In particular embodiments, the settings may be defined in a JSON list that is included in the device context. It may comprise the following details for each setting. “Setting ID” may be the ID of the setting, which may be unified across surfaces. “Parent” may be the immediate parent of the setting. This may be blank if the setting is at the root. This may be used to build routes for navigation and user education. “IsAvailable” may be a flag that informs the assistant system 140 whether the setting should be accessible to the user. This may help the assistant system 140 block out disabled or experimental settings while still providing the signals for the NLU module 210. “Setting type” may be the type of the setting. Valid options for this may comprise section, numeric, Boolean, string, or list. “Queryable” may be that if the setting can be read, this flag may be true. “Mutable” may be that if the setting can be manipulated, this flag may be true. “Route” may be an identifier or path that the client registers that the assistant system 140 sends back as-is when the assistant system 140 wants to navigate to this particular setting. “List of options” may be that if the setting has different options, this may list the subset the assistant system 140 wants to expose. The details of settings may additionally comprise data type, flags, type specific details such as min/max and delta for numbers.

[0144] In particular embodiments, the dynamic settings feature may be associated with onboarding surfaces. There may be a few things required on the client-side to enable a surface. First, the client device may need to implement the settings registry interface or use standard implementation, which needs to be accessible by the assistant system and needs to be able to access the settings. The client device may also need to implement setting classes or wrappers around the settings (e.g., based on model-based and rules-based classifiers), including implementing setters for settings that can be changed, implementing getters for settings that can be queried, and providing a route link to make settings navigable. The client device may additionally need to implement the callback handlers (for navigation and reaching the setting).

[0145] In particular embodiments, the NLU module 210 may determine semantic label predictions for dynamic settings task which may be different from NLU semantic label predictions for legacy tasks. As an example and not by way of limitation, for a voice command “set the brightness to 5,” the intent on a smart tablet may be [IN:SET\_BRIGHTNESS], which may be based on rule-based classification (RBC) and statistical model predictions, but the intent on a VR headset may be [IN:UPDATE\_SETTING], which may be based on RBC-only predictions. FIG. 7A illustrates an example flow diagram for NLU semantic predictions on a VR headset. FIG. 7B illustrates an example flow diagram for NLU semantic predictions on a tablet. Maintaining a division like this may be unideal, complicate NLU live traffic annotation, make debugging NLU more difficult, lacks breadth due to coverage for RBC-only intents, and be technical debt generally.

[0146] To address the aforementioned issues for NLU semantic label predictions, the assistant system 140 may utilize modeled predictions for dynamic settings task. FIG. 8A illustrates an example flow diagram for modeled device context. As an example and not by way of limitation, the assistant system 140 may use device context dependent statistical models to homogenize model prediction and alleviate (some) live traffic grading confusion. FIG. 8B illustrates an example flow diagram for intent handler transform. As an example and not by way of limitation, the assistant system 140 may utilize intent handler mapping, which may require no data work for modeling and act as information transformer.

[0147] To address the aforementioned issues for NLU semantic label predictions, the assistant system 140 may also utilize NLU subsets legacy intents or predict legacy intents in parallel with dynamic settings intents. In particular embodiments, the assistant system 140 may use a decentralized modeling architecture to provide parallel legacy and dynamic settings intents. In the event the assistant system 140 transition some particular client devices to dynamic settings, the assistant system 140 may then stand-down legacy domain. Regardless, device domain may be split and this may require downstream accommodation. FIG. 9 illustrates an example flow diagram for providing parallel legacy and dynamic settings intents for “pump up the volume to 5.” The decentralized modeling architecture may have the technical advantages such as low-risk implementation, settings being able to be added in bulk, significantly less code per setting, no need for additional agents, assistant system 140 not getting involved in settings implementations, and paving the way for on-device offline handling.

## Artificial/Augmented Reality Systems

[0148] FIG. 10A illustrates an example artificial reality system 1000A. In particular embodiments, the artificial reality system 1000A may comprise a headset 1004, a controller 1006, and a computing system 1008. A user 1002 may wear the headset 1004 that may display visual artificial reality content to the user 1002. The headset 1004 may include an audio device that may provide audio artificial reality content to the user 1002. The headset 1004 may include one or more cameras which can capture images and videos of environments. The headset 1004 may include an eye tracking system to determine the vergence distance of the user 1002. The headset 1004 may be referred to as a head-mounted display (HMD). The controller 1006 may comprise a trackpad and one or more buttons. The controller 1006 may receive inputs from the user 1002 and relay the inputs to the computing system 1008. The controller 1006 may also provide haptic feedback to the user 1002. The computing system 1008 may be connected to the headset 1004 and the controller 1006 through cables or wireless connections. The computing system 1008 may control the headset 1004 and the controller 1006 to provide the artificial reality content to and receive inputs from the user 1002. The computing system 1008 may be a standalone host computer system, an on-board computer system integrated with the headset 1004, a mobile device, or any other hardware platform capable of providing artificial reality content to and receiving inputs from the user 1002.

[0149] In particular embodiments, the headset 1004 may have two separate internal displays, one for each eye of the user 1002. As illustrated in FIG. 10A, the headset 1004 may completely cover the user's field of view. By being the exclusive provider of visual information to the user 1002, the headset 1004 achieves the goal of providing an immersive artificial-reality experience. One consequence of this, however, is that the user 1002 would not be able to see the physical environment surrounding him, as his vision is shielded by the headset 1004. As such, a passthrough feature may be needed to provide the user with real-time visual information about his physical surroundings. The headset 1004 may comprise several external-facing cameras 1007A-1007C. In particular embodiments, cameras 1007A-1007B may be grayscale cameras and camera 1007C may be an RGB camera. Although a number of cameras 1007 are shown, artificial reality system 1000A may include any number of cameras.

[0150] In particular embodiments, the headset 1004 may have external-facing cameras, such as the three forward-facing cameras 1007A-1007C shown in FIG. 10A. While only three forward-facing cameras 1007A-1007C are shown, the headset 1004 may have any number of cameras facing any direction (e.g., an upward-facing camera to capture the ceiling or room lighting, a downward-facing camera to capture a portion of the user's face and/or body, a backward-facing camera to capture a portion of what's behind the user, and/or an internal camera for capturing the user's eye gaze for eye-tracking purposes). The external-facing cameras are configured to capture the physical environment around the user and may do so continuously to generate a sequence of frames (e.g., as a video). As previously explained, although images captured by the forward-facing cameras 1007A-1007C may be directly displayed to the user 1002 via the headset 1004, doing so would not provide the user with an accurate view of the physical

environment since the cameras 1007A-C cannot physically be located at the exact same location as the user's eyes. As such, the passthrough feature may use a re-projection technique that may generate a 3D representation of the physical environment and then renders images based on the 3D representation from the viewpoints of the user's eyes.

[0151] The 3D representation may be generated based on depth measurements of physical objects observed by the cameras 1007A-1007C. Depth may be measured in a variety of ways. In particular embodiments, depth may be computed based on stereo images. For example, the three forward-facing cameras 1007A-1007C may share an overlapping field of view and be configured to capture images simultaneously. As a result, the same physical object may be captured by the cameras 1007A-1007C at the same time. For example, a particular feature of an object may appear at one pixel  $p_A$  in the image captured by camera 1007A, and the same feature may appear at another pixel  $p_B$  in the image captured by camera 1007B. As long as the depth measurement system knows that the two pixels correspond to the same feature, it could use triangulation techniques to compute the depth of the observed feature. For example, based on the camera 1007A's position within a 3D space and the pixel location of  $p_A$  relative to the camera 1007A's field of view, a line could be projected from the camera 1007A and through the pixel  $p_A$ . A similar line could be projected from the other camera 1007B and through the pixel  $p_B$ . Since both pixels are supposed to correspond to the same physical feature, the two lines should intersect. The two intersecting lines and an imaginary line drawn between the two cameras 1007A and 1007B form a triangle, which could be used to compute the distance of the observed feature from either camera 1007A or 1007B or a point in space where the observed feature is located. The same can be done between either of cameras 1007A-1007B and camera 1007C.

[0152] In particular embodiments, the pose (e.g., position and orientation) of the headset 1004 within the environment may be needed. For example, in order to render the appropriate display for the user 1002 while he is moving about in a virtual environment, the system 1000A would need to determine his position and orientation at any moment. Based on the pose of the headset 1004, the system 1000A may further determine the viewpoint of either of the cameras 1007A-1007C or either of the user's eyes. In particular embodiments, the headset 1004 may be equipped with inertial-measurement units ("IMU"). The data generated by the IMU, along with the stereo imagery captured by the external-facing cameras 1007A-1007B, allow the system 1000A to compute the pose of the headset 1004 using, for example, SLAM (simultaneous localization and mapping) or other suitable techniques.

[0153] In particular embodiments, the artificial reality system 1000A may further have one or more controllers 1006 that enable the user 1002 to provide inputs. The controller 1006 may communicate with the headset 1004 or a separate computing system 1008 via a wireless or wired connection. The controller 1006 may have any number of buttons or other mechanical input mechanisms. In addition, the controller 1006 may have an IMU so that the position of the controller 1006 may be tracked. The controller 1006 may further be tracked based on predetermined patterns on the controller. For example, the controller 1006 may have several infrared LEDs or other known observable features that collectively form a predetermined pattern. Using a

sensor or camera, the system **1000A** may be able to capture an image of the predetermined pattern on the controller. Based on the observed orientation of those patterns, the system may compute the controller's position and orientation relative to the sensor or camera.

[**0154**] The artificial reality system **1000A** may further include a computing system **1008**. The computer unit may be a stand-alone unit that is physically separate from the headset **1004**, or it may be integrated with the headset **1004**. In embodiments where the computing system **1008** is a separate unit, it may be communicatively coupled to the headset **1004** via a wireless or wired link. The computing system **1008** may be a high-performance device, such as a desktop or laptop, or a resource-limited device, such as a mobile phone. A high-performance device may have a dedicated GPU and a high-capacity or constant power source. A resource-limited device, on the other hand, may not have a GPU and may have limited battery capacity. As such, the algorithms that could be practically used by an artificial reality system **1000A** depends on the capabilities of its computing system **1008**.

[**0155**] In embodiments where the computing system **1008** is a high-performance device, an embodiment of the passthrough feature may be designed as follows. Through the external-facing cameras **1007A-1007C** of the headset **1004**, a sequence of images of the surrounding physical environment may be captured. The information captured by the cameras **1007A-1007C**, however, would be misaligned with what the user's eyes would capture since the cameras could not spatially coincide with the user's eyes (e.g., the cameras would be located some distance away from the user's eyes and, consequently, have different viewpoints). As such, simply displaying what the cameras captured to the user would not be an accurate representation of what the user **1002** should perceive.

[**0156**] Instead of simply displaying what was captured, the passthrough feature would re-project information captured by the external-facing cameras **1007A-1007C** to the user **1002**. Each pair of simultaneously captured stereo images may be used to estimate the depths of observed features. As explained above, to measure depth using triangulation, the computing system **1008** would need to find correspondences between the stereo images. For example, the computing system **1008** would determine which two pixels in the pair of stereo images correspond to the same observed feature. A high-performance computing system **1008** may solve the correspondence problem using its GPU and optical flow techniques, which are optimized for such tasks. The correspondence information may then be used to compute depths using triangulation techniques. Based on the computed depths of the observed features, the computing system **1008** could determine where those features are located within a 3D space (since the computing system **1008** also knows where the cameras are in that 3D space). The result may be represented by a dense 3D point cloud, with each point corresponding to an observed feature. The dense point cloud may then be used to generate 3D models of objects in the environment. When the system renders a scene for display, the system could perform visibility tests from the perspectives of the user's eyes. For example, the system may cast rays into the 3D space from a viewpoint that corresponds to each eye of the user. In this manner, the rendered scene that is displayed to the user would be computed from the perspective of the user's eyes, rather than from the

perspective of the external-facing cameras **1007A-1007C**. In particular embodiments, the system may use dynamic distortion correction to apply to the rendered scene.

[**0157**] The process described above, however, may not be feasible for a resource-limited computing unit (e.g., a mobile phone may be the main computational unit for the HMD). For example, unlike systems with powerful computational resources and ample energy sources, a mobile phone cannot rely on GPUs and computationally-expensive algorithms (e.g., optical flow) to perform depth measurements and generate an accurate 3D model of the environment. Thus, to provide passthrough on resource-limited devices, an optimized process may be needed.

