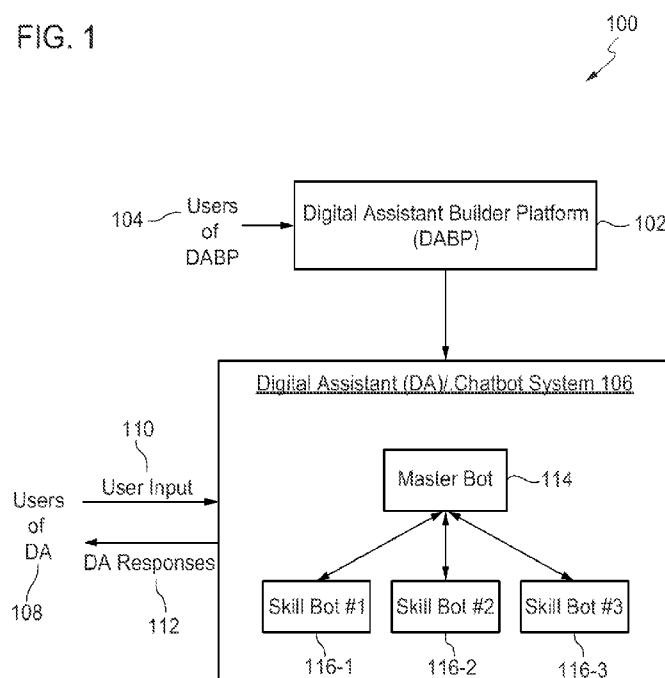




- (51) International Patent Classification:
G06N 3/08 (2006.01) *G06N 20/00* (2019.01)
- (21) International Application Number:
PCT/US2022/040530
- (22) International Filing Date:
17 August 2022 (17.08.2022)
- (25) Filing Language: English
- (26) Publication Language: English
- (30) Priority Data:
63/254,740 12 October 2021 (12.10.2021) US
17/735,651 03 May 2022 (03.05.2022) US
- (71) Applicant: **ORACLE INTERNATIONAL CORPORATION** [US/US]; 500 Oracle Parkway M/S 50P7, Redwood Shores, California 94065 (US).
- (72) Inventors: **VU, Thanh Tien**; 500 Oracle Parkway M/S 50P7, Redwood Shores, California 94065 (US). **PHAM, Thomas**; 500 Oracle Parkway M/S 50P7, Redwood Shores, California 94065 (US). **NEZAMI, Omid Mohamad**; 500 Oracle Parkway M/S 50P7, Redwood Shores, California 94065 (US). **JOHNSON, Mark Edward**; 500 Oracle Parkway M/S 50P7, Redwood Shores, California 94065 (US). **DUONG, Thanh Long**; 500 Oracle Parkway M/S 50P7, Redwood Shores, California 94065 (US). **HOANG, Cong Duy Vu**; 500 Oracle Parkway M/S 50P7, Redwood Shores, California 94065 (US).
- (74) Agent: **GAUDRY, Katherine** et al.; 1100 Peachtree Street NE, Suite 2800, Atlanta, Georgia 30309 (US).
- (81) Designated States (*unless otherwise indicated, for every kind of national protection available*): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CV, CZ, DE, DJ, DK, DM,

(54) Title: FINE-TUNING MULTI-HEAD NETWORK FROM A SINGLE TRANSFORMER LAYER OF PRE-TRAINED LANGUAGE MODEL

FIG. 1



(57) **Abstract:** Techniques are provided for customizing or fine-tuning a pre-trained version of a machine-learning model that includes multiple layers and is configured to process audio or textual language input. Each of the multiple layers is configured with a plurality of layer-specific pre-trained parameter values corresponding to a plurality of parameters, and each of the multiple layers is configured to implement multi-head attention. An incomplete subset of the multiple layers is identified for which corresponding layer-specific pre-trained parameter values are to be fine-tuned using a client data set. The machine-learning model is fine-tuned using the client data set to generate an updated version of the machine-learning model, where the layer-specific pre-trained parameter values configured for each layer of one of more of the multiple layers not included in the incomplete subset are frozen during the fine-tuning. Use of the updated version of the machine-learning model is facilitated.

DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IQ, IR, IS, IT, JM, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

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

Published:

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

FINE-TUNING MULTI-HEAD NETWORK FROM A SINGLE TRANSFORMER LAYER OF PRE-TRAINED LANGUAGE MODEL

CROSS-REFERENCE TO RELATED APPLICATION

- 5 **[0001]** This application claim priority to U.S. Non-Provisional Application No. 17/735,651, filed May 3, 2022, which claims the benefit of and the priority to U.S. Provisional Patent Application 63/254,740, filed on October 12, 2021. Each of these applications is hereby incorporated by reference in its entirety for all purposes.

FIELD OF THE INVENTION

- 10 **[0002]** The present disclosure relates generally to selectively fine-tuning select layers of a pre-trained language model. More specifically, the selective fine-tuning includes conducting a training that includes updating parameters within an incomplete subset of a set of layers within the pre-trained language model to be updated, while freezing parameters within one or more other layers of the pre-trained language model during the training.

BACKGROUND

- 15 **[0003]** Many users around the world are on instant messaging or chat platforms in order to get instant reaction. Organizations often use these instant messaging or chat platforms to engage with customers (or end users) in live conversations. However, it can be very costly for organizations to employ service people to engage in live communication with customers or
20 end users. Chatbots or bots have begun to be developed to simulate conversations with end users, especially over the Internet. End users can communicate with bots through messaging apps that the end users have already installed and used. An intelligent bot, generally powered by artificial intelligence (AI), can communicate more intelligently and contextually in live conversations, and thus may allow for a more natural conversation between the bot and the
25 end users for improved conversational experience. Instead of the end user learning a fixed set of keywords or commands that the bot knows how to respond to, an intelligent bot may be able to understand the end user's intention based upon user utterances in natural language and respond accordingly.

- 30 **[0004]** However, chatbots are difficult to build because these automated solutions require specific knowledge in certain fields and the application of certain techniques that may be solely within the capabilities of specialized developers. As part of building such chatbots, a

developer may first understand the needs of enterprises and end users. The developer may then analyze and make decisions related to, for example, selecting data sets to be used for the analysis, preparing the input data sets for analysis (*e.g.*, cleansing the data, extracting, formatting, and/or transforming the data prior to analysis, performing data features engineering, etc.), identifying an appropriate machine learning (ML) technique(s) or model(s) for performing the analysis, and improving the technique or model to improve results/outcomes based upon feedback. The task of identifying an appropriate model may include developing multiple models, possibly in parallel, iteratively testing and experimenting with these models, before identifying a particular model (or models) for use. Further, supervised learning-based solutions typically involve a training phase, followed by an application (*i.e.*, inference) phase, and iterative loops between the training phase and the application phase. The developer may be responsible for carefully implementing and monitoring these phases to achieve optimal solutions.

[0005] Thus, it would be advantageous to identify a technique by which a model may be developed for a specific use case more efficiently.

BRIEF SUMMARY

[0006] Techniques disclosed herein relate generally to chatbots. More specifically and without limitation, techniques disclosed herein relate to training chatbots by accessing a pre-trained model and fine-tuning parameters in an incomplete subset of multiple layers in the model.

[0007] In various embodiments, a computer-implemented method is provided that includes: accessing a pre-trained version of a machine-learning model that includes multiple layers and is configured to process audio or textual language input, each of the multiple layers being configured with a plurality of layer-specific pre-trained parameter values corresponding to a plurality of parameters, and each of the multiple layers being configured to implement multi-head attention; accessing a client data set; identifying an incomplete subset of the multiple layers for which corresponding layer-specific pre-trained parameter values are to be fine-tuned using the client data set, fine-tuning the machine-learning model using the client data set to generate an updated version of the machine-learning model, wherein the layer-specific pre-trained parameter values configured for each layer of one of more of the multiple layers not included in the incomplete subset are frozen during the fine-tuning; and facilitating use of the updated version of the machine-learning model.

[0008] In some embodiments, the pre-trained machine-learning model includes a self-attention model, and where the incomplete subset of the multiple layers includes at least one neural network in the self-attention model.

5 [0009] In some embodiments, the incomplete subset of the multiple layers includes two layers or one layer, and wherein the pre-trained machine-learning model includes at least five layers.

[0010] In some embodiments, the plurality of parameters includes at least 500,000 parameters, and wherein the updating the plurality of parameter values is performed using one or more Central Processing Units.

10 [0011] In some embodiments, the updating of the parameter values includes: accessing a set of cached values corresponding to the one or more of the multiple layers not included in the incomplete subsets.

[0012] In some embodiments, a quantity of the plurality of parameter values configured for the incomplete subset of the multiple layers is 30% or less as compared to a quantity
15 parameters configured for the machine-learning model.

[0013] In some embodiments, facilitating use of the updated version includes translating linguistic input into a command or query.

[0014] In some embodiments, the multiple layers includes at least 4 layers, and wherein each of the incomplete subset of the multiple layers is positioned at Layer 2 or higher in the
20 multiple layers.

[0015] In various embodiments, a system is provided that includes one or more data processors and a non-transitory computer readable storage medium containing instructions which, when executed on the one or more data processors, cause the one or more data processors to perform part or all of one or more methods disclosed herein.

25 [0016] In various embodiments, a computer-program product is provided that is tangibly embodied in a non-transitory machine-readable storage medium and that includes instructions configured to cause one or more data processors to perform part or all of one or more methods disclosed herein.

[0017] The techniques described above and below may be implemented in a number of
30 ways and in a number of contexts. Several example implementations and contexts are

provided with reference to the following figures, as described below in more detail. However, the following implementations and contexts are but a few of many.

BRIEF DESCRIPTION OF THE DRAWINGS

5 [0018] FIG. 1 is a simplified block diagram of a distributed environment incorporating an exemplary embodiment.

[0019] FIG. 2 is a simplified block diagram of a computing system implementing a master bot according to certain embodiments.

[0020] FIG. 3 is a simplified block diagram of a computing system implementing a skill bot according to certain embodiments.

10 [0021] FIG. 4 is a simplified block diagram of a chatbot training and deployment system in accordance with various embodiments.

[0022] FIG. 5 illustrates a process flow for augmenting a training data set with keywords in accordance with various embodiments.

15 [0023] FIG. 6 depicts a simplified diagram of a distributed system for implementing various embodiments.

[0024] FIG. 7 is a simplified block diagram of one or more components of a system environment by which services provided by one or more components of an embodiment system may be offered as cloud services, in accordance with various embodiments.

20 [0025] FIG. 8 illustrates an example computer system that may be used to implement various embodiments.

[0026] FIGS. 9A-9C illustrate architectures of an exemplary model that can be finetuned in accordance with various embodiments.

[0027] FIG. 10 shows a table that presents data corresponding to various approaches for fine-tuning training of a model.

25 [0028] FIG. 11 illustrates layers in a model that may be pre-trained and fine-tuned in accordance with select embodiments of the invention.

DETAILED DESCRIPTION

[0029] In the following description, for the purposes of explanation, specific details are set forth in order to provide a thorough understanding of certain embodiments. However, it will be apparent that various embodiments may be practiced without these specific details. The figures and description are not intended to be restrictive. The word “exemplary” is used herein to mean “serving as an example, instance, or illustration.” Any embodiment or design described herein as “exemplary” is not necessarily to be construed as preferred or advantageous over other embodiments or designs.

Introduction

10 [0030] A digital assistant is an artificial intelligent driven interface that helps users accomplish a variety of tasks in natural language conversations. For each digital assistant, a customer may assemble one or more skills. Skills (also described herein as chatbots, bots, or skill bots) are individual bots that are focused on specific types of tasks, such as tracking inventory, submitting time cards, and creating expense reports. When an end user engages
15 with the digital assistant, the digital assistant evaluates the end user input and routes the conversation to and from the appropriate chatbot. The digital assistant can be made available to end users through a variety of channels such as FACEBOOK® Messenger, SKYPE MOBILE® messenger, or a Short Message Service (SMS). Channels carry the chat back and forth from end users on various messaging platforms to the digital assistant and its various
20 chatbots. The channels may also support user agent escalation, event-initiated conversations, and testing.

[0031] Intents allow the chatbot to understand what the user wants the chatbot to do. Intents are comprised of permutations of typical user requests and statements, which are also referred to as utterances (e.g., get account balance, make a purchase, etc.). As used herein, an
25 utterance or a message may refer to a set of words (e.g., one or more sentences) exchanged during a conversation with a chatbot. Intents may be created by providing a name that illustrates some user action (e.g., order a pizza) and compiling a set of real-life user statements, or utterances that are commonly associated with triggering the action. Because the chatbot’s cognition is derived from these intents, each intent may be created from a data
30 set that is robust (one to two dozen utterances) and varied, so that the chatbot may interpret ambiguous user input. A rich set of utterances enables a chatbot to understand what the user wants when it receives messages like “Forget this order!” or “Cancel delivery!”—messages

that mean the same thing, but are expressed differently. Collectively, the intents, and the utterances that belong to them, make up a training corpus for the chatbot. By training a model with the corpus, a customer may essentially turn that model into a reference tool for resolving end user input to a single intent. A customer can improve the acuity of the chatbot's cognition
5 through rounds of intent testing and intent training.

[0032] However, building a chatbot that can determine the end users' intents based upon user utterances, is a challenging task in part due to the subtleties and ambiguity of natural languages and the dimension of the input space (e.g., possible user utterances) and the size of the output space (number of intents). An illustrative example of this difficulty arises from
10 characteristics of natural language, such as employing euphemisms, synonyms, or ungrammatical speech to express intent. For example, an utterance may express an intent to order a pizza without explicitly mentioning pizza, ordering, or delivery explicitly. For example, in certain regional vernaculars, "pizza" is referred to as "pie." These tendencies in natural language, such as imprecision or variability, give rise to uncertainty and introduce
15 confidence as a parameter for prediction of intent, as opposed to explicit indication of intent, for example, through inclusion of keywords. As such, the chatbot may need to be trained, monitored, debugged, and retrained in order to improve the performance of the chatbot and user experience with the chatbot. In conventional systems, training systems are provided for training and retraining machine-learning models of the digital assistant or chatbot in spoken
20 language understanding (SLU) and natural language processing (NLP). Conventionally, the models used for chatbot systems are trained in NLP with "manufactured" utterances for any intent. For example, the utterance "Do you do price changes?" may be used to train a classifier model of a chatbot system to classify this type of utterance as the intent – "Do you offer a price match." The training of models with manufactured utterances helps initially train
25 the chatbot system for providing services and then the chatbot system may be re-trained once it is deployed and starts getting real utterances from users.