[**0158**] FIG. **10B** illustrates an example augmented reality (AR) system **1000B**. The augmented reality system **1000B** may include a head-mounted display (HMD) **1010** (e.g., glasses) comprising a frame **1012**, one or more displays **1014**, and a computing system **1020**. The displays **1014** may be transparent or translucent allowing a user wearing the HMD **1010** to look through the displays **1014** to see the real world and displaying visual artificial reality content to the user at the same time. The HMD **1010** may include an audio device that may provide audio artificial reality content to users. The HMD **1010** may include one or more cameras which can capture images and videos of environments. The HMD **1010** may include an eye tracking system to track the vergence movement of the user wearing the HMD **1010**. The augmented reality system **1000B** may further include a controller comprising a trackpad and one or more buttons. The controller may receive inputs from users and relay the inputs to the computing system **1020**. The controller may also provide haptic feedback to users. The computing system **1020** may be connected to the HMD **1010** and the controller through cables or wireless connections. The computing system **1020** may control the HMD **1010** and the controller to provide the augmented reality content to and receive inputs from users. The computing system **1020** may be a standalone host computer system, an on-board computer system integrated with the HMD **1010**, a mobile device, or any other hardware platform capable of providing artificial reality content to and receiving inputs from users.

#### Voice Command Integration

[**0159**] In particular embodiments, a client system may implement voice commands within an extended reality (XR) environment (e.g., augmented reality (AR) or virtual reality (VR)) via a voice SDK, which allows XR applications to easily integrate voice commands on XR devices (e.g., the client system). In particular embodiments, the client system may implement voice commands in combination with gestures within XR environments via a voice SDK. As XR content becomes more immersive, voice integration may make the content more engaging to interact with various applications. That is, people typically interact with the real world using their voice and hands. Currently, navigating through XR content may be cumbersome as the user may need to navigate through unfamiliar nested menus. As an example and not by way of limitation, a user may need to click through various virtual menus by ray-casting from a controller or inputting text through a virtual keyboard to navigate to a desired destination. To improve upon the user experience, the XR system may implement voice commands, combined with one or more other modalities (e.g., gesture, pose, eye gaze, etc.) to allow a user to perform voice

navigation/search, voice FAQ, and voice-driven gameplay & experiences. More information on implementing voice commands may be found in U.S. patent application Ser. No. 18/050,037, filed 26 Oct. 2022, which is incorporated by reference.

**[0160]** FIGS. 11A-11B illustrate an example flow diagram of processing an audio input. The process 1100 may be performed by a client system as described herein. The client system may be embodied as an XR system. In particular embodiments, the process 1100 may start at step 1102, where a client system receives an utterance from a user. At step 1104, the client system may use ASR and a natural language model to process the utterance. The client system may process the utterance to generate one or more intents and one or more entities. The client system may determine whether it was able to understand the request. If the client system is unable to understand the request, the process 1100 continues to step 1106 where a generic error response 1108 is outputted to the user. In particular embodiments, the client system may partially understand the request or has a low confidence in understanding the request. In particular embodiments, if the client system partially understands the request or has low confidence in understanding the request, the client system may determine whether the client system is able to handle any of the identified intents or slots at step 1110. In particular embodiments, the client system may generate a request to have the user repeat themselves. In particular embodiments, the client system may confirm what the user is attempting to request. In particular embodiments, the client system may present an audio output to the user to respond to the user utterance. If the client system is able to understand the request, the process 1100 may continue to step 1110, where the client system determines whether the client system has the ability to handle the request. As an example and not by way of limitation, the client system may attempt to complete the request to determine whether the client system is able to handle the request. If the client system is unable to handle the request, the process 1100 continues to step 1112 where the client system output one or more error responses 1114 to the user. If the client system is able to handle the request, the process 1100 continues to step 1116 where the client system determines whether the client system has all the information needed to perform and action or respond to the user request. If the client system does not have all the information needed, then the process 1100 continues to step 1118 where the client system may output follow up questions 1120 to request further information from the user. If the client system does have all the information, the client system may proceed to step 1122 (shown in FIG. 11B) to perform an action to respond to the user utterance. The process 1100 continues to a response step 1124, 1128, 1132, 1136, 1140, or 1144 based on the identified intents and entities in the user utterance. The process 1100 can continue to step 1124 to respond with one or more confirmations 1126. The process 1100 can continue to step 1128 to respond with one or more lists 1130. The process 1100 can continue to step 1132 to respond with one or more media actions 1134. The process 1100 can continue to step 1136 to respond with one or more answers to questions 1138. The process 1100 can continue to step 1140 to respond with one or more device setting updates 1142. The process 1100 can continue to step 1144 to perform another type of action or response to the user utterance.

**[0161]** FIG. 12 illustrates an example flow diagram of processing an audio input. In particular embodiments, a client system 1202 (e.g., client system 130) may be embodied as a XR display device, such as an AR headset or a VR headset. In particular embodiments, the client system 1202 may perform one or more processes as described herein. The client system 1202 may interface the assistant system 1204 (e.g., assistant system 140). In particular embodiments, the assistant system 1204 may comprise one or more computing systems. In particular embodiments, the assistant system 1204 may be embodied as an NLP tool to process audio inputs. In particular embodiments, the client system 1202 may have an application 1206 installed on it. The application 1206 may be used to generate and render XR content (e.g., elements and/or environment). In particular embodiments the application 1206 may specify one or more activation methods for triggering reception of audio inputs. As an example and not by way of limitation, a user may trigger an invocation model to place an NPC in an XR environment into a listening mode. In particular embodiments, the application 1206 may send a request to the assistant service 1208 to place one or more microphones of the client system 1202 into a listening mode. In particular embodiments, the assistant service 1208 may communicate with the operating system 1210 to determine whether the application 1206 has permission to access the microphone. In response to the operating system 1210 determining that the application 1206 has access, the operating system 1210 may open access to the application 1206. In particular embodiments, the application 1206 may request to stream the audio inputs received to the assistant system 1204 to process. In particular embodiments, the assistant system 1204 may use an NLP tool to process any received audio inputs. In particular embodiments, the assistant service 1208 may stream audio inputs received from one or more microphones of the client system 1202 to the speech recognition model 1212 of the assistant system 1204. In particular embodiments, the speech recognition model 1212 may transcribe the audio received to text. The speech recognition model 1212 may send the transcribed text to the natural language understanding model 1214. The NLU model may match the text to intent and extract entities for the slots. In particular embodiments, the NLU model 1214 may send the results back to the application 1206. In particular embodiments, the NLU model 1214 may send the intents, slots, and entities to a dialog manager 1216. The dialog manager 1216 may resolve the intents and slots received from the NLU model. In particular embodiments, the assistant system 1204 may perform a task or send instructions to execute a task to the client system 1202. The dialog manager 1216 may send instructions to the application 1206 to perform a task based on the received intents, slots, and entities from the NLU model. In particular embodiments, the assistant system 1204 may generate a result from processing the intents, slots, and entities from the NLU model 1214 and render a response through the text-to-speech module 1218. The response from the text-to-speech module 1218 may send the response to the application 1206 to output to the user of the client system 1202. As an example and not by way of limitation, if the user asked an NPC in an XR environment associated with the application 1206 “What do you have for sale?” the text-to-speech module 1218 may generate a response going through this process and have the application 1206 output the response, “I have baked goods for sale. Would you like to buy some?”

In particular embodiments, the client system 1202 may generate the response through the application 1206 by receiving the intents, slots, and entities from the NLU model 1214.

[0162] FIG. 13 illustrates an example architecture of a system 1300 to process a user input. In particular embodiments, the system 1300 may include an engine 1302. The engine 1302 can be embodied as a game engine. In particular embodiments the engine 1302 may interact with the platform service 1304 through a process boundary 1314. In particular embodiments, the platform service 1304 may be embodied as an assistant system 140. In particular embodiments, the engine 1302 may include a toolkit 1304, one or more applications 1308, a voice SDK 1312, engine application triggers 1324, and engine application callbacks 1358. In particular embodiments, the platform service 1304 may include the voice SDK service process 1316. In particular embodiments, the application 1308 may initially render an XR environment or XR elements, such as object 1318. In particular embodiments, the application 1308 may determine the context of a user of the application 1308. The application 1308 may determine the context of the user with respect to one or more objects 1318 within the XR environment. In particular embodiments, the application 1308 may access a toolkit 1304 to provide additional functionality to the XR environment. In particular embodiments, the object 1318 may send information, such as location of the user with respect to the object 1318, location of the object, context of the application 1308, and the like to the invocation module 1320. In particular embodiments, the invocation module 1320 may determine whether to activate an attention system based on parameters (e.g., user gaze, distance, and the like) received from the object 1318 and as described herein. In particular embodiments, if the invocation module 1320 determines to activate an attention system, the invocation module 1320 may send a request to the attention system module 1322 to change an attention state of the object 1318. In particular embodiments, the toolkit 1304 may track and manage the attention systems and attention states of multiple objects 1318 across different applications 1308. In particular embodiments, the attention system module 1322 may send instructions to the application 1308 to render a different form of the object 1318 to reflect a different attention state. As an example and not by way of limitation, if the object 1318 is a virtual cat initially in a napping position. As a user of the application 1308 approaches the object 1318, the attention system module 1322 may send instructions to the application 1308 to render the virtual cat into a microphone on attention state, where the virtual cat can change into a sitting upright position. In particular embodiments, the application 1308 may receive audio inputs from the user of the application 1308. In response to receiving an audio input, the application 1308 may send a signal to the engine application triggers 1324 to process the audio input. In particular embodiments, the engine application triggers may specify certain conditions to trigger processing an audio input as described herein. As an example and not by way of limitation, the user of the application 1308 may need to perform a gesture and then say an audio input. In particular embodiments, the engine application triggers 1324 may send the audio input from the application 1308 to the voice SDK 1312. In particular embodiments, the voice SDK 1312 determines whether to be

activated at step 1326 to process an audio input through signals received from the engine application triggers 1324 and the toolkit 1304.

[0163] In particular embodiments, the voice SDK 1312 may use step 1326 to determine to process an audio input. At step 1328, the voice SDK 1312 may determine whether or not the platform of the client system running the application 1308 is supported. The voice SDK 1312 may provide additional features to processing an audio input as described herein. As an example and not by way of limitation, the voice SDK 1312 may quickly process audio inputs using customized on-device NLU models. In particular embodiments, if the voice SDK 1312 determines that the platform is not supported, then the audio input may be sent to an NLP tool native 1335. The NLP tool native may activate a microphone input 1350 at step 1348. The microphone input 1350 may call on an API 1354 to process the audio input. In particular embodiments, the API 1354 may be to call on an NLP tool to process the audio input. In particular embodiments, the API 1354 may send back the results of the NLP tool to the response handler 1352. In particular embodiments, the response handler 1352 may send the response to a voice SDK response handler 1356. If the voice SDK 1312 determines that the platform is supported 1328, then the voice SDK platform integration 1330 may be activated at step 1332. In particular embodiments, only certain client systems may be equipped with the right hardware to implement certain features of the voice SDK. The audio input may be passed to the voice SDK service process 1316 by the voice SDK integration 1330 through the process boundary 1314. The voice SDK service process 1316 may determine whether the application 1308 and/or the client system running the application 1308 may have permission to use the voice SDK service process 1316. If the application 1308 and/or client system does not have permission, then the voice SDK service process 1316 may pass the audio input back to the NLP tool native 1335 to handle the audio input. In particular embodiments, if the voice SDK service process 1316 determines that the application 1308 and/or client system has permission, then the audio input gets passed to the NLP platform integration tool 1336. In particular embodiments, the voice SDK service process 1316 may activate the NLP platform integration tool 1336 at step 1338. The audio input may be passed to an OVR microphone input 1340. The OVR microphone input 1340 may pass the audio input to an API 1344. In particular embodiments, the API 1344 may call an NLP tool to process the audio input. In particular embodiments, the result of the NLP tool may be passed to a response handler 1342 of the NLP platform integration tool 1336. In particular embodiments, the response handler 1342 may send the response through the process boundary 1314 to the platform SDK 1346 of the voice SDK platform integration 1330. In particular embodiments, the results of the response handler 1352 and the platform SDK 1346 may be sent to the voice SDK response handler 1356. In particular embodiments, the voice SDK response handler 1356 may generate a response to send back to the application 1308 or the toolkit 1304. In particular embodiments, the voice SDK response handler 1356 may send a response to the engine application callbacks 1358 that may activate the attention system module 1322 or the application 1308 as needed. As an example and not by way of limitation, if the user of the application 1308 had called out an NPC in the XR environment, the voice SDK response



handler **1356** may determine to change an attention state of the NPC using the attention system module **1322** and to render a response to the user through the application **1308**. The engine application callbacks **1358** may call both the toolkit and the application **1308**.

#### Attention System

**[0164]** In particular embodiments, a client system may implement voice commands within an XR environment via a voice SDK, which allows XR applications to easily integrate voice commands on XR devices (e.g., the client system). In particular embodiments, a client system may implement an attention system to provide audio-visual cues to a microphone status in XR environments. There are sometimes issues with users identifying and understanding which entities/objects are interactable by voice within an XR environment. Additionally, users sometimes are not knowledgeable on what voice commands are available to them for certain contexts, such as for assistant experiences on voice-forward and voice-only devices and immersive in-app experiences in XR. To help the user distinguish which objects are interactable via voice command, an attention system may be used to provide audio-visual cues that let users know various attention states of a XR object, such as when the XR object is ready to receive a voice command, when the microphone is active, and when the system is processing the voice command. More information on implementing an attention system may be found in U.S. patent application Ser. No. 18/050,039, filed 26 Oct. 2022, which is incorporated by reference.