[0033] As part of the NLP processing for a utterance, the digital assistant is trained to understand the meaning of the utterance, which involves identifying one or more intents and one or more entities corresponding to the utterance. The entity extraction in the digital
30 assistant has two phases: named entity recognition and entity resolution via the named entity recognizer. The particular training problem addressed here pertains to named entity recognizer (NER). Entities can be understood as nouns, and are often referred to as slots. Entities are typically things like date, time, cities, names, brands etc., (a few common entities

used as examples herein are domain independent, called System Entities: PERSON, NUMBER, CURRENCY, DATE_TIME; however, it should be understood that this disclosure is not limited to these system entities and is applicable any entity type such as composite entities, entity roles, entity lists, and the like). For example with respect to a travel bot, capturing the cities of departure, destination, travel mode, price, dates and times is at the foundation of the interface. However, capturing the entities is a difficult task for the digital assistant because users enter data in various languages under various contexts and randomly in no particular order. The following Table 1 shows a couple of examples of the entity extraction problem input (utterances) and outputs (entities):

Table 1:

Input Utterance	Entity Extraction Output
\$60 for a cab ride on 12/2	CURRENCY: \$60 TIME_DATE: 12/2
show me rowland's directs	PERSON: rowland

Yet, capturing these entities are important for the digital assistant to take action based on the user's intent.

[0034] Conventional training of models for named entity recognition starts with pre-labeled data. However, the number of unique tokens (e.g., words, symbols, date representations, time representations, etc.) that a chatbot system may receive is extraordinarily higher, and the quantity of various ordered combinations of tokens is orders of magnitude higher. Given the large and complex input space, named entity recognizers include a very large number of parameters (e.g., hundreds of millions of parameters) in order to be able to accurately identify an classify entities of interest. Training a named entity recognizer to reliably detect different tokens and token combinations and to accurately map the tokens and token combinations to corresponding entities requires using a very large training set and a substantial amount of training time.

[0035] One approach for reducing training time and a size of a training data set is to use transfer learning. That is, a model can be initialized with parameter values that were learned in association with a first context, and the values can then be fine-tuned using training data associated with a second context. While transfer learning can offer tremendous efficiency advantages, the time and resources committed to the fine-tuning still depend on the number

of parameters for which values are being fine-tuned. Thus, in the chatbox-system context, even if a machine-learning model is trained using transfer learning, the training may still require a large size of training data and extensive commitment of time and computing resources for the training to achieve threshold performance metrics.

5 [0036] Accordingly, a different approach is needed to address these problems. Some embodiments described herein relate to defining a model for a given chatbot-system use case by fine-tuning a portion of a pre-trained model using client-specific data. The client-specific data may include data provided by an individual client, data associated with a given platform, data associated with a given use, data associated with a given context, and/or data associated
10 with a given time period.

[0037] In some embodiments, a machine-learning model is a NER and is trained using a training data set. The machine-learning model may include at least two, at least three, at least five, at least seven, at least ten, or at least fifteen layers. The machine-learning model may include at least two layers using a multi-head attention technique (e.g., including a multi-head
15 attention network), at least three layers using a multi-head attention technique, at least five layers using a multi-head attention technique, at least seven layers using a multi-head attention technique, at least ten layers using a multi-head attention technique, or at least fifteen layers using a multi-head attention technique. The multi-head attention technique can include a self-attention technique. Using the multi-head attention technique can include (for
20 example) using a self-attention model, using a multi-head model or using a transformer model. The machine-learning model can include a Bidirectional Encoder Representations from Transformers (BERT) model. The trained machine-learning model can include a multilingual BERT model. The trained machine-learning model can include one or more transformer layers (e.g., at least two transformer layers, at least four transformer layers, at
25 least six transformer layers, or at least eight transformer layers) and an embedding layer.

[0038] Training the machine-learning model can include learning a value for each of a set of parameters of the machine-learning model. The set of parameters may include (for example) at least 100,000 parameters, at least 200,000 parameters, at least 500,000 parameters, at least 1,000,000 parameters, at least 5,000,000 parameters, at least 10,000,000
30 parameters, at least 50,000,000 parameters, at least 100,000,000 parameters, at least 200,000,000 parameters, or at least 500,000,000 parameters.

[0039] The training data may be associated with (e.g., may have been provided by, accessed for, selected by, or automatically identified for) a given use case, a given combination of use cases, a given application, a given combination of applications, a given client, and/or a given combination of clients. Subsequently, the trained model may be used
5 for initialization of a model that is fine-tuned using a client data set that corresponds to a different use case, a different combination of use cases, a different application, a different combination of applications, a different client, and/or a different combination of clients relative to the training data set.

[0040] After the initialization, values of the parameters associated with: at least one layer,
10 at least two layers, at least three layers, at least five layers, at least eight layers may be at least temporarily frozen to prevent modification of the parameter values at least temporarily. Each of one, more, or all of the layers for which the values of the parameters were frozen may be at a lower level than each layer for which the values of the parameters were not frozen. For example, determining for which layers parameter values are to be frozen can include
15 selecting a threshold layer, such that values of parameters of layers below the threshold layer are to be frozen while values of parameters of the threshold layer and any layer above the threshold layer are not frozen (and can be fine-tuned using the client data set).

[0041] In some alternative or additional embodiments, for each layer of one or more of the model's layers (e.g., for each layer not selected for freezing parameter values), a first subset
20 of the parameters of the layer is identified for freezing parameter values and a second subset of the parameters of the layers is identified for refraining from freezing parameter values (such that the values can be fine-tuned using the client data set).

[0042] The first subset may include parameters of one or more networks and/or sub-models in the layer, and the second subset may include parameters of one or more other networks
25 and/or sub-models in the layer. For example, the first subset may include parameters associated with a multi-head attention network or a self-attention network, and the second subset may include a feed forward network.

[0043] One advantage of this approach is that the machine-learning model may be fine-tuned using less data than would be required to fine-tune all of the parameters in the
30 machine-learning model. Further, performing a fine-tuning of values of an incomplete subset of the parameters can be performed more quickly relative to fine-tuning values of all of the parameters.

[0044] If values of parameters of one or more layers below a threshold layer are frozen, intermediate values corresponding to outputs from the frozen layers can be calculated during one fine-tuning iteration, cached and used across other iterations in the fine-tuning. Thus, the model need not determine the intermediate values for each data set in the training data. This
5 caching can decrease the speed of the fine-tuning and decrease the computational resources used during the fine-tuning.

[0045] Thus, fine-tuning the machine-learning model in a manner in which values of only an incomplete subset of the model's parameters are adjusted has advantages over fine-tuning the model by adjusting values of more or all of the parameters. For example, the fine-tuning
10 can be performed using a smaller data set, more quickly, and/or using fewer computational resources when only an incomplete subset of the model's parameters are adjusted. By enabling caching of intermediate values, these advantages are achieved to an even higher degree. Nonetheless, the inventor has discovered that the accuracy of predictions generated by a machine-learning model for which values of only an incomplete subset of parameters
15 were updated during fine-tune rival the accuracy of a comparable model for which values of all parameters were updated during fine-tuning.