**[0165]** In particular embodiments, the attention system may use one or more characteristics of an XR assistant to generate motion and audio cues that are more animated and/or realistic to convey response and emotions. More information on rendering one or more displays of an XR display device may be found in U.S. patent application Ser. No. 17/877,568, filed 29 Jul. 2022, which is incorporated by reference.

**[0166]** In particular embodiments, the assistant system **140** may present different attention states or attention substates to a user in an XR context. In particular embodiments, the attention system may utilize the assistant system to present different attention states or attention substates. More information on presenting different attention states or attention substates to a user may be found in U.S. patent application Ser. No. 17/934,898, filed 23 Sep. 2022, which is incorporated by reference.

**[0167]** FIG. 14 illustrates example indications of an attention state of an object. In particular embodiments, an application of a client system may render an XR environment. The XR environment may include a plurality of XR objects. For the interactable XR objects, an indicator of the attention state may be presented in conjunction with the interactable XR objects to provide an indication that the user of the application may interact with specific objects. In particular embodiments, the attention state indicators may include a microphone off attention state indicator **1402**, a microphone on attention state indicator **1404**, listening attention state indicator **1406**, an error attention state indicator **1408**, a response attention state indicator **1410**, a mismatched understanding attention state indicator **1412**, and a matched understanding attention state indicator **1414**. In particular embodiments, one or more attention state indicators may be presented simultaneously for one or more XR objects. As an

example and not by way of limitation, an XR object may have a response attention state indicator **1410** and listening attention state indicator **1406** displayed at the same time. In particular embodiments, the application may determine where to render the attention state indicators for each interactable XR object.

#### Large Language Model for Voice-Driven NPC Interactions

**[0168]** In particular embodiments, a client system and/or one or more computing systems (e.g., a server-side assistant system) may implement artificial-intelligence-driven (AI-driven) dialog. Typically, in games containing non-playable characters (NPCs), the NPCs may be unable to handle out-of-domain (OOD) requests well. Current interactions with NPCs in games may break immersion due to awkward controls, such as buttons to be clicked, limited choices, and repetitive answers. This may especially be the case for a virtual reality (VR) environment for a game. For instance, NPCs in a game may have a set number of responses for a user input. Responses to OOD requests may often be repetitive (e.g., “Sorry, I can’t do that”, “Sorry, I don’t know the answer to that”) and further break immersion from the environment (e.g., VR environment). Dialogue trees may be extensive and difficult to design well. Additionally, for an immersive game, there may be numerous NPCs to build the game world and provide an immersive experience for users. Therefore, it may be very time-consuming to build out dialogue trees for all of the NPCs. To solve this issue of poor OOD responses, AI-driven dialog may be used to power the responses of NPCs.

**[0169]** In particular embodiments, to use AI-driven dialog in responses of NPCs in games the client system may use a combination of the following components: Voice SDK dictation and text-to-speech (TTS), natural language interface voice intents (for natural-language understanding), reactive voice attention system, presence gaze detection, large language model (LLM) integration, and customizable AI character persona design via LLM prompts, and script writing. Voice SDK may be a voice-interactive platform that leverages natural language processing (NLP) models to bring voice AI functionalities to third-party developers and creator ecosystems. Voice SDK may enable many useful functionalities to third-party developers and creator ecosystems. As an example, Voice SDK may be able to process user voice inputs using TTS and natural language interface voice intents to determine the intent associated with the voice inputs. The use of Voice SDK may allow key transactional purchase flows and other transactions with NPCs to be handled purely by voice responses and actions. Other common actions may be trained and templated via NLP tools and models. As another example and not by way of limitation, instead of designing complex dialogue trees, an AI-driven dialog implemented using Voice SDK and a natural language interface may only need to define character parameters and the transactional/narrative information of the character. The LLM may handle generating the responses the character would output to users that interact with the character. When fully immersed into a VR environment, users may want to interact with characters like they would with another human being in a real transactional environment. By letting an LLM handle generating the responses, this approach saves resources spent needing to develop a complex dialogue tree. This approach also may mitigate key pain points in VR interactions and may improve the developer ability to create

authentic, immersive voice experiences centered around the ubiquitous NPC, which are staples of games and app experiences. As an example and not by way of limitation, for a shopkeeper NPC may help facilitate in-app transactions and purchases. Through enabling generative responses, the interaction with the shopkeeper NPC would be given more depth and personality to the NPC while still allowing for pre-coded responses to purchasing requests. By doing so, this interaction and many others like it may mirror real world situations, such as a shopkeeper taking orders and making chitchat with a customer while fielding questions they may not have context to answer.

**[0170]** In particular embodiments, the reactive voice attention system may provide an additional visual layer where a user can see when an NPC the user is approaching is listening or not listening and also when the NPC is understanding or not understanding. This reactive voice attention system may have both an attention state and an understanding state that may be indicated. In particular embodiments, a single visual space may be used to render both an attention state and an understanding state. As an example and not by way of limitation, an icon above NPCs may be displayed to indicate both an attention state and an understanding state. While the goal is to help maintain immersion, the indication of both an attention state and an understanding state may enable the user to know when a response is in-domain or OOD so they know when they are going beyond the scope of the game or application. A positive understanding state may be indicated when an NPC provides an in-domain response. A negative understanding state may be indicated when an NPC provides an OOD response. Multiple attention and understanding states may be rendered for multiple NPCs simultaneously. As an example and not by way of limitation, if a user is within a VR environment with multiple NPCs and proceeds to ask a question, the user can see which of the NPCs are listening and which NPCs are understanding the question. As a result, multiple NPCs may respond to a single request, where some may respond with in-domain answers and some with OOD answers (e.g., chit chat answers). While the reactive attention voice system may provide a visual indication, NPCs may be customized to provide one or more visual cues corresponding to an attention state and an understanding state. As an example and not by way of limitation, an NPC may look in the general direction of the user if there is a positive attention state. Various icons may be used to indicate an attention state and an understanding state. As an example and not by way of limitation, an ear icon may indicate whether an NPC is listening, a question mark may indicate the NPC is responding to an OOD request, a checkmark may indicate the NPC is responding to an in-domain request, among other icons.

**[0171]** In particular embodiments, in-domain questions, which may be questions corresponding to intents that are associated with an example shopkeeper NPC, may be answered normally based on dialogue trees programmed for the NPC. OOD questions may use the LLM. The LLM may auto generate quality immersive content. The LLM may mitigate latency and error handling user experience pain points by engaging in-character generative dialogue with less repetition. Developers and creators may further customize NPC's AI responses for more authentic user connection. The OOD requests may be tracked and provided to developers. The tracking of OOD requests may provide further information to developers on what users are inquiring and

enable developers to program dialogue for frequent OOD requests. NPCs may manage OOD requests by redirecting users to requests to that the AI supports. As an example and not by way of limitation, for a shopkeeper being asked about a skirmish that happened yesterday, the shopkeeper may respond “. . . . I heard about the skirmish that happened yesterday. So you interested in buying anything?” The user's audio inputs may be analyzed for sentiment, such as whether the user is angry, sad, etc. The system may be able to customize responses based on the sentiment of the user. The synthesized speech and animations of the responses may be customized based on a desired sentiment of the NPC. As an example and not by way of limitation, the NPC may respond in a happy manner if the user provided an audio input with a happy sentiment. The OOD dialog can be tracked and used in subsequent in-domain responses. As an example and not by way of limitation, if a user is talking with a bartender NPC and telling the NPC about a sword the user recently purchased (OOD chit chat), then ask the bartender for a drink (which is an in-domain request), the bartender may leverage the prior OOD dialog with the in-domain response. For instance, the bartender may say “Be careful with your sword after you finish your drink!” This allows for real conversations with the NPC that go beyond pre-programmed dialogue trees. The LLM may generate in-game actions for NPCs. The responses from the LLM can be fed back into the natural language interface to cause the in-game action to happen.

**[0172]** In particular embodiments, to enable this AI-driven dialog, the components including Voice SDK, a natural language interface, and the LLM may be hosted on servers of a single entity to ensure minimal latency. By having the components hosted in a central location, the AI-driven dialog may be provided for a real-time game experience and other real-time experiences.

**[0173]** In particular embodiments, a client system may implement AI-driven dialog. In particular embodiments, one or more computing systems may implement AI-driven dialog. In particular embodiments, a combination of a client system and one or more computing systems may implement AI-driven dialog. In particular embodiments, the client system may be embodied as an XR display device. In particular embodiments, the XR display device may receive an audio input from a first user of the XR display device. As an example and not by way of limitation, the XR display device may receive the audio input via a microphone coupled to the XR display device. In particular embodiments, the XR display device may be associated with an XR environment comprising a plurality of XR objects. As an example and not by way of limitation, the XR display device may present an XR environment (e.g., a virtual reality environment) including XR objects (e.g., virtual reality objects) to the user. In particular embodiments, the XR display device may receive additional audio inputs. Although this disclosure describes receiving an audio input in a particular manner, this disclosure contemplates receiving an audio input in any suitable manner.

**[0174]** In particular embodiments, an XR display device may process an audio input to identify one or more intents and one or more slots associated with the audio input. In particular embodiments, the XR display device may process the audio input using a natural language understanding (NLU) model. Although this disclosure describes processing

an audio input in a particular manner, this disclosure contemplates processing an audio input in any suitable manner.

**[0175]** In particular embodiments, the XR display device may identify a first XR object from the plurality of XR objects that is in an active listening state. As an example and not by way of limitation, the XR display device may determine whether certain parameters have been met to place the first XR object into an active listening state. In particular embodiments, the first XR object may be associated with a first set of intents and a first set of slots. In particular embodiments, the XR display device may identify one or more additional XR objects from the plurality of XR objects that are in an active listening state. Although this disclosure describes identifying a first XR object in a particular manner, this disclosure contemplates identifying a first XR object in any suitable manner.

**[0176]** In particular embodiments, the XR display device may determine that either the first set of intents or the first set of slots do not comprise the one or more identified intents or the one or more identified slots associated with the audio input. As an example and not by way of limitation, the XR display device may determine that the audio input does not contain any intents or slots that are a part of the first XR objects set of intents or set of slots. In particular embodiments, the XR display device may determine whether or not a set of intents or set of slots associated with any XR object may comprise the one or more identified intents or the one or more identified slots. In particular embodiments, the XR display device may receive, by one or more computing systems, one or more of an updated first set of intents or an updated first set of slots to replace the first set of intents or the first set of slots, respectively. In particular embodiments, the updated first set of intents may comprise the one or more identified intents or the updated first set of slots may comprise the one or more identified slots. Although this disclosure describes determining that either set of intents or set of slots do not comprise an identified intent or identified slot in particular manner, this disclosure contemplates determining that either set of intents or set of slots do not comprise an identified intent or identified slot in any suitable manner.

**[0177]** In particular embodiments, the XR display device may generate an out-of-domain (OOD) response. In particular embodiments, the XR display device may generate an OOD response based on one or more characteristics of the first XR object using a large language model (LLM). In particular embodiments, the OOD response may reference one or more of the one or more identified intents or the one or more identified slots associated with the audio input. In particular embodiments, the OOD response may further reference one or more intents associated with the first set of intents or one or more slots associated with the first set of slots. In particular embodiments, the OOD response may prompt the first user to provide an audio input including one or more intents associated with the first set of intents or one or more slots associated with the first set of slots. In particular embodiments, the one or more characteristics of the first XR object comprises one or more of a background of the first XR object or a current state of the first XR object. If the first set of intents or first set of slots are updated to include one or more identified intents or one or more identified slots, the XR display device may generate, using the LLM, an in-domain response based on the one or more characteristics of the first XR object. In particular embodi-

ments, the in-domain response may reference one or more of the one or more identified intents or the one or more identified slots associated with the audio input. Although this disclosure describes generating an OOD response in a particular manner, this disclosure contemplates generating an OOD response in any suitable manner.