Bot and Analytic Systems

[0046] A bot (also referred to as a skill, chatbot, chatterbot, or talkbot) is a computer program that can perform conversations with end users. The bot can generally respond to natural-language messages (e.g., questions or comments) through a messaging application that uses natural-language messages. Enterprises may use one or more bot systems to communicate with end users through a messaging application. The messaging application, which may be referred to as a channel, may be an end user preferred messaging application that the end user has already installed and familiar with. Thus, the end user does not need to download and install new applications in order to chat with the bot system. The messaging application may include, for example, over-the-top (OTT) messaging channels (such as Facebook Messenger, Facebook WhatsApp, WeChat, Line, Kik, Telegram, Talk, Skype, Slack, or SMS), virtual private assistants (such as Amazon Dot, Echo, or Show, Google Home, Apple HomePod, etc.), mobile and web app extensions that extend native or hybrid/responsive mobile apps or web applications with chat capabilities, or voice based input (such as devices or apps with interfaces that use Siri, Cortana, Google Voice, or other speech input for interaction).

[0047] In some examples, a bot system may be associated with a Uniform Resource Identifier (URI). The URI may identify the bot system using a string of characters. The URI may be used as a webhook for one or more messaging application systems. The URI may include, for example, a Uniform Resource Locator (URL) or a Uniform Resource Name (URN). The bot system may be designed to receive a message (e.g., a hypertext transfer protocol (HTTP) post call message) from a messaging application system. The HTTP post call message may be directed to the URI from the messaging application system. In some embodiments, the message may be different from a HTTP post call message. For example, the bot system may receive a message from a Short Message Service (SMS). While discussion herein may refer to communications that the bot system receives as a message, it should be understood that the message may be an HTTP post call message, a SMS message, or any other type of communication between two systems.

[0048] End users may interact with the bot system through a conversational interaction (sometimes referred to as a conversational user interface (UI)), just as interactions between people. In some cases, the interaction may include the end user saying “Hello” to the bot and the bot responding with a “Hi” and asking the end user how it can help. In some cases, the

interaction may also be a transactional interaction with, for example, a banking bot, such as transferring money from one account to another; an informational interaction with, for example, a HR bot, such as checking for vacation balance; or an interaction with, for example, a retail bot, such as discussing returning purchased goods or seeking technical support.

[0049] In some embodiments, the bot system may intelligently handle end user interactions without interaction with an administrator or developer of the bot system. For example, an end user may send one or more messages to the bot system in order to achieve a desired goal. A message may include certain content, such as text, emojis, audio, image, video, or other method of conveying a message. In some embodiments, the bot system may convert the content into a standardized form (e.g., a representational state transfer (REST) call against enterprise services with the proper parameters) and generate a natural language response. The bot system may also prompt the end user for additional input parameters or request other additional information. In some embodiments, the bot system may also initiate communication with the end user, rather than passively responding to end user utterances. Described herein are various techniques for identifying an explicit invocation of a bot system and determining an input for the bot system being invoked. In certain embodiments, explicit invocation analysis is performed by a master bot based on detecting an invocation name in an utterance. In response to detection of the invocation name, the utterance may be refined for input to a skill bot associated with the invocation name.

[0050] A conversation with a bot may follow a specific conversation flow including multiple states. The flow may define what would happen next based on an input. In some embodiments, a state machine that includes user defined states (e.g., end user intents) and actions to take in the states or from state to state may be used to implement the bot system. A conversation may take different paths based on the end user input, which may impact the decision the bot makes for the flow. For example, at each state, based on the end user input or utterances, the bot may determine the end user's intent in order to determine the appropriate next action to take. As used herein and in the context of an utterance, the term "intent" refers to an intent of the user who provided the utterance. For example, the user may intend to engage a bot in conversation for ordering pizza, so that the user's intent could be represented through the utterance "Order pizza." A user intent can be directed to a particular task that the user wishes a chatbot to perform on behalf of the user. Therefore, utterances can be phrased

as questions, commands, requests, and the like, that reflect the user's intent. An intent may include a goal that the end user would like to accomplish.

[0051] In the context of the configuration of a chatbot, the term "intent" is used herein to refer to configuration information for mapping a user's utterance to a specific task/action or category of task/action that the chatbot can perform. In order to distinguish between the intent of an utterance (i.e., a user intent) and the intent of a chatbot, the latter is sometimes referred to herein as a "bot intent." A bot intent may comprise a set of one or more utterances associated with the intent. For instance, an intent for ordering pizza can have various permutations of utterances that express a desire to place an order for pizza. These associated utterances can be used to train an intent classifier of the chatbot to enable the intent classifier to subsequently determine whether an input utterance from a user matches the order pizza intent. A bot intent may be associated with one or more dialog flows for starting a conversation with the user and in a certain state. For example, the first message for the order pizza intent could be the question "What kind of pizza would you like?" In addition to associated utterances, a bot intent may further comprise named entities that relate to the intent. For example, the order pizza intent could include variables or parameters used to perform the task of ordering pizza, e.g., topping 1, topping 2, pizza type, pizza size, pizza quantity, and the like. The value of an entity is typically obtained through conversing with the user.

[0052] FIG. 1 is a simplified block diagram of an environment 100 incorporating a chatbot system according to certain embodiments. Environment 100 comprises a digital assistant builder platform (DABP) 102 that enables users of DABP 102 to create and deploy digital assistants or chatbot systems. DABP 102 can be used to create one or more digital assistants (or DAs) or chatbot systems. For example, as shown in Fig. 1, user 104 representing a particular enterprise can use DABP 102 to create and deploy a digital assistant 106 for users of the particular enterprise. For example, DABP 102 can be used by a bank to create one or more digital assistants for use by the bank's customers. The same DABP 102 platform can be used by multiple enterprises to create digital assistants. As another example, an owner of a restaurant (e.g., a pizza shop) may use DABP 102 to create and deploy a digital assistant that enables customers of the restaurant to order food (e.g., order pizza).

[0053] For purposes of this disclosure, a "digital assistant" is an entity that helps users of the digital assistant accomplish various tasks through natural language conversations. A

digital assistant can be implemented using software only (e.g., the digital assistant is a digital entity implemented using programs, code, or instructions executable by one or more processors), using hardware, or using a combination of hardware and software. A digital assistant can be embodied or implemented in various physical systems or devices, such as in a computer, a mobile phone, a watch, an appliance, a vehicle, and the like. A digital assistant is also sometimes referred to as a chatbot system. Accordingly, for purposes of this disclosure, the terms digital assistant and chatbot system are interchangeable.

[0054] A digital assistant, such as digital assistant 106 built using DABP 102, can be used to perform various tasks via natural language-based conversations between the digital assistant and its users 108. As part of a conversation, a user may provide one or more user inputs 110 to digital assistant 106 and get responses 112 back from digital assistant 106. A conversation can include one or more of inputs 110 and responses 112. Via these conversations, a user can request one or more tasks to be performed by the digital assistant and, in response, the digital assistant is configured to perform the user-requested tasks and respond with appropriate responses to the user.

[0055] User inputs 110 are generally in a natural language form and are referred to as utterances. A user utterance 110 can be in text form, such as when a user types in a sentence, a question, a text fragment, or even a single word and provides it as input to digital assistant 106. In some embodiments, a user utterance 110 can be in audio input or speech form, such as when a user says or speaks something that is provided as input to digital assistant 106. The utterances are typically in a language spoken by the user 108. For example, the utterances may be in English, or some other language. When an utterance is in speech form, the speech input is converted to text form utterances in that particular language and the text utterances are then processed by digital assistant 106. Various speech-to-text processing techniques may be used to convert a speech or audio input to a text utterance, which is then processed by digital assistant 106. In some embodiments, the speech-to-text conversion may be done by digital assistant 106 itself.

[0056] An utterance, which may be a text utterance or a speech utterance, can be a fragment, a sentence, multiple sentences, one or more words, one or more questions, combinations of the aforementioned types, and the like. Digital assistant 106 is configured to apply natural language understanding (NLU) techniques to the utterance to understand the meaning of the user input. As part of the NLU processing for a utterance, digital assistant 106

is configured to perform processing to understand the meaning of the utterance, which involves identifying one or more intents and one or more entities corresponding to the utterance. Upon understanding the meaning of an utterance, digital assistant 106 may perform one or more actions or operations responsive to the understood meaning or intents. For
5 purposes of this disclosure, it is assumed that the utterances are text utterances that have been provided directly by a user 108 of digital assistant 106 or are the results of conversion of input speech utterances to text form. This however is not intended to be limiting or restrictive in any manner.

[0057] For example, a user 108 input may request a pizza to be ordered by providing an
10 utterance such as "I want to order a pizza." Upon receiving such an utterance, digital assistant 106 is configured to understand the meaning of the utterance and take appropriate actions. The appropriate actions may involve, for example, responding to the user with questions requesting user input on the type of pizza the user desires to order, the size of the pizza, any toppings for the pizza, and the like. The responses provided by digital assistant 106 may also
15 be in natural language form and typically in the same language as the input utterance. As part of generating these responses, digital assistant 106 may perform natural language generation (NLG). For the user ordering a pizza, via the conversation between the user and digital assistant 106, the digital assistant may guide the user to provide all the requisite information for the pizza order, and then at the end of the conversation cause the pizza to be ordered.
20 Digital assistant 106 may end the conversation by outputting information to the user indicating that the pizza has been ordered.

[0058] At a conceptual level, digital assistant 106 performs various processing in response to an utterance received from a user. In some embodiments, this processing involves a series or pipeline of processing steps including, for example, understanding the meaning of the
25 input utterance (sometimes referred to as Natural Language Understanding (NLU), determining an action to be performed in response to the utterance, where appropriate causing the action to be performed, generating a response to be output to the user responsive to the user utterance, outputting the response to the user, and the like. The NLU processing can include parsing the received input utterance to understand the structure and meaning of the
30 utterance, refining and reforming the utterance to develop a better understandable form (e.g., logical form) or structure for the utterance. Generating a response may include using NLG techniques.

[0059] The NLU processing performed by a digital assistant, such as digital assistant 106, can include various NLP related processing such as sentence parsing (e.g., tokenizing, lemmatizing, identifying part-of-speech tags for the sentence, identifying named entities in the sentence, generating dependency trees to represent the sentence structure, splitting a sentence into clauses, analyzing individual clauses, resolving anaphoras, performing chunking, and the like). In certain embodiments, the NLU processing or portions thereof is performed by digital assistant 106 itself. In some other embodiments, digital assistant 106 may use other resources to perform portions of the NLU processing. For example, the syntax and structure of an input utterance sentence may be identified by processing the sentence using a parser, a part-of-speech tagger, and/or a named entity recognizer. In one implementation, for the English language, a parser, a part-of-speech tagger, and a named entity recognizer such as ones provided by the Stanford Natural Language Processing (NLP) Group are used for analyzing the sentence structure and syntax. These are provided as part of the Stanford CoreNLP toolkit.

[0060] While the various examples provided in this disclosure show utterances in the English language, this is meant only as an example. In certain embodiments, digital assistant 106 is also capable of handling utterances in languages other than English. Digital assistant 106 may provide subsystems (e.g., components implementing NLU functionality) that are configured for performing processing for different languages. These subsystems may be implemented as pluggable units that can be called using service calls from an NLU core server. This makes the NLU processing flexible and extensible for each language, including allowing different orders of processing. A language pack may be provided for individual languages, where a language pack can register a list of subsystems that can be served from the NLU core server.

[0061] A digital assistant, such as digital assistant 106 depicted in FIG. 1, can be made available or accessible to its users 108 through a variety of different channels, such as but not limited to, via certain applications, via social media platforms, via various messaging services and applications, and other applications or channels. A single digital assistant can have several channels configured for it so that it can be run on and be accessed by different services simultaneously.

[0062] A digital assistant or chatbot system generally contains or is associated with one or more skills. In certain embodiments, these skills are individual chatbots (referred to as skill

bots) that are configured to interact with users and fulfill specific types of tasks, such as tracking inventory, submitting timecards, creating expense reports, ordering food, checking a bank account, making reservations, buying a widget, and the like. For example, for the embodiment depicted in Fig. 1, digital assistant or chatbot system 106 includes skills 116-1, 116-2, and so on. For purposes of this disclosure, the terms "skill" and "skills" are used synonymously with the terms "skill bot" and "skill bots," respectively.

[0063] Each skill associated with a digital assistant helps a user of the digital assistant complete a task through a conversation with the user, where the conversation can include a combination of text or audio inputs provided by the user and responses provided by the skill bots. These responses may be in the form of text or audio messages to the user and/or using simple user interface elements (e.g., select lists) that are presented to the user for the user to make selections.

[0064] There are various ways in which a skill or skill bot can be associated or added to a digital assistant. In some instances, a skill bot can be developed by an enterprise and then added to a digital assistant using DABP 102. In other instances, a skill bot can be developed and created using DABP 102 and then added to a digital assistant created using DABP 102. In yet other instances, DABP 102 provides an online digital store (referred to as a "skills store") that offers multiple skills directed to a wide range of tasks. The skills offered through the skills store may also expose various cloud services. In order to add a skill to a digital assistant being generated using DABP 102, a user of DABP 102 can access the skills store via DABP 102, select a desired skill, and indicate that the selected skill is to be added to the digital assistant created using DABP 102. A skill from the skills store can be added to a digital assistant as is or in a modified form (for example, a user of DABP 102 may select and clone a particular skill bot provided by the skills store, make customizations or modifications to the selected skill bot, and then add the modified skill bot to a digital assistant created using DABP 102).

[0065] Various different architectures may be used to implement a digital assistant or chatbot system. For example, in certain embodiments, the digital assistants created and deployed using DABP 102 may be implemented using a master bot/child(or sub) bot paradigm or architecture. According to this paradigm, a digital assistant is implemented as a master bot that interacts with one or more child bots that are skill bots. For example, in the embodiment depicted in FIG. 1, digital assistant 106 comprises a master bot 114 and skill

bots 116-1, 116-2, etc. that are child bots of master bot 114. In certain embodiments, digital assistant 106 is itself considered to act as the master bot.

[0066] A digital assistant implemented according to the master-child bot architecture enables users of the digital assistant to interact with multiple skills through a unified user interface, namely via the master bot. When a user engages with a digital assistant, the user input is received by the master bot. The master bot then performs processing to determine the meaning of the user input utterance. The master bot then determines whether the task requested by the user in the utterance can be handled by the master bot itself, else the master bot selects an appropriate skill bot for handling the user request and routes the conversation to the selected skill bot. This enables a user to converse with the digital assistant through a common single interface and still provide the capability to use several skill bots configured to perform specific tasks. For example, for a digital assistance developed for an enterprise, the master bot of the digital assistant may interface with skill bots with specific functionalities, such as a CRM bot for performing functions related to customer relationship management (CRM), an ERP bot for performing functions related to enterprise resource planning (ERP), an HCM bot for performing functions related to human capital management (HCM), etc. This way the end user or consumer of the digital assistant need only know how to access the digital assistant through the common master bot interface and behind the scenes multiple skill bots are provided for handling the user request.