**[0178]** In particular embodiments, the XR display device may render, for one or more displays of the XR display device, an output image comprising the first XR object. In particular embodiments, the XR display device may present, by the one or more displays of the XR display device, the output image and the OOD response. In particular embodiments, the XR display device may determine a first attention state of the first XR object based on a first context of the first user. In particular embodiments, the first attention state may indicate a status of the XR object to interact with one or more first voice commands for one or more functions enabled by the XR display device. In particular embodiments, the XR display device may render, for one or more displays of the XR display device, an output image comprising the first XR object and an indication of the first attention state of the first XR object. In particular embodiments, the XR display device may present, by the one or more displays of the XR display device, the output image. In particular embodiments, the indication of the first attention state may comprise one or more of an icon above the first XR object or a visual cue of the first XR object. In particular embodiments, the XR display device may determine a first understanding state of the first XR object based on whether the first set of intents or the first set of slots comprise the one or more identified intents or the one or more identified slots associated with the audio input. In particular embodiments, the XR display device may render, for one or more displays of the XR display device, an output image comprising the first XR object and an indication of the first understanding state of the first XR object. In particular embodiments, the indication of the first attention state may comprise one or more of an icon above the first XR object or a visual cue of the first XR object. In particular embodiments, the XR display device may analyze the audio input to identify one or more sentiments associated with the audio input. In particular embodiments, the XR display device may render, for one or more displays of the XR display device, an output image comprising the first XR object based on the identified one or more sentiments. Although this disclosure describes rendering an output image in a particular manner, this disclosure contemplates rendering an output image in any suitable manner.

**[0179]** In particular embodiments, the XR display device may track a dialog state of a current dialog session. In particular embodiments, the dialog state may comprise one or more candidate tasks corresponding to the one or more identified intents or the one or more identified slots. In particular embodiments, the XR display device may send, to one or more computing systems, information corresponding to the dialog state. Although this disclosure describes tracking a dialog state in a particular manner, this disclosure contemplates tracking a dialog state in any suitable manner.

**[0180]** In particular embodiments, the XR display device may receive one or more subsequent audio inputs from the first user of the XR display device. In particular embodiments, the XR display device may process, using the NLU model, the subsequent audio input to identify one or more intents and one or more slots associated with the subsequent

audio input. In particular embodiments, the XR display device may determine that the first set of intents or the first set of slots comprise the one or more identified intents or the one or more identified slots associated with the subsequent audio input. In particular embodiments, the XR display device may generate, using the LLM, an in-domain response based on one or more characteristics of the first XR object. In particular embodiments, the in-domain response may reference the OOD response. Although this disclosure describes receiving audio inputs in a particular manner, this disclosure contemplates receiving audio inputs in any suitable manner.

**[0181]** While this disclosure describes an XR display device performing one or more actions with respect to a first XR object, this disclosure contemplates the XR display device performing one or more actions with respect to any number of XR objects. In particular embodiments, the XR display device may identify a second XR object from the plurality of XR objects that is in an active listening state, where the second XR object is associated with a second set of intents and a second set of slots. In particular embodiments, the XR display device may determine whether the second set of intents or the second set of slots comprise the one or more identified intents or the one or more identified slots associated with the audio input. In particular embodiments, if the second set of intents or second set of slots do not comprise the one or more identified intents or the one or more identified slots associated with the audio input, then the XR display device may generate, using the LLM, a second OOD response based on one or more characteristics of the second XR object, wherein the second OOD response references one or more of the one or more identified intents or the one or more identified slots associated with the audio input. In particular embodiments, if the second set of intents or second set of slots do comprise the one or more identified intents or the one or more identified slots associated with the audio input, then the XR display device may generate, using the LLM, an in-domain response based on one or more characteristics of the second XR object. In particular embodiments, the in-domain response may reference the OOD response of the first XR object.

**[0182]** FIGS. 15A-15B illustrates an example extended reality (XR) environment 1500 containing AI voice-driven interactions. Referring to FIG. 15A, in particular embodiments, the environment 1500 may comprise an XR object or NPC 1502 embodied as a swordsmith. In particular embodiments, the environment 1500 may include an icon 1504 indicative of an attention state of the NPC 1502. In particular embodiments, a user of an XR display device may be presented one or more images corresponding to the XR environment 1500 and including the NPC 1502, icon 1504 and state 1506 of the NPC 1502. The initial state 1506 of the NPC 1502 may be in a listening state. Referring to FIG. 15B, in particular embodiments, the environment 1500 may change when the XR display device receives an audio input 1510 from the user. As an example and not by way of limitation, the audio input 1510 may comprise “Oh tell me about your swords”. In particular embodiments, the audio input 1510 may be processed as described herein. In particular embodiments, the environment 1500 may change to reflect the NPC 1502 processing the audio input 1510. As an example and not by way of limitation, the environment 1500 may present an icon 1508 indicative of an understanding state of the XR object (e.g., NPC 1502). The icon 1508 may

indicate that the NPC 1502 understands the audio input (e.g., the audio input comprises one or more intents or one or more slots in the set of intents or set of slots associated with the NPC 1502). In particular embodiments, the state 1506 of the NPC 1502 may change to indicate the NPC 1502 is speaking. In particular embodiments, given the audio input comprises one or more intents or one or more slots that are in-domain of the NPC 1502, the NPC 1502 may return an in-domain response 1512. In particular embodiments, the XR display device may present the environment 1500 with the in-domain response 1512 in one or more modalities (e.g., audio, visual, etc.).

**[0183]** FIGS. 16A-16B illustrates another example extended reality (XR) environment 1600 containing AI voice-driven interactions. Referring to FIG. 16A, in particular embodiments, the environment 1600 may comprise an XR object or NPC 1602 embodied as a barkeep. In particular embodiments, the environment 1600 may include an icon 1604 indicative of an attention state of the NPC 1602. In particular embodiments, a user of an XR display device may be presented one or more images corresponding to the XR environment 1600 and including the NPC 1602, icon 1604, and a state 1606 of the NPC 1602. The initial state 1606 of the NPC 1602 may be in a listening state. Referring to FIG. 16B, in particular embodiments, the environment 1600 may change when the XR display device receives an audio input 1610 from the user. As an example and not by way of limitation, the audio input 1610 may comprise “Do the roving goblins hear about not cause you any concern?” In particular embodiments, the audio input 1610 may be processed as described herein. In particular embodiments, the environment 1610 may change to reflect the NPC 1602 processing the audio input 1610. As an example and not by way of limitation, the environment 1600 may present an icon 1608 indicative of an understanding state of the XR object (e.g., NPC 1602). In particular embodiments, the icon 1608 may indicate that the NPC 1602 does not understand the audio input (e.g., the audio input does not comprise one or more intents or one or more slots in the set of intents or set of slots associated with the NPC 1602). In particular embodiments, the state 1606 of the NPC 1602 may change to indicate the NPC 1602 is speaking. In particular embodiments, given the audio input does not comprise one or more intents or one or more slots that are in-domain of the NPC 1602, the NPC 1602 may return an OOD response 1612. In particular embodiments, the XR display device may present the environment 1600 with the OOD response 1612 in one or more modalities (e.g., audio, visual, etc.).

**[0184]** FIG. 17 illustrates an example method 1700 for implementing AI-driven dialog. In particular embodiments, one or more computing systems embodied as server-side assistant system may perform the method of implementing AI-driven dialog. In particular embodiments, a client system may also perform the method of implementing AI-driven dialog. In particular embodiments, a client system may be embodied as an extended reality (XR) display device. The method may begin at step 1710, where an XR display device may receive, by the XR display device, an audio input from a first user of the XR display device. In particular embodiments, the XR display device may be associated with an XR environment comprising a plurality of XR objects. At step 1720, the XR display device may process, using a natural language understanding (NLU) model, the audio input to identify one or more intents and one or more slots associated

with the audio input. At step **1730**, the XR display device may identify a first XR object from the plurality of XR objects that is in an active listening state. In particular embodiments, the first XR object may be associated with a first set of intents and a first set of slots. At step **1740**, the XR display device may determine that either the first set of intents or the first set of slots do not comprise the one or more identified intents or the one or more identified slots associated with the audio input. At step **1750**, the XR display device may generate, using a large language model (LLM), an out-of-domain (OOD) response based on one or more characteristics of the first XR object. In particular embodiments, the OOD response may reference one or more of the one or more identified intents or the one or more identified slots associated with the audio input. Particular embodiments may repeat one or more steps of the method of FIG. **17**, where appropriate. Although this disclosure describes and illustrates particular steps of the method of FIG. **17** as occurring in a particular order, this disclosure contemplates any suitable steps of the method of FIG. **17** occurring in any suitable order. Moreover, although this disclosure describes and illustrates an example method for implementing AI-driven dialog including the particular steps of the method of FIG. **17**, this disclosure contemplates any suitable method for implementing AI-driven dialog including any suitable steps, which may include all, some, or none of the steps of the method of FIG. **17**, where appropriate. Furthermore, although this disclosure describes and illustrates particular components, devices, or systems carrying out particular steps of the method of FIG. **17**, this disclosure contemplates any suitable combination of any suitable components, devices, or systems carrying out any suitable steps of the method of FIG. **17**.

#### Bilingual Models for Live Translation

**[0185]** In particular embodiments, the assistant system **140** may use a single bilingual ASR model to handle generation of text for downstream services in scenarios where two languages are both present (i.e., mixed-language or “code-switched” inputs). The single bilingual ASR model may receive input comprising a mix of words from two languages and generate output comprising text with words correctly transcribed. The text string may be then passed to the downstream services, such as a translation model which is also bilingual. The output text string can also include an output tag for each word indicating the language ID. This way the downstream service (e.g., translation model) would know what language the text is in. The translation model may then convert the input string to the desired language. Having a single bilingual ASR model instead of two separate monolingual ASR models may save memory, which is particularly helpful for compact assistant-enabled devices such as smart glasses. One example use case for the single bilingual ASR model may be live translation in travel scenarios, where participants can switch between two languages and the assistant system **140** can allow for a more natural conversation. For example, a wearer of smart glasses may ask “what’s the tv show ‘La Casa De Papel’ about?” The assistant system **140** may then use the bilingual ASR model to generate a transcription including both English and Spanish text-strings, executes tasks based on the English and Spanish text-strings, and generate a question-answering (O&A) response accordingly. The smart glasses may display or speak: “‘La Casa De Papel’ is about a group of burglars

pulling off the most daring robbery in the history of Spain.” Although this disclosure describes using particular models for particular application by particular systems in a particular manner, this disclosure contemplates using any suitable model for any suitable application by any suitable system in any suitable manner.

**[0186]** Conventional systems may use two monolingual ASR models (e.g., an English ASR model and a Spanish ASR model) to handle generation of text in scenarios where two languages are both present. For example, a typical scenario may be translation between the two languages and the two monolingual ASR models can translate in each direction separately. A problem with monolingual ASR models may be that they fail when speech in a particular language is sent to ASR for the wrong language. Similarly, they may fail when the speech input contains a mix of words in the two languages. For example, in translation, conventional systems may typically have each person talk into a monolingual ASR model. The text may be then provided to a translation model, which translates it to the other language. This may be a problem for mixed-language (code-switched) inputs. The conventional systems may either have to try to translate input in both languages (generating partial junk in both ASR text outputs) or parse audio input into respective languages and then send the partial audio to the correct monolingual ASR model. In addition, generating mixed-language outputs may not appear to be possible for these types of conventional systems. Another problem of using two monolingual ASR models may be that having two monolingual ASR models takes too much space, which may be not suitable for mobile client devices (e.g., smart glasses) that are constrained on storage and processing capabilities.

**[0187]** The solution for the aforementioned problems of conventional systems may be to use a single bilingual ASR model for both languages, e.g., English and Spanish. The experiments of this disclosure show that the dictation accuracy for a single bilingual ASR model is comparable to monolingual models. The single bilingual ASR model has better performance than monolingual models on dictation datasets with 4.07% relative WER improvement over an English ASR model and 10.27% relative WER improvement over a Spanish ASR model. On the other hand, having a single bilingual ASR model for both languages instead of two separate monolingual ASR models can save memory.

**[0188]** In particular embodiments, the system may use training data focused on the travel scenario, where two people would have an open-domain conversation. As an example and not by way of limitation, for English the system may use supervised and unsupervised video data and dictation data collected from smart glasses. For Spanish, given that we may generally have less data than for English and no dedicated collected dictation data, the system may use whatever data that is available, e.g., supervised and unsupervised video data and live-traffic data from smart tablets, etc. All the data may be distorted with speed and noise perturbation. Table 2 shows the original number of hours for each dataset, before distortion. In particular embodiments, the assistant system **140** may train a RNN-T model, which also supports punctuation. Thus, the data may be prepared to include dot, comma and question marks as well.

TABLE 2

Dataset statistics.		
Language	Dataset	# hours
English	Video supervised	22.6K
	Video unsupervised	1.6M (final number, used in training)
	Smart glasses dictation	7K (final, together with assistant)
Spanish	Video supervised	11.2K
	Video unsupervised	140K
	Appen	848.5
	Applause	16.3
	Tablet Live Traffic	1250

[0189] In particular embodiments, for faster turn-around time, one may use the data described above, minus the unsupervised video data. Afterwards, unsupervised data may be integrated in model training for the most promising directions.

[0190] For evaluation, the embodiments disclosed herein used a dataset associated with dictation. Thus, for English we used the smart glasses dictation dataset and for Spanish we used a dictation dataset that was already collected with 2500 utterances.