[0067] In certain embodiments, in a master bot/child bots infrastructure, the master bot is configured to be aware of the available list of skill bots. The master bot may have access to metadata that identifies the various available skill bots, and for each skill bot, the capabilities of the skill bot including the tasks that can be performed by the skill bot. Upon receiving a user request in the form of an utterance, the master bot is configured to, from the multiple available skill bots, identify or predict a specific skill bot that can best serve or handle the user request. The master bot then routes the utterance (or a portion of the utterance) to that specific skill bot for further handling. Control thus flows from the master bot to the skill bots. The master bot can support multiple input and output channels. In certain embodiments, routing may be performed with the aid of processing performed by one or more available skill bots. For example, as discussed below, a skill bot can be trained to infer an intent for an utterance and to determine whether the inferred intent matches an intent with which the skill bot is configured. Thus, the routing performed by the master bot can involve

the skill bot communicating to the master bot an indication of whether the skill bot has been configured with an intent suitable for handling the utterance.

[0068] While the embodiment in FIG. 1 shows digital assistant 106 comprising a master bot 114 and skill bots 116-1, 116-2, and 116-3, this is not intended to be limiting. A digital assistant can include various other components (e.g., other systems and subsystems) that provide the functionalities of the digital assistant. These systems and subsystems may be implemented only in software (e.g., code, instructions stored on a computer-readable medium and executable by one or more processors), in hardware only, or in implementations that use a combination of software and hardware.

[0069] DABP 102 provides an infrastructure and various services and features that enable a user of DABP 102 to create a digital assistant including one or more skill bots associated with the digital assistant. In some instances, a skill bot can be created by cloning an existing skill bot, for example, cloning a skill bot provided by the skills store. As previously indicated, DABP 102 provides a skills store or skills catalog that offers multiple skill bots for performing various tasks. A user of DABP 102 can clone a skill bot from the skills store. As needed, modifications or customizations may be made to the cloned skill bot. In some other instances, a user of DABP 102 created a skill bot from scratch using tools and services offered by DABP 102. As previously indicated, the skills store or skills catalog provided by DABP 102 may offer multiple skill bots for performing various tasks.

[0070] In certain embodiments, at a high level, creating or customizing a skill bot involves the following steps:

- (1) Configuring settings for a new skill bot
- (2) Configuring one or more intents for the skill bot
- (3) Configuring one or more entities for one or more intents
- (4) Training the skill bot
- (5) Creating a dialog flow for the skill bot
- (6) Adding custom components to the skill bot as needed
- (7) Testing and deploying the skill bot

Each of the above steps is briefly described below.

[0071] (1) Configuring settings for a new skill bot – Various settings may be configured for the skill bot. For example, a skill bot designer can specify one or more invocation names for the skill bot being created. These invocation names can then be used by users of a digital assistant to explicitly invoke the skill bot. For example, a user can input an invocation name in the user's utterance to explicitly invoke the corresponding skill bot.

[0072] (2) Configuring one or more intents and associated example utterances for the skill bot– The skill bot designer specifies one or more intents (also referred to as bot intents) for a skill bot being created. The skill bot is then trained based upon these specified intents. These intents represent categories or classes that the skill bot is trained to infer for input utterances. Upon receiving an utterance, a trained skill bot infers an intent for the utterance, where the inferred intent is selected from the predefined set of intents used to train the skill bot. The skill bot then takes an appropriate action responsive to an utterance based upon the intent inferred for that utterance. In some instances, the intents for a skill bot represent tasks that the skill bot can perform for users of the digital assistant. Each intent is given an intent identifier or intent name. For example, for a skill bot trained for a bank, the intents specified for the skill bot may include "CheckBalance," "TransferMoney," "DepositCheck," and the like.

[0073] For each intent defined for a skill bot, the skill bot designer may also provide one or more example utterances that are representative of and illustrate the intent. These example utterances are meant to represent utterances that a user may input to the skill bot for that intent. For example, for the CheckBalance intent, example utterances may include "What's my savings account balance?", "How much is in my checking account?", "How much money do I have in my account," and the like. Accordingly, various permutations of typical user utterances may be specified as example utterances for an intent.

[0074] The intents and the their associated example utterances are used as training data to train the skill bot. Various different training techniques may be used. As a result of this training, a predictive model is generated that is configured to take an utterance as input and output an intent inferred for the utterance by the predictive model. In some instances, input utterances are provided to an intent analysis engine, which is configured to use the trained model to predict or infer an intent for the input utterance. The skill bot may then take one or more actions based upon the inferred intent.

[0075] (3) Configuring entities for one or more intents of the skill bot – In some instances, additional context may be needed to enable the skill bot to properly respond to a user

utterance. For example, there may be situations where a user input utterance resolves to the same intent in a skill bot. For instance, in the above example, utterances "What's my savings account balance?" and "How much is in my checking account?" both resolve to the same CheckBalance intent, but these utterances are different requests asking for different things.

5 To clarify such requests, one or more entities are added to an intent. Using the banking skill bot example, an entity called AccountType, which defines values called "checking" and "saving" may enable the skill bot to parse the user request and respond appropriately. In the above example, while the utterances resolve to the same intent, the value associated with the AccountType entity is different for the two utterances. This enables the skill bot to perform possibly different actions for the two utterances in spite of them resolving to the same intent. 10 One or more entities can be specified for certain intents configured for the skill bot. Entities are thus used to add context to the intent itself. Entities help describe an intent more fully and enable the skill bot to complete a user request.

[0076] In certain embodiments, there are two types of entities: (a) built-in entities provided 15 by DABP 102, and (2) custom entities that can be specified by a skill bot designer. Built-in entities are generic entities that can be used with a wide variety of bots. Examples of built-in entities include, without limitation, entities related to time, date, addresses, numbers, email addresses, duration, recurring time periods, currencies, phone numbers, URLs, and the like. Custom entities are used for more customized applications. For example, for a banking skill, 20 an AccountType entity may be defined by the skill bot designer that enables various banking transactions by checking the user input for keywords like checking, savings, and credit cards, etc.

[0077] (4) Training the skill bot – A skill bot is configured to receive user input in the form of utterances parse or otherwise process the received input, and identify or select an intent 25 that is relevant to the received user input. As indicated above, the skill bot has to be trained for this. In certain embodiments, a skill bot is trained based upon the intents configured for the skill bot and the example utterances associated with the intents (collectively, the training data), so that the skill bot can resolve user input utterances to one of its configured intents. In certain embodiments, the skill bot uses a predictive model that is trained using the training 30 data and allows the skill bot to discern what users say (or in some cases, are trying to say). DABP 102 provides various different training techniques that can be used by a skill bot designer to train a skill bot, including various machine-learning based training techniques, rules-based training techniques, and/or combinations thereof. In certain embodiments, a

portion (e.g., 80%) of the training data is used to train a skill bot model and another portion (e.g., the remaining 20%) is used to test or verify the model. Once trained, the trained model (also sometimes referred to as the trained skill bot) can then be used to handle and respond to user utterances. In certain cases, a user's utterance may be a question that requires only a
5 single answer and no further conversation. In order to handle such situations, a Q&A (question-and-answer) intent may be defined for a skill bot. This enables a skill bot to output replies to user requests without having to update the dialog definition. Q&A intents are created in a similar manner as regular intents. The dialog flow for Q&A intents can be different from that for regular intents.