[0191] In particular embodiments, the assistant system 140 may use a plurality of approaches to train the bilingual ASR model. We may also combine the two languages to have a single sentence-piece model. During training, English and Spanish may be given equal weights overall (e.g., 0.5 each). The model architecture may be at a high level similar to an English ASR model. However, there may be some configurations that changed, given that we are not including Assistant as a separate domain. Below we list some example approaches for training the bilingual ASR model.

[0192] In particular embodiments, the assistant system 140 may train the bilingual ASR model based on a joint English and Spanish model. This may be simply combining all the English and Spanish data sources during model training. As an example and not by way of limitation, the assistant system 140 may use SPM size of both 4095 and 9000. A larger SPM may improve the performance. In particular embodiments, the assistant system 140 may train the bilingual ASR model based on the joint English and Spanish model where Spanish is uppercased. The difference may be that all the Spanish data is uppercased. This had a much higher WER for Spanish, due to higher deletion rate than for the other models.

[0193] In particular embodiments, the assistant system 140 may train the bilingual ASR model based on a joint model with language identifier (ID). We may give as input the language identifier as an extra feature during model training. In particular embodiments, we may randomly sample 10% of the data with the “unknown” language-ID tag, which we may then use during decoding.

[0194] In particular embodiments, the assistant system 140 may train the bilingual ASR model based on a joint English and Spanish model using language tags. Instead of simply combining the English and Spanish data sources, the assistant system 140 may also add a language tag during data preparation, where each sentence has an attached tag, <en> or <es> at the beginning of the sentence.

[0195] The embodiments disclosed herein conducted evaluation with punctuation. As described above, the RNN-T bilingual model may also support punctuation (e.g.,

dot, comma and question mark). The experiments results include numbers on the punctuation metrics. Overall, even if the WER improves, the punctuation is more variable, with improvements in some cases and degradation in others.

[0196] The bilingual ASR model may be combined with multi-channel audio inputs in order to have a fully working ASR model that suppresses crosstalk and recognizes both English and Spanish. The bilingual ASR model may also use the crosstalk filtering technology to cut out crosstalk to make sure that the crosstalk doesn’t get translated. This may also help in the scenario where one of the persons switches languages to make sure the system doesn’t get confused who is the wearer themselves.

[0197] The bilingual ASR model may be further combined with localized named-entity translation. The bilingual ASR model may be used to handle mixed-language inputs that reference named entities.

[0198] Some example use cases of the bilingual ASR model are as follows. One example use case is handling user commands with mixed languages. For example, a bilingual speaker may speak to Portal, “hey portal, Que hora es ahora?” The assistant system 140 may use the bilingual ASR model to generate a transcription including the English text-string, which is determined to be a wake-word for a smart tablet, and the Spanish text-string. Both text-strings may be provided to downstream services to determine the corresponding task. The smart tablet may reply/display the answering in both languages. As another example, a user may say “book a dinner reservation at Ristorante Italiano.” The assistant system 140 may use the bilingual ASR model to generate both English and Italian text-strings and book the reservation at the right place. The assistant system 140 may then reply “I’ve booked the dinner reservation at 7 pm tomorrow at Ristorante Italiano.” Another example use case is live translation. For example, the user wearing smart glasses may ask “what does ‘ciao’ mean?” The assistant system 140 may reply “‘Ciao’ means ‘hello’ or ‘bye’ in Italian.” As another example, an English-speaker wearing smart glasses may be having a conversation with a Spanish speaker. The bilingual ASR model can take utterances from both people in both English and Spanish, and generate translation accordingly. For example, the English-speaker may say “So, can you just explain what is uh system uh how it works?” The assistant system 140 may use the bilingual ASR model to generate a transcription of English text-strings and translate it to “Entonces, ¿puedes explicarme qué es el Sistema uh cómo funciona?” which may be shown to both the Spanish-speaker and English-speaker. The Spanish-speaker may then say “Eh si el systema eh usa los microfonos para entender que yo so.” The assistant system 140 may then use the bilingual ASR model to generate Spanish text-strings and translates it to “hey yeah the system eh use the microphones to understand that I’m the one talking” and show/say it to the English-speaker. Another use case may be travel assistance. A user traveling in Barcelona may ask the assistant system 140: “where is Basilica de Santa Maria del Pi?” The assistant system 140 may reply “Basilica de Santa Maria del Pi is located at Plaça del Pi, 7.”

#### Social Graphs

[0199] FIG. 18 illustrates an example social graph 1800. In particular embodiments, the social-networking system 160 may store one or more social graphs 1800 in one or more data stores. In particular embodiments, the social graph 1800

may include multiple nodes—which may include multiple user nodes **1802** or multiple concept nodes **1804**—and multiple edges **1806** connecting the nodes. Each node may be associated with a unique entity (i.e., user or concept), each of which may have a unique identifier (ID), such as a unique number or username. The example social graph **1800** illustrated in FIG. **18** is shown, for didactic purposes, in a two-dimensional visual map representation. In particular embodiments, a social-networking system **160**, a client system **130**, an assistant system **140**, or a third-party system **170** may access the social graph **1800** and related social-graph information for suitable applications. The nodes and edges of the social graph **1800** may be stored as data objects, for example, in a data store (such as a social-graph database). Such a data store may include one or more searchable or queryable indexes of nodes or edges of the social graph **1800**.

**[0200]** In particular embodiments, a user node **1802** may correspond to a user of the social-networking system **160** or the assistant system **140**. As an example and not by way of limitation, a user may be an individual (human user), an entity (e.g., an enterprise, business, or third-party application), or a group (e.g., of individuals or entities) that interacts or communicates with or over the social-networking system **160** or the assistant system **140**. In particular embodiments, when a user registers for an account with the social-networking system **160**, the social-networking system **160** may create a user node **1802** corresponding to the user, and store the user node **1802** in one or more data stores. Users and user nodes **1802** described herein may, where appropriate, refer to registered users and user nodes **1802** associated with registered users. In addition or as an alternative, users and user nodes **1802** described herein may, where appropriate, refer to users that have not registered with the social-networking system **160**. In particular embodiments, a user node **1802** may be associated with information provided by a user or information gathered by various systems, including the social-networking system **160**. As an example and not by way of limitation, a user may provide his or her name, profile picture, contact information, birth date, sex, marital status, family status, employment, education background, preferences, interests, or other demographic information. In particular embodiments, a user node **1802** may be associated with one or more data objects corresponding to information associated with a user. In particular embodiments, a user node **1802** may correspond to one or more web interfaces.

**[0201]** In particular embodiments, a concept node **1804** may correspond to a concept. As an example and not by way of limitation, a concept may correspond to a place (such as, for example, a movie theater, restaurant, landmark, or city); a website (such as, for example, a website associated with the social-networking system **160** or a third-party website associated with a web-application server); an entity (such as, for example, a person, business, group, sports team, or celebrity); a resource (such as, for example, an audio file, video file, digital photo, text file, structured document, or application) which may be located within the social-networking system **160** or on an external server, such as a web-application server; real or intellectual property (such as, for example, a sculpture, painting, movie, game, song, idea, photograph, or written work); a game; an activity; an idea or theory; another suitable concept; or two or more such concepts. A concept node **1804** may be associated with information of a concept provided by a user or information

gathered by various systems, including the social-networking system **160** and the assistant system **140**. As an example and not by way of limitation, information of a concept may include a name or a title; one or more images (e.g., an image of the cover page of a book); a location (e.g., an address or a geographical location); a website (which may be associated with a URL); contact information (e.g., a phone number or an email address); other suitable concept information; or any suitable combination of such information. In particular embodiments, a concept node **1804** may be associated with one or more data objects corresponding to information associated with concept node **1804**. In particular embodiments, a concept node **1804** may correspond to one or more web interfaces.

**[0202]** In particular embodiments, a node in the social graph **1800** may represent or be represented by a web interface (which may be referred to as a “profile interface”). Profile interfaces may be hosted by or accessible to the social-networking system **160** or the assistant system **140**. Profile interfaces may also be hosted on third-party websites associated with a third-party system **170**. As an example and not by way of limitation, a profile interface corresponding to a particular external web interface may be the particular external web interface and the profile interface may correspond to a particular concept node **1804**. Profile interfaces may be viewable by all or a selected subset of other users. As an example and not by way of limitation, a user node **1802** may have a corresponding user-profile interface in which the corresponding user may add content, make declarations, or otherwise express himself or herself. As another example and not by way of limitation, a concept node **1804** may have a corresponding concept-profile interface in which one or more users may add content, make declarations, or express themselves, particularly in relation to the concept corresponding to concept node **1804**.

**[0203]** In particular embodiments, a concept node **1804** may represent a third-party web interface or resource hosted by a third-party system **170**. The third-party web interface or resource may include, among other elements, content, a selectable or other icon, or other inter-actable object representing an action or activity. As an example and not by way of limitation, a third-party web interface may include a selectable icon such as “like,” “check-in,” “eat,” “recommend,” or another suitable action or activity. A user viewing the third-party web interface may perform an action by selecting one of the icons (e.g., “check-in”), causing a client system **130** to send to the social-networking system **160** a message indicating the user’s action. In response to the message, the social-networking system **160** may create an edge (e.g., a check-in-type edge) between a user node **1802** corresponding to the user and a concept node **1804** corresponding to the third-party web interface or resource and store edge **1806** in one or more data stores.

**[0204]** In particular embodiments, a pair of nodes in the social graph **1800** may be connected to each other by one or more edges **1806**. An edge **1806** connecting a pair of nodes may represent a relationship between the pair of nodes. In particular embodiments, an edge **1806** may include or represent one or more data objects or attributes corresponding to the relationship between a pair of nodes. As an example and not by way of limitation, a first user may indicate that a second user is a “friend” of the first user. In response to this indication, the social-networking system **160** may send a “friend request” to the second user. If the second user

confirms the “friend request,” the social-networking system 160 may create an edge 1806 connecting the first user’s user node 1802 to the second user’s user node 1802 in the social graph 1800 and store edge 1806 as social-graph information in one or more of data stores 164. In the example of FIG. 18, the social graph 1800 includes an edge 1806 indicating a friend relation between user nodes 1802 of user “A” and user “B” and an edge indicating a friend relation between user nodes 1802 of user “C” and user “B.” Although this disclosure describes or illustrates particular edges 1806 with particular attributes connecting particular user nodes 1802, this disclosure contemplates any suitable edges 1806 with any suitable attributes connecting user nodes 1802. As an example and not by way of limitation, an edge 1806 may represent a friendship, family relationship, business or employment relationship, fan relationship (including, e.g., liking, etc.), follower relationship, visitor relationship (including, e.g., accessing, viewing, checking-in, sharing, etc.), subscriber relationship, superior/subordinate relationship, reciprocal relationship, non-reciprocal relationship, another suitable type of relationship, or two or more such relationships. Moreover, although this disclosure generally describes nodes as being connected, this disclosure also describes users or concepts as being connected. Herein, references to users or concepts being connected may, where appropriate, refer to the nodes corresponding to those users or concepts being connected in the social graph 1800 by one or more edges 1806. The degree of separation between two objects represented by two nodes, respectively, is a count of edges in a shortest path connecting the two nodes in the social graph 1800. As an example and not by way of limitation, in the social graph 1800, the user node 1802 of user “C” is connected to the user node 1802 of user “A” via multiple paths including, for example, a first path directly passing through the user node 1802 of user “B,” a second path passing through the concept node 1804 of company “CompanyName” and the user node 1802 of user “D,” and a third path passing through the user nodes 1802 and concept nodes 1804 representing school “SchoolName,” user “G,” company “CompanyName,” and user “D.” User “C” and user “A” have a degree of separation of two because the shortest path connecting their corresponding nodes (i.e., the first path) includes two edges 1806.

[0205] In particular embodiments, an edge 1806 between a user node 1802 and a concept node 1804 may represent a particular action or activity performed by a user associated with user node 1802 toward a concept associated with a concept node 1804. As an example and not by way of limitation, as illustrated in FIG. 18, a user may “like,” “attended,” “played,” “listened,” “cooked,” “worked at,” or “read” a concept, each of which may correspond to an edge type or subtype. A concept-profile interface corresponding to a concept node 1804 may include, for example, a selectable “check in” icon (such as, for example, a clickable “check in” icon) or a selectable “add to favorites” icon. Similarly, after a user clicks these icons, the social-networking system 160 may create a “favorite” edge or a “check in” edge in response to a user’s action corresponding to a respective action. As another example and not by way of limitation, a user (user “C”) may listen to a particular song (“SongName”) using a particular application (a third-party online music application). In this case, the social-networking system 160 may create a “listened” edge 1806 and a “used” edge (as illustrated in FIG. 18) between user nodes 1802

corresponding to the user and concept nodes 1804 corresponding to the song and application to indicate that the user listened to the song and used the application. Moreover, the social-networking system 160 may create a “played” edge 1806 (as illustrated in FIG. 18) between concept nodes 1804 corresponding to the song and the application to indicate that the particular song was played by the particular application. In this case, “played” edge 1806 corresponds to an action performed by an external application (the third-party online music application) on an external audio file (the song “SongName”). Although this disclosure describes particular edges 1806 with particular attributes connecting user nodes 1802 and concept nodes 1804, this disclosure contemplates any suitable edges 1806 with any suitable attributes connecting user nodes 1802 and concept nodes 1804. Moreover, although this disclosure describes edges between a user node 1802 and a concept node 1804 representing a single relationship, this disclosure contemplates edges between a user node 1802 and a concept node 1804 representing one or more relationships. As an example and not by way of limitation, an edge 1806 may represent both that a user likes and has used at a particular concept. Alternatively, another edge 1806 may represent each type of relationship (or multiples of a single relationship) between a user node 1802 and a concept node 1804 (as illustrated in FIG. 18 between user node 1802 for user “E” and concept node 1804 for “online music application”).

[0206] In particular embodiments, the social-networking system 160 may create an edge 1806 between a user node 1802 and a concept node 1804 in the social graph 1800. As an example and not by way of limitation, a user viewing a concept-profile interface (such as, for example, by using a web browser or a special-purpose application hosted by the user’s client system 130) may indicate that he or she likes the concept represented by the concept node 1804 by clicking or selecting a “Like” icon, which may cause the user’s client system 130 to send to the social-networking system 160 a message indicating the user’s liking of the concept associated with the concept-profile interface. In response to the message, the social-networking system 160 may create an edge 1806 between user node 1802 associated with the user and concept node 1804, as illustrated by “like” edge 1806 between the user and concept node 1804. In particular embodiments, the social-networking system 160 may store an edge 1806 in one or more data stores. In particular embodiments, an edge 1806 may be automatically formed by the social-networking system 160 in response to a particular user action. As an example and not by way of limitation, if a first user uploads a picture, reads a book, watches a movie, or listens to a song, an edge 1806 may be formed between user node 1802 corresponding to the first user and concept nodes 1804 corresponding to those concepts. Although this disclosure describes forming particular edges 1806 in particular manners, this disclosure contemplates forming any suitable edges 1806 in any suitable manner.

#### Vector Spaces and Embeddings

[0207] FIG. 19 illustrates an example view of a vector space 1900. In particular embodiments, an object or an n-gram may be represented in a d-dimensional vector space, where d denotes any suitable number of dimensions. Although the vector space 1900 is illustrated as a three-dimensional space, this is for illustrative purposes only, as



the vector space **1900** may be of any suitable dimension. In particular embodiments, an n-gram may be represented in the vector space **1900** as a vector referred to as a term embedding. Each vector may comprise coordinates corresponding to a particular point in the vector space **1900** (i.e., the terminal point of the vector). As an example and not by way of limitation, vectors **1910**, **1920**, and **1930** may be represented as points in the vector space **1900**, as illustrated in FIG. **19**. An n-gram may be mapped to a respective vector representation. As an example and not by way of limitation, n-grams  $t_1$  and  $t_2$  may be mapped to vectors  $\vec{v}_1$  and  $\vec{v}_2$  in the vector space **1900**, respectively, by applying a function  $\vec{\pi}$  defined by a dictionary such that  $\vec{v}_1 = \vec{\pi}(t_1)$  and  $\vec{v}_2 = \vec{\pi}(t_2)$ . As another example and not by way of limitation, a dictionary trained to map text to a vector representation may be utilized, or such a dictionary may be itself generated via training. As another example and not by way of limitation, a word-embeddings model may be used to map an n-gram to a vector representation in the vector space **1900**. In particular embodiments, an n-gram may be mapped to a vector representation in the vector space **1900** by using a machine learning model (e.g., a neural network). The machine learning model may have been trained using a sequence of training data (e.g., a corpus of objects each comprising n-grams).

[**0208**] In particular embodiments, an object may be represented in the vector space **1900** as a vector referred to as a feature vector or an object embedding. As an example and not by way of  $\vec{v}_1$  limitation, objects  $e_1$  and  $e_2$  may be mapped to vectors  $\vec{v}_1$  and  $\vec{v}_2$  in the vector space **1900**, respectively, by applying a function  $\vec{\pi}$ , such that  $\vec{v}_1 = \vec{\pi}(e_1)$  and  $\vec{v}_2 = \vec{\pi}(e_2)$ . In particular embodiments, an object may be mapped to a vector based on one or more properties, attributes, or features of the object, relationships of the object with other objects, or any other suitable information associated with the object. As an example and not by way of limitation, a function  $\vec{\pi}$  may map objects to vectors by feature extraction, which may start from an initial set of measured data and build derived values (e.g., features). As an example and not by way of limitation, an object comprising a video or an image may be mapped to a vector by using an algorithm to detect or isolate various desired portions or shapes of the object. Features used to calculate the vector may be based on information obtained from edge detection, corner detection, blob detection, ridge detection, scale-invariant feature transformation, edge direction, changing intensity, autocorrelation, motion detection, optical flow, thresholding, blob extraction, template matching, Hough transformation (e.g., lines, circles, ellipses, arbitrary shapes), or any other suitable information. As another example and not by way of limitation, an object comprising audio data may be mapped to a vector based on features such as a spectral slope, a tonality coefficient, an audio spectrum centroid, an audio spectrum envelope, a Mel-frequency cepstrum, or any other suitable information. In particular embodiments, when an object has data that is either too large to be efficiently processed or comprises redundant data, a function  $\vec{\pi}$  may map the object to a vector using a transformed reduced set of features (e.g., feature selection). In particular embodiments, a function  $\vec{\pi}$  may map an object  $e$  to a vector  $\vec{\pi}(e)$  based on one or more n-grams associated

with object  $e$ . Although this disclosure describes representing an n-gram or an object in a vector space in a particular manner, this disclosure contemplates representing an n-gram or an object in a vector space in any suitable manner.

[**0209**] In particular embodiments, the social-networking system **160** may calculate a similarity metric of vectors in vector space **1900**. A similarity metric may be a cosine similarity, a Minkowski distance, a Mahalanobis distance, a Jaccard similarity coefficient, or any suitable similarity metric. As an example and not by way of limitation, a similarity metric of  $\vec{v}_1$  and  $\vec{v}_2$  may be a cosine similarity

$$\frac{\vec{v}_1 \cdot \vec{v}_2}{\|\vec{v}_1\| \|\vec{v}_2\|}$$

As another example and not by way of limitation, a similarity metric of  $\vec{v}_1$  and  $\vec{v}_2$  may be a Euclidean distance  $\|\vec{v}_1 - \vec{v}_2\|$ . A similarity metric of two vectors may represent how similar the two objects or n-grams corresponding to the two vectors, respectively, are to one another, as measured by the distance between the two vectors in the vector space **1900**. As an example and not by way of limitation, vector **1910** and vector **1920** may correspond to objects that are more similar to one another than the objects corresponding to vector **1910** and vector **1930**, based on the distance between the respective vectors. Although this disclosure describes calculating a similarity metric between vectors in a particular manner, this disclosure contemplates calculating a similarity metric between vectors in any suitable manner.

[**0210**] More information on vector spaces, embeddings, feature vectors, and similarity metrics may be found in U.S. patent application Ser. No. 14/949,436, filed 23 Nov. 2015, U.S. patent application Ser. No. 15/286,315, filed 5 Oct. 2016, and U.S. patent application Ser. No. 15/365,789, filed 30 Nov. 2016, each of which is incorporated by reference.

Artificial Neural Networks

[**0211**] FIG. **20** illustrates an example artificial neural network (“ANN”) **2000**. In particular embodiments, an ANN may refer to a computational model comprising one or more nodes. Example ANN **2000** may comprise an input layer **2010**, hidden layers **2020**, **2030**, **2040**, and an output layer **2050**. Each layer of the ANN **2000** may comprise one or more nodes, such as a node **2005** or a node **2015**. In particular embodiments, each node of an ANN may be connected to another node of the ANN. As an example and not by way of limitation, each node of the input layer **2010** may be connected to one of more nodes of the hidden layer **2020**. In particular embodiments, one or more nodes may be a bias node (e.g., a node in a layer that is not connected to and does not receive input from any node in a previous layer). In particular embodiments, each node in each layer may be connected to one or more nodes of a previous or subsequent layer. Although FIG. **20** depicts a particular ANN with a particular number of layers, a particular number of nodes, and particular connections between nodes, this disclosure contemplates any suitable ANN with any suitable number of layers, any suitable number of nodes, and any suitable connections between nodes. As an example and not by way of limitation, although FIG. **20** depicts a connection between each node of the input layer **2010** and each node of

the hidden layer **2020**, one or more nodes of the input layer **2010** may not be connected to one or more nodes of the hidden layer **2020**.

[**0212**] In particular embodiments, an ANN may be a feedforward ANN (e.g., an ANN with no cycles or loops where communication between nodes flows in one direction beginning with the input layer and proceeding to successive layers). As an example and not by way of limitation, the input to each node of the hidden layer **2020** may comprise the output of one or more nodes of the input layer **2010**. As another example and not by way of limitation, the input to each node of the output layer **2050** may comprise the output of one or more nodes of the hidden layer **2040**. In particular embodiments, an ANN may be a deep neural network (e.g., a neural network comprising at least two hidden layers). In particular embodiments, an ANN may be a deep residual network. A deep residual network may be a feedforward ANN comprising hidden layers organized into residual blocks. The input into each residual block after the first residual block may be a function of the output of the previous residual block and the input of the previous residual block. As an example and not by way of limitation, the input into residual block N may be  $F(x)+x$ , where  $F(x)$  may be the output of residual block N-1,  $x$  may be the input into residual block N-1. Although this disclosure describes a particular ANN, this disclosure contemplates any suitable ANN.

[**0213**] In particular embodiments, an activation function may correspond to each node of an ANN. An activation function of a node may define the output of a node for a given input. In particular embodiments, an input to a node may comprise a set of inputs. As an example and not by way of limitation, an activation function may be an identity function, a binary step function, a logistic function, or any other suitable function. As another example and not by way of limitation, an activation function for a node k may be the sigmoid function

$$F_k(s_k) = \frac{1}{1 + e^{-s_k}},$$

the hyperbolic tangent function

$$F_k(s_k) = \frac{e^k - e^{-s_k}}{e^k + e^{-s_k}},$$

the rectifier  $F_k(s_k)=\max(0,s_k)$ , or any other suitable function  $F_k(s_k)$ , where  $s_k$  may be the effective input to node k. In particular embodiments, the input of an activation function corresponding to a node may be weighted. Each node may generate output using a corresponding activation function based on weighted inputs. In particular embodiments, each connection between nodes may be associated with a weight. As an example and not by way of limitation, a connection **2025** between the node **2005** and the node **2015** may have a weighting coefficient of 0.4, which may indicate that 0.4 multiplied by the output of the node **2005** is used as an input to the node **2015**. As another example and not by way of limitation, the output  $y_k$  of node k may be  $y_k=F_k(s_k)$ , where  $F_k$  may be the activation function corresponding to node k,  $s_k=\sum_j(w_{jk}x_j)$  may be the effective input to node k,  $x_j$  may be the output of a node j connected to node k, and  $w_{jk}$  may be

the weighting coefficient between node j and node k. In particular embodiments, the input to nodes of the input layer may be based on a vector representing an object. Although this disclosure describes particular inputs to and outputs of nodes, this disclosure contemplates any suitable inputs to and outputs of nodes. Moreover, although this disclosure may describe particular connections and weights between nodes, this disclosure contemplates any suitable connections and weights between nodes.

[**0214**] In particular embodiments, an ANN may be trained using training data. As an example and not by way of limitation, training data may comprise inputs to the ANN **2000** and an expected output. As another example and not by way of limitation, training data may comprise vectors each representing a training object and an expected label for each training object. In particular embodiments, training an ANN may comprise modifying the weights associated with the connections between nodes of the ANN by optimizing an objective function. As an example and not by way of limitation, a training method may be used (e.g., the conjugate gradient method, the gradient descent method, the stochastic gradient descent) to backpropagate the sum-of-squares error measured as a distances between each vector representing a training object (e.g., using a cost function that minimizes the sum-of-squares error). In particular embodiments, an ANN may be trained using a dropout technique. As an example and not by way of limitation, one or more nodes may be temporarily omitted (e.g., receive no input and generate no output) while training. For each training object, one or more nodes of the ANN may have some probability of being omitted. The nodes that are omitted for a particular training object may be different than the nodes omitted for other training objects (e.g., the nodes may be temporarily omitted on an object-by-object basis). Although this disclosure describes training an ANN in a particular manner, this disclosure contemplates training an ANN in any suitable manner.