10 **[0078]** (5) Creating a dialog flow for the skill bot -- A dialog flow specified for a skill bot describes how the skill bot reacts as different intents for the skill bot are resolved responsive to received user input. The dialog flow defines operations or actions that a skill bot will take, e.g., how the skill bot responds to user utterances, how the skill bot prompts users for input, how the skill bot returns data. A dialog flow is like a flowchart that is followed by the skill
15 bot. The skill bot designer specifies a dialog flow using a language, such as markdown language. In certain embodiments, a version of YAML called OBotML may be used to specify a dialog flow for a skill bot. The dialog flow definition for a skill bot acts as a model for the conversation itself, one that lets the skill bot designer choreograph the interactions between a skill bot and the users that the skill bot services.

20 **[0079]** In certain embodiments, the dialog flow definition for a skill bot contains three sections:

- (a) a context section
- (b) a default transitions section
- (c) a states section

25 **[0080]** Context section -- The skill bot designer can define variables that are used in a conversation flow in the context section. Other variables that may be named in the context section include, without limitation: variables for error handling, variables for built-in or custom entities, user variables that enable the skill bot to recognize and persist user preferences, and the like.

30 **[0081]** Default transitions section -- Transitions for a skill bot can be defined in the dialog flow states section or in the default transitions section. The transitions defined in the default

transition section act as a fallback and get triggered when there are no applicable transitions defined within a state, or the conditions required to trigger a state transition cannot be met. The default transitions section can be used to define routing that allows the skill bot to gracefully handle unexpected user actions.

- 5 **[0082]** States section – A dialog flow and its related operations are defined as a sequence of transitory states, which manage the logic within the dialog flow. Each state node within a dialog flow definition names a component that provides the functionality needed at that point in the dialog. States are thus built around the components. A state contains component-specific properties and defines the transitions to other states that get triggered after the
- 10 component executes.

[0083] Special case scenarios may be handled using the states sections. For example, there might be times when you want to provide users the option to temporarily leave a first skill they are engaged with to do something in a second skill within the digital assistant. For example, if a user is engaged in a conversation with a shopping skill (e.g., the user has made

15 some selections for purchase), the user may want to jump to a banking skill (e.g., the user may want to ensure that he/she has enough money for the purchase), and then return to the shopping skill to complete the user's order. To address this, an action in the first skill can be configured to initiate an interaction with the second different skill in the same digital assistant and then return to the original flow.

- 20 **[0084]** (6) Adding custom components to the skill bot – As described above, states specified in a dialog flow for a skill bot name components that provide the functionality needed corresponding to the states. Components enable a skill bot to perform functions. In certain embodiments, DABP 102 provides a set of preconfigured components for performing a wide range of functions. A skill bot designer can select one of more of these preconfigured
- 25 components and associate them with states in the dialog flow for a skill bot. The skill bot designer can also create custom or new components using tools provided by DABP 102 and associate the custom components with one or more states in the dialog flow for a skill bot.

[0085] (7) Testing and deploying the skill bot – DABP 102 provides several features that enable the skill bot designer to test a skill bot being developed. The skill bot can then be

30 deployed and included in a digital assistant.

[0086] While the description above describes how to create a skill bot, similar techniques may also be used to create a digital assistant (or the master bot). At the master bot or digital assistant level, built-in system intents may be configured for the digital assistant. These built-in system intents are used to identify general tasks that the digital assistant itself (i.e., the master bot) can handle without invoking a skill bot associated with the digital assistant.

Examples of system intents defined for a master bot include: (1) Exit: applies when the user signals the desire to exit the current conversation or context in the digital assistant; (2) Help: applies when the user asks for help or orientation; and (3) UnresolvedIntent: applies to user input that doesn't match well with the exit and help intents. The digital assistant also stores information about the one or more skill bots associated with the digital assistant. This information enables the master bot to select a particular skill bot for handling an utterance.

[0087] At the master bot or digital assistant level, when a user inputs a phrase or utterance to the digital assistant, the digital assistant is configured to perform processing to determine how to route the utterance and the related conversation. The digital assistant determines this using a routing model, which can be rules-based, AI-based, or a combination thereof. The digital assistant uses the routing model to determine whether the conversation corresponding to the user input utterance is to be routed to a particular skill for handling, is to be handled by the digital assistant or master bot itself per a built-in system intent, or is to be handled as a different state in a current conversation flow.

[0088] In certain embodiments, as part of this processing, the digital assistant determines if the user input utterance explicitly identifies a skill bot using its invocation name. If an invocation name is present in the user input, then it is treated as explicit invocation of the skill bot corresponding to the invocation name. In such a scenario, the digital assistant may route the user input to the explicitly invoked skill bot for further handling. If there is no specific or explicit invocation, in certain embodiments, the digital assistant evaluates the received user input utterance and computes confidence scores for the system intents and the skill bots associated with the digital assistant. The score computed for a skill bot or system intent represents how likely the user input is representative of a task that the skill bot is configured to perform or is representative of a system intent. Any system intent or skill bot with an associated computed confidence score exceeding a threshold value (e.g., a Confidence Threshold routing parameter) is selected as a candidate for further evaluation. The digital assistant then selects, from the identified candidates, a particular system intent or a skill bot for further handling of the user input utterance. In certain embodiments, after one

or more skill bots are identified as candidates, the intents associated with those candidate skills are evaluated (according to the intent model for each skill) and confidence scores are determined for each intent. In general, any intent that has a confidence score exceeding a threshold value (e.g., 70%) is treated as a candidate intent. If a particular skill bot is selected, then the user utterance is routed to that skill bot for further processing. If a system intent is selected, then one or more actions are performed by the master bot itself according to the selected system intent.

[0089] FIG. 2 is a simplified block diagram of a master bot (MB) system 200 according to certain embodiments. MB system 200 can be implemented in software only, hardware only, or a combination of hardware and software. MB system 200 includes a pre-processing subsystem 210, a multiple intent subsystem (MIS) 220, an explicit invocation subsystem (EIS) 230, a skill bot invoker 240, and a data store 250. MB system 200 depicted in FIG. 2 is merely an example of an arrangement of components in a master bot. One of ordinary skill in the art would recognize many possible variations, alternatives, and modifications. For example, in some implementations, MB system 200 may have more or fewer systems or components than those shown in FIG. 2, may combine two or more subsystems, or may have a different configuration or arrangement of subsystems.

[0090] Pre-processing subsystem 210 receives an utterance “A” 202 from a user and processes the utterance through a language detector 212 and a language parser 214. As indicated above, an utterance can be provided in various ways including audio or text. The utterance 202 can be a sentence fragment, a complete sentence, multiple sentences, and the like. Utterance 202 can include punctuation. For example, if the utterance 202 is provided as audio, the pre-processing subsystem 210 may convert the audio to text using a speech-to-text converter (not shown) that inserts punctuation marks into the resulting text, e.g., commas, semicolons, periods, etc.

[0091] Language detector 212 detects the language of the utterance 202 based on the text of the utterance 202. The manner in which the utterance 202 is handled depends on the language since each language has its own grammar and semantics. Differences between languages are taken into consideration when analyzing the syntax and structure of an utterance.