Privacy

[**0215**] In particular embodiments, one or more objects (e.g., content or other types of objects) of a computing system may be associated with one or more privacy settings. The one or more objects may be stored on or otherwise associated with any suitable computing system or application, such as, for example, a social-networking system **160**, a client system **130**, an assistant system **140**, a third-party system **170**, a social-networking application, an assistant application, a messaging application, a photo-sharing application, or any other suitable computing system or application. Although the examples discussed herein are in the context of an online social network, these privacy settings may be applied to any other suitable computing system. Privacy settings (or “access settings”) for an object may be stored in any suitable manner, such as, for example, in association with the object, in an index on an authorization server, in another suitable manner, or any suitable combination thereof. A privacy setting for an object may specify how the object (or particular information associated with the object) can be accessed, stored, or otherwise used (e.g., viewed, shared, modified, copied, executed, surfaced, or identified) within the online social network. When privacy settings for an object allow a particular user or other entity to access that object, the object may be described as being “visible” with respect to that user or other entity. As an

example and not by way of limitation, a user of the online social network may specify privacy settings for a user-profile page that identify a set of users that may access work-experience information on the user-profile page, thus excluding other users from accessing that information.

**[0216]** In particular embodiments, privacy settings for an object may specify a “blocked list” of users or other entities that should not be allowed to access certain information associated with the object. In particular embodiments, the blocked list may include third-party entities. The blocked list may specify one or more users or entities for which an object is not visible. As an example and not by way of limitation, a user may specify a set of users who may not access photo albums associated with the user, thus excluding those users from accessing the photo albums (while also possibly allowing certain users not within the specified set of users to access the photo albums). In particular embodiments, privacy settings may be associated with particular social-graph elements. Privacy settings of a social-graph element, such as a node or an edge, may specify how the social-graph element, information associated with the social-graph element, or objects associated with the social-graph element can be accessed using the online social network. As an example and not by way of limitation, a particular photo may have a privacy setting specifying that the photo may be accessed only by users tagged in the photo and friends of the users tagged in the photo. In particular embodiments, privacy settings may allow users to opt in to or opt out of having their content, information, or actions stored/logged by the social-networking system **160** or assistant system **140** or shared with other systems (e.g., a third-party system **170**). Although this disclosure describes using particular privacy settings in a particular manner, this disclosure contemplates using any suitable privacy settings in any suitable manner.

**[0217]** In particular embodiments, privacy settings may be based on one or more nodes or edges of a social graph **1800**. A privacy setting may be specified for one or more edges **1806** or edge-types of the social graph **1800**, or with respect to one or more nodes **1802**, **1804** or node-types of the social graph **1800**. The privacy settings applied to a particular edge **1806** connecting two nodes may control whether the relationship between the two entities corresponding to the nodes is visible to other users of the online social network. Similarly, the privacy settings applied to a particular node may control whether the user or concept corresponding to the node is visible to other users of the online social network. As an example and not by way of limitation, a first user may share an object to the social-networking system **160**. The object may be associated with a concept node **1804** connected to a user node **1802** of the first user by an edge **1806**. The first user may specify privacy settings that apply to a particular edge **1806** connecting to the concept node **1804** of the object, or may specify privacy settings that apply to all edges **1806** connecting to the concept node **1804**. As another example and not by way of limitation, the first user may share a set of objects of a particular object-type (e.g., a set of images). The first user may specify privacy settings with respect to all objects associated with the first user of that particular object-type as having a particular privacy setting (e.g., specifying that all images posted by the first user are visible only to friends of the first user and/or users tagged in the images).

**[0218]** In particular embodiments, the social-networking system **160** may present a “privacy wizard” (e.g., within a

webpage, a module, one or more dialog boxes, or any other suitable interface) to the first user to assist the first user in specifying one or more privacy settings. The privacy wizard may display instructions, suitable privacy-related information, current privacy settings, one or more input fields for accepting one or more inputs from the first user specifying a change or confirmation of privacy settings, or any suitable combination thereof. In particular embodiments, the social-networking system **160** may offer a “dashboard” functionality to the first user that may display, to the first user, current privacy settings of the first user. The dashboard functionality may be displayed to the first user at any appropriate time (e.g., following an input from the first user summoning the dashboard functionality, following the occurrence of a particular event or trigger action). The dashboard functionality may allow the first user to modify one or more of the first user’s current privacy settings at any time, in any suitable manner (e.g., redirecting the first user to the privacy wizard).

**[0219]** Privacy settings associated with an object may specify any suitable granularity of permitted access or denial of access. As an example and not by way of limitation, access or denial of access may be specified for particular users (e.g., only me, my roommates, my boss), users within a particular degree-of-separation (e.g., friends, friends-of-friends), user groups (e.g., the gaming club, my family), user networks (e.g., employees of particular employers, students or alumni of particular university), all users (“public”), no users (“private”), users of third-party systems **170**, particular applications (e.g., third-party applications, external websites), other suitable entities, or any suitable combination thereof. Although this disclosure describes particular granularities of permitted access or denial of access, this disclosure contemplates any suitable granularities of permitted access or denial of access.

**[0220]** In particular embodiments, one or more servers **162** may be authorization/privacy servers for enforcing privacy settings. In response to a request from a user (or other entity) for a particular object stored in a data store **164**, the social-networking system **160** may send a request to the data store **164** for the object. The request may identify the user associated with the request and the object may be sent only to the user (or a client system **130** of the user) if the authorization server determines that the user is authorized to access the object based on the privacy settings associated with the object. If the requesting user is not authorized to access the object, the authorization server may prevent the requested object from being retrieved from the data store **164** or may prevent the requested object from being sent to the user. In the search-query context, an object may be provided as a search result only if the querying user is authorized to access the object, e.g., if the privacy settings for the object allow it to be surfaced to, discovered by, or otherwise visible to the querying user. In particular embodiments, an object may represent content that is visible to a user through a newsfeed of the user. As an example and not by way of limitation, one or more objects may be visible to a user’s “Trending” page. In particular embodiments, an object may correspond to a particular user. The object may be content associated with the particular user, or may be the particular user’s account or information stored on the social-networking system **160**, or other computing system. As an example and not by way of limitation, a first user may view one or more second users of an online social network through a “People You May Know” function of the online

social network, or by viewing a list of friends of the first user. As an example and not by way of limitation, a first user may specify that they do not wish to see objects associated with a particular second user in their newsfeed or friends list. If the privacy settings for the object do not allow it to be surfaced to, discovered by, or visible to the user, the object may be excluded from the search results. Although this disclosure describes enforcing privacy settings in a particular manner, this disclosure contemplates enforcing privacy settings in any suitable manner.

[0221] In particular embodiments, different objects of the same type associated with a user may have different privacy settings. Different types of objects associated with a user may have different types of privacy settings. As an example and not by way of limitation, a first user may specify that the first user's status updates are public, but any images shared by the first user are visible only to the first user's friends on the online social network. As another example and not by way of limitation, a user may specify different privacy settings for different types of entities, such as individual users, friends-of-friends, followers, user groups, or corporate entities. As another example and not by way of limitation, a first user may specify a group of users that may view videos posted by the first user, while keeping the videos from being visible to the first user's employer. In particular embodiments, different privacy settings may be provided for different user groups or user demographics. As an example and not by way of limitation, a first user may specify that other users who attend the same university as the first user may view the first user's pictures, but that other users who are family members of the first user may not view those same pictures.

[0222] In particular embodiments, the social-networking system 160 may provide one or more default privacy settings for each object of a particular object-type. A privacy setting for an object that is set to a default may be changed by a user associated with that object. As an example and not by way of limitation, all images posted by a first user may have a default privacy setting of being visible only to friends of the first user and, for a particular image, the first user may change the privacy setting for the image to be visible to friends and friends-of-friends.

[0223] In particular embodiments, privacy settings may allow a first user to specify (e.g., by opting out, by not opting in) whether the social-networking system 160 or assistant system 140 may receive, collect, log, or store particular objects or information associated with the user for any purpose. In particular embodiments, privacy settings may allow the first user to specify whether particular applications or processes may access, store, or use particular objects or information associated with the user. The privacy settings may allow the first user to opt in or opt out of having objects or information accessed, stored, or used by specific applications or processes. The social-networking system 160 or assistant system 140 may access such information in order to provide a particular function or service to the first user, without the social-networking system 160 or assistant system 140 having access to that information for any other purposes. Before accessing, storing, or using such objects or information, the social-networking system 160 or assistant system 140 may prompt the user to provide privacy settings specifying which applications or processes, if any, may access, store, or use the object or information prior to allowing any such action. As an example and not by way of

limitation, a first user may transmit a message to a second user via an application related to the online social network (e.g., a messaging app), and may specify privacy settings that such messages should not be stored by the social-networking system 160 or assistant system 140.

[0224] In particular embodiments, a user may specify whether particular types of objects or information associated with the first user may be accessed, stored, or used by the social-networking system 160 or assistant system 140. As an example and not by way of limitation, the first user may specify that images sent by the first user through the social-networking system 160 or assistant system 140 may not be stored by the social-networking system 160 or assistant system 140. As another example and not by way of limitation, a first user may specify that messages sent from the first user to a particular second user may not be stored by the social-networking system 160 or assistant system 140. As yet another example and not by way of limitation, a first user may specify that all objects sent via a particular application may be saved by the social-networking system 160 or assistant system 140.

[0225] In particular embodiments, privacy settings may allow a first user to specify whether particular objects or information associated with the first user may be accessed from particular client systems 130 or third-party systems 170. The privacy settings may allow the first user to opt in or opt out of having objects or information accessed from a particular device (e.g., the phone book on a user's smart phone), from a particular application (e.g., a messaging app), or from a particular system (e.g., an email server). The social-networking system 160 or assistant system 140 may provide default privacy settings with respect to each device, system, or application, and/or the first user may be prompted to specify a particular privacy setting for each context. As an example and not by way of limitation, the first user may utilize a location-services feature of the social-networking system 160 or assistant system 140 to provide recommendations for restaurants or other places in proximity to the user. The first user's default privacy settings may specify that the social-networking system 160 or assistant system 140 may use location information provided from a client system 130 of the first user to provide the location-based services, but that the social-networking system 160 or assistant system 140 may not store the location information of the first user or provide it to any third-party system 170. The first user may then update the privacy settings to allow location information to be used by a third-party image-sharing application in order to geo-tag photos.

[0226] In particular embodiments, privacy settings may allow a user to specify one or more geographic locations from which objects can be accessed. Access or denial of access to the objects may depend on the geographic location of a user who is attempting to access the objects. As an example and not by way of limitation, a user may share an object and specify that only users in the same city may access or view the object. As another example and not by way of limitation, a first user may share an object and specify that the object is visible to second users only while the first user is in a particular location. If the first user leaves the particular location, the object may no longer be visible to the second users. As another example and not by way of limitation, a first user may specify that an object is visible only to second users within a threshold distance from the first user. If the first user subsequently changes location, the

original second users with access to the object may lose access, while a new group of second users may gain access as they come within the threshold distance of the first user. [0227] In particular embodiments, the social-networking system 160 or assistant system 140 may have functionalities that may use, as inputs, personal or biometric information of a user for user-authentication or experience-personalization purposes. A user may opt to make use of these functionalities to enhance their experience on the online social network. As an example and not by way of limitation, a user may provide personal or biometric information to the social-networking system 160 or assistant system 140. The user's privacy settings may specify that such information may be used only for particular processes, such as authentication, and further specify that such information may not be shared with any third-party system 170 or used for other processes or applications associated with the social-networking system 160 or assistant system 140. As another example and not by way of limitation, the social-networking system 160 may provide a functionality for a user to provide voice-print recordings to the online social network. As an example and not by way of limitation, if a user wishes to utilize this function of the online social network, the user may provide a voice recording of his or her own voice to provide a status update on the online social network. The recording of the voice-input may be compared to a voice print of the user to determine what words were spoken by the user. The user's privacy setting may specify that such voice recording may be used only for voice-input purposes (e.g., to authenticate the user, to send voice messages, to improve voice recognition in order to use voice-operated features of the online social network), and further specify that such voice recording may not be shared with any third-party system 170 or used by other processes or applications associated with the social-networking system 160.

#### Systems and Methods

[0228] FIG. 21 illustrates an example computer system 2100. In particular embodiments, one or more computer systems 2100 perform one or more steps of one or more methods described or illustrated herein. In particular embodiments, one or more computer systems 2100 provide functionality described or illustrated herein. In particular embodiments, software running on one or more computer systems 2100 performs one or more steps of one or more methods described or illustrated herein or provides functionality described or illustrated herein. Particular embodiments include one or more portions of one or more computer systems 2100. Herein, reference to a computer system may encompass a computing device, and vice versa, where appropriate. Moreover, reference to a computer system may encompass one or more computer systems, where appropriate.