[0092] Language parser 214 parses the utterance 202 to extract part of speech (POS) tags for individual linguistic units (e.g., words) in the utterance 202. POS tags include, for example, noun (NN), pronoun (PN), verb (VB), and the like. Language parser 214 may also

tokenize the linguistic units of the utterance 202 (e.g., to convert each word into a separate token) and lemmatize words. A lemma is the main form of a set of words as represented in a dictionary (e.g., “run” is the lemma for run, runs, ran, running, etc.). Other types of pre-processing that the language parser 214 can perform include chunking of compound

5 expressions, e.g., combining “credit” and “card” into a single expression “credit_card.”

Language parser 214 may also identify relationships between the words in the utterance 202.

For example, in some embodiments, the language parser 214 generates a dependency tree that indicates which part of the utterance (e.g. a particular noun) is a direct object, which part of the utterance is a preposition, and so on. The results of the processing performed by the

10 language parser 214 form extracted information 205 and are provided as input to MIS 220 together with the utterance 202 itself.

[0093] As indicated above, the utterance 202 can include more than one sentence. For purposes of detecting multiple intents and explicit invocation, the utterance 202 can be treated as a single unit even if it includes multiple sentences. However, in certain

15 embodiments, pre-processing can be performed, e.g., by the pre-processing subsystem 210, to identify a single sentence among multiple sentences for multiple intents analysis and explicit invocation analysis. In general, the results produced by MIS 220 and EIS 230 are substantially the same regardless of whether the utterance 202 is processed at the level of an individual sentence or as a single unit comprising multiple sentences.

20 **[0094]** MIS 220 determines whether the utterance 202 represents multiple intents.

Although MIS 220 can detect the presence of multiple intents in the utterance 202, the processing performed by MIS 220 does not involve determining whether the intents of the utterance 202 match to any intents that have been configured for a bot. Instead, processing to determine whether an intent of the utterance 202 matches a bot intent can be performed by an intent classifier 242 of the MB system 200 or by an intent classifier of a skill bot (e.g., as

25 shown in the embodiment of FIG. 3). The processing performed by MIS 220 assumes that there exists a bot (e.g., a particular skill bot or the master bot itself) that can handle the utterance 202. Therefore, the processing performed by MIS 220 does not require knowledge of what bots are in the chatbot system (e.g., the identities of skill bots registered with the

30 master bot) or knowledge of what intents have been configured for a particular bot.

[0095] To determine that the utterance 202 includes multiple intents, the MIS 220 applies one or more rules from a set of rules 252 in the data store 250. The rules applied to the

utterance 202 depend on the language of the utterance 202 and may include sentence patterns that indicate the presence of multiple intents. For example, a sentence pattern may include a coordinating conjunction that joins two parts (e.g., conjuncts) of a sentence, where both parts correspond to a separate intent. If the utterance 202 matches the sentence pattern, it can be
5 inferred that the utterance 202 represents multiple intents. It should be noted that an utterance with multiple intents does not necessarily have different intents (e.g., intents directed to different bots or to different intents within the same bot). Instead, the utterance could have separate instances of the same intent, e.g. “Place a pizza order using payment account X, then place a pizza order using payment account Y.”

10 **[0096]** As part of determining that the utterance 202 represents multiple intents, the MIS 220 also determines what portions of the utterance 202 are associated with each intent. MIS 220 constructs, for each intent represented in an utterance containing multiple intents, a new utterance for separate processing in place of the original utterance, e.g., an utterance “B” 206 and an utterance “C” 208, as depicted in FIG. 2. Thus, the original utterance 202 can be split
15 into two or more separate utterances that are handled one at a time. MIS 220 determines, using the extracted information 205 and/or from analysis of the utterance 202 itself, which of the two or more utterances should be handled first. For example, MIS 220 may determine that the utterance 202 contains a marker word indicating that a particular intent should be handled first. The newly formed utterance corresponding to this particular intent (e.g., one of
20 utterance 206 or utterance 208) will be the first to be sent for further processing by EIS 230. After a conversation triggered by the first utterance has ended (or has been temporarily suspended), the next highest priority utterance (e.g., the other one of utterance 206 or utterance 208) can then be sent to the EIS 230 for processing.

[0097] EIS 230 determines whether the utterance that it receives (e.g., utterance 206 or
25 utterance 208) contains an invocation name of a skill bot. In certain embodiments, each skill bot in a chatbot system is assigned a unique invocation name that distinguishes the skill bot from other skill bots in the chatbot system. A list of invocation names can be maintained as part of skill bot information 254 in data store 250. An utterance is deemed to be an explicit invocation when the utterance contains a word match to an invocation name. If a bot is not
30 explicitly invoked, then the utterance received by the EIS 230 is deemed a non-explicitly invoking utterance 234 and is input to an intent classifier (e.g., intent classifier 242) of the master bot to determine which bot to use for handling the utterance. In some instances, the intent classifier 242 will determine that the master bot should handle a non-explicitly

invoking utterance. In other instances, the intent classifier 242 will determine a skill bot to route the utterance to for handling.

5 [0098] The explicit invocation functionality provided by the EIS 230 has several advantages. It can reduce the amount of processing that the master bot has to perform. For example, when there is an explicit invocation, the master bot may not have to do any intent classification analysis (e.g., using the intent classifier 242), or may have to do reduced intent classification analysis for selecting a skill bot. Thus, explicit invocation analysis may enable selection of a particular skill bot without resorting to intent classification analysis.

10 [0099] Also, there may be situations where there is an overlap in functionalities between multiple skill bots. This may happen, for example, if the intents handled by the two skill bots overlap or are very close to each other. In such a situation, it may be difficult for the master bot to identify which of the multiple skill bots to select based upon intent classification analysis alone. In such scenarios, the explicit invocation disambiguates the particular skill bot to be used.

15 [0100] In addition to determining that an utterance is an explicit invocation, the EIS 230 is responsible for determining whether any portion of the utterance should be used as input to the skill bot being explicitly invoked. In particular, EIS 230 can determine whether part of the utterance is not associated with the invocation. The EIS 230 can perform this determination through analysis of the utterance and/or analysis of the extracted information 205. EIS 230
20 can send the part of the utterance not associated with the invocation to the invoked skill bot in lieu of sending the entire utterance that was received by the EIS 230. In some instances, the input to the invoked skill bot is formed simply by removing any portion of the utterance associated with the invocation. For example, “I want to order pizza using Pizza Bot” can be shortened to “I want to order pizza” since “using Pizza Bot” is relevant to the invocation of
25 the pizza bot, but irrelevant to any processing to be performed by the pizza bot. In some instances, EIS 230 may reformat the part to be sent to the invoked bot, e.g., to form a complete sentence. Thus, the EIS 230 determines not only that there is an explicit invocation, but also what to send to the skill bot when there is an explicit invocation. In some instances, there may not be any text to input to the bot being invoked. For example, if the utterance was
30 “Pizza Bot”, then the EIS 230 could determine that the pizza bot is being invoked, but there is no text to be processed by the pizza bot. In such scenarios, the EIS 230 may indicate to the skill bot invoker 240 that there is nothing to send.

[0101] Skill bot invoker 240 invokes a skill bot in various ways. For instance, skill bot invoker 240 can invoke a bot in response to receiving an indication 235 that a particular skill bot has been selected as a result of an explicit invocation. The indication 235 can be sent by the EIS 230 together with the input for the explicitly invoked skill bot. In this scenario, the skill bot invoker 240 will turn control of the conversation over to the explicitly invoked skill bot. The explicitly invoked skill bot will determine an appropriate