[0229] This disclosure contemplates any suitable number of computer systems 2100. This disclosure contemplates computer system 2100 taking any suitable physical form. As example and not by way of limitation, computer system 2100 may be an embedded computer system, a system-on-chip (SOC), a single-board computer system (SBC) (such as, for example, a computer-on-module (COM) or system-on-module (SOM)), a desktop computer system, a laptop or notebook computer system, an interactive kiosk, a mainframe, a mesh of computer systems, a mobile telephone, a personal digital assistant (PDA), a server, a tablet computer

system, or a combination of two or more of these. Where appropriate, computer system 2100 may include one or more computer systems 2100; be unitary or distributed; span multiple locations; span multiple machines; span multiple data centers; or reside in a cloud, which may include one or more cloud components in one or more networks. Where appropriate, one or more computer systems 2100 may perform without substantial spatial or temporal limitation one or more steps of one or more methods described or illustrated herein. As an example and not by way of limitation, one or more computer systems 2100 may perform in real time or in batch mode one or more steps of one or more methods described or illustrated herein. One or more computer systems 2100 may perform at different times or at different locations one or more steps of one or more methods described or illustrated herein, where appropriate.

[0230] In particular embodiments, computer system 2100 includes a processor 2102, memory 2104, storage 2106, an input/output (I/O) interface 2108, a communication interface 2110, and a bus 2112. Although this disclosure describes and illustrates a particular computer system having a particular number of particular components in a particular arrangement, this disclosure contemplates any suitable computer system having any suitable number of any suitable components in any suitable arrangement.

[0231] In particular embodiments, processor 2102 includes hardware for executing instructions, such as those making up a computer program. As an example and not by way of limitation, to execute instructions, processor 2102 may retrieve (or fetch) the instructions from an internal register, an internal cache, memory 2104, or storage 2106; decode and execute them; and then write one or more results to an internal register, an internal cache, memory 2104, or storage 2106. In particular embodiments, processor 2102 may include one or more internal caches for data, instructions, or addresses. This disclosure contemplates processor 2102 including any suitable number of any suitable internal caches, where appropriate. As an example and not by way of limitation, processor 2102 may include one or more instruction caches, one or more data caches, and one or more translation lookaside buffers (TLBs). Instructions in the instruction caches may be copies of instructions in memory 2104 or storage 2106, and the instruction caches may speed up retrieval of those instructions by processor 2102. Data in the data caches may be copies of data in memory 2104 or storage 2106 for instructions executing at processor 2102 to operate on; the results of previous instructions executed at processor 2102 for access by subsequent instructions executing at processor 2102 or for writing to memory 2104 or storage 2106; or other suitable data. The data caches may speed up read or write operations by processor 2102. The TLBs may speed up virtual-address translation for processor 2102. In particular embodiments, processor 2102 may include one or more internal registers for data, instructions, or addresses. This disclosure contemplates processor 2102 including any suitable number of any suitable internal registers, where appropriate. Where appropriate, processor 2102 may include one or more arithmetic logic units (ALUs); be a multi-core processor; or include one or more processors 2102. Although this disclosure describes and illustrates a particular processor, this disclosure contemplates any suitable processor.

[0232] In particular embodiments, memory 2104 includes main memory for storing instructions for processor 2102 to

execute or data for processor **2102** to operate on. As an example and not by way of limitation, computer system **2100** may load instructions from storage **2106** or another source (such as, for example, another computer system **2100**) to memory **2104**. Processor **2102** may then load the instructions from memory **2104** to an internal register or internal cache. To execute the instructions, processor **2102** may retrieve the instructions from the internal register or internal cache and decode them. During or after execution of the instructions, processor **2102** may write one or more results (which may be intermediate or final results) to the internal register or internal cache. Processor **2102** may then write one or more of those results to memory **2104**. In particular embodiments, processor **2102** executes only instructions in one or more internal registers or internal caches or in memory **2104** (as opposed to storage **2106** or elsewhere) and operates only on data in one or more internal registers or internal caches or in memory **2104** (as opposed to storage **2106** or elsewhere). One or more memory buses (which may each include an address bus and a data bus) may couple processor **2102** to memory **2104**. Bus **2112** may include one or more memory buses, as described below. In particular embodiments, one or more memory management units (MMUs) reside between processor **2102** and memory **2104** and facilitate accesses to memory **2104** requested by processor **2102**. In particular embodiments, memory **2104** includes random access memory (RAM). This RAM may be volatile memory, where appropriate. Where appropriate, this RAM may be dynamic RAM (DRAM) or static RAM (SRAM). Moreover, where appropriate, this RAM may be single-ported or multi-ported RAM. This disclosure contemplates any suitable RAM. Memory **2104** may include one or more memories **2104**, where appropriate. Although this disclosure describes and illustrates particular memory, this disclosure contemplates any suitable memory.

**[0233]** In particular embodiments, storage **2106** includes mass storage for data or instructions. As an example and not by way of limitation, storage **2106** may include a hard disk drive (HDD), a floppy disk drive, flash memory, an optical disc, a magneto-optical disc, magnetic tape, or a Universal Serial Bus (USB) drive or a combination of two or more of these. Storage **2106** may include removable or non-removable (or fixed) media, where appropriate. Storage **2106** may be internal or external to computer system **2100**, where appropriate. In particular embodiments, storage **2106** is non-volatile, solid-state memory. In particular embodiments, storage **2106** includes read-only memory (ROM). Where appropriate, this ROM may be mask-programmed ROM, programmable ROM (PROM), erasable PROM (EPROM), electrically erasable PROM (EEPROM), electrically alterable ROM (EAROM), or flash memory or a combination of two or more of these. This disclosure contemplates mass storage **2106** taking any suitable physical form. Storage **2106** may include one or more storage control units facilitating communication between processor **2102** and storage **2106**, where appropriate. Where appropriate, storage **2106** may include one or more storages **2106**. Although this disclosure describes and illustrates particular storage, this disclosure contemplates any suitable storage.

**[0234]** In particular embodiments, I/O interface **2108** includes hardware, software, or both, providing one or more interfaces for communication between computer system **2100** and one or more I/O devices. Computer system **2100** may include one or more of these I/O devices, where

appropriate. One or more of these I/O devices may enable communication between a person and computer system **2100**. As an example and not by way of limitation, an I/O device may include a keyboard, keypad, microphone, monitor, mouse, printer, scanner, speaker, still camera, stylus, tablet, touch screen, trackball, video camera, another suitable I/O device or a combination of two or more of these. An I/O device may include one or more sensors. This disclosure contemplates any suitable I/O devices and any suitable I/O interfaces **2108** for them. Where appropriate, I/O interface **2108** may include one or more device or software drivers enabling processor **2102** to drive one or more of these I/O devices. I/O interface **2108** may include one or more I/O interfaces **2108**, where appropriate. Although this disclosure describes and illustrates a particular I/O interface, this disclosure contemplates any suitable I/O interface.

**[0235]** In particular embodiments, communication interface **2110** includes hardware, software, or both providing one or more interfaces for communication (such as, for example, packet-based communication) between computer system **2100** and one or more other computer systems **2100** or one or more networks. As an example and not by way of limitation, communication interface **2110** may include a network interface controller (NIC) or network adapter for communicating with an Ethernet or other wire-based network or a wireless NIC (WNIC) or wireless adapter for communicating with a wireless network, such as a WI-FI network. This disclosure contemplates any suitable network and any suitable communication interface **2110** for it. As an example and not by way of limitation, computer system **2100** may communicate with an ad hoc network, a personal area network (PAN), a local area network (LAN), a wide area network (WAN), a metropolitan area network (MAN), or one or more portions of the Internet or a combination of two or more of these. One or more portions of one or more of these networks may be wired or wireless. As an example, computer system **2100** may communicate with a wireless PAN (WPAN) (such as, for example, a BLUETOOTH WPAN), a WI-FI network, a WI-MAX network, a cellular telephone network (such as, for example, a Global System for Mobile Communications (GSM) network), or other suitable wireless network or a combination of two or more of these. Computer system **2100** may include any suitable communication interface **2110** for any of these networks, where appropriate. Communication interface **2110** may include one or more communication interfaces **2110**, where appropriate. Although this disclosure describes and illustrates a particular communication interface, this disclosure contemplates any suitable communication interface.

**[0236]** In particular embodiments, bus **2112** includes hardware, software, or both coupling components of computer system **2100** to each other. As an example and not by way of limitation, bus **2112** may include an Accelerated Graphics Port (AGP) or other graphics bus, an Enhanced Industry Standard Architecture (EISA) bus, a front-side bus (FSB), a HYPERTRANSPORT (HT) interconnect, an Industry Standard Architecture (ISA) bus, an INFINIBAND interconnect, a low-pin-count (LPC) bus, a memory bus, a Micro Channel Architecture (MCA) bus, a Peripheral Component Interconnect (PCI) bus, a PCI-Express (PCIe) bus, a serial advanced technology attachment (SATA) bus, a Video Electronics Standards Association local (VLB) bus, or another suitable bus or a combination of two or more of these. Bus **2112** may include one or more buses **2112**, where appropriate.

Although this disclosure describes and illustrates a particular bus, this disclosure contemplates any suitable bus or interconnect.

**[0237]** Herein, a computer-readable non-transitory storage medium or media may include one or more semiconductor-based or other integrated circuits (ICs) (such, as for example, field-programmable gate arrays (FPGAs) or application-specific ICs (ASICs)), hard disk drives (HDDs), hybrid hard drives (HHDs), optical discs, optical disc drives (ODDs), magneto-optical discs, magneto-optical drives, floppy diskettes, floppy disk drives (FDDs), magnetic tapes, solid-state drives (SSDs), RAM-drives, SECURE DIGITAL cards or drives, any other suitable computer-readable non-transitory storage media, or any suitable combination of two or more of these, where appropriate. A computer-readable non-transitory storage medium may be volatile, non-volatile, or a combination of volatile and non-volatile, where appropriate.

#### Miscellaneous

**[0238]** Herein, “or” is inclusive and not exclusive, unless expressly indicated otherwise or indicated otherwise by context. Therefore, herein, “A or B” means “A, B, or both,” unless expressly indicated otherwise or indicated otherwise by context. Moreover, “and” is both joint and several, unless expressly indicated otherwise or indicated otherwise by context. Therefore, herein, “A and B” means “A and B, jointly or severally,” unless expressly indicated otherwise or indicated otherwise by context.

**[0239]** The scope of this disclosure encompasses all changes, substitutions, variations, alterations, and modifications to the example embodiments described or illustrated herein that a person having ordinary skill in the art would comprehend. The scope of this disclosure is not limited to the example embodiments described or illustrated herein. Moreover, although this disclosure describes and illustrates respective embodiments herein as including particular components, elements, feature, functions, operations, or steps, any of these embodiments may include any combination or permutation of any of the components, elements, features, functions, operations, or steps described or illustrated anywhere herein that a person having ordinary skill in the art would comprehend. Furthermore, reference in the appended claims to an apparatus or system or a component of an apparatus or system being adapted to, arranged to, capable of, configured to, enabled to, operable to, or operative to perform a particular function encompasses that apparatus, system, component, whether or not it or that particular function is activated, turned on, or unlocked, as long as that apparatus, system, or component is so adapted, arranged, capable, configured, enabled, operable, or operative. Additionally, although this disclosure describes or illustrates particular embodiments as providing particular advantages, particular embodiments may provide none, some, or all of these advantages.

What is claimed is:

1. A method comprising, by one or more computing systems:

receiving, from a first client system associated with a first user, a voice command for changing a device setting associated with a second client system, wherein the second client system is the first client system or a different client system;

accessing a registry storing device settings associated with a plurality of client systems, wherein the plurality of client systems comprise the second client system;

identifying the device setting associated with the second client system from the registry;

changing the device setting based on the voice command; and

sending, to the first client system responsive to the voice command, a status of changing the device setting associated with the second client system.

2. A method comprising, by an extended reality (XR) display device:

receiving, by the XR display device, an audio input from a first user of the XR display device, wherein the XR display device is associated with an XR environment comprising a plurality of XR objects;

processing, using a natural language understanding (NLU) model, the audio input to identify one or more intents and one or more slots associated with the audio input;

identifying a first XR object from the plurality of XR objects that is in an active listening state, wherein the first XR object is associated with a first set of intents and a first set of slots;

determining that either the first set of intents or the first set of slots do not comprise the one or more identified intents or the one or more identified slots associated with the audio input; and

generating, using a large language model (LLM), an out-of-domain (OOD) response based on one or more characteristics of the first XR object, wherein the OOD response references one or more of the one or more identified intents or the one or more identified slots associated with the audio input.

3. A method comprising, by one or more computing systems:

receiving, from a client system, one or more utterances comprising one or more first words in a first language and one or more second words in a second language;

generating, based on a single bilingual automatic-speech-recognition (ASR) model, a transcription of the one or more utterances, wherein the transcription comprises one or more first text strings in the first language and one or more second text strings in the second language;

executing one or more tasks based on the one or more first text strings in the first language and the one or more second text strings in the second language; and

sending, to the client system, instructions for presenting a response responsive to the one or more utterances, wherein the response is based on both the first and second languages.

\* \* \* \* \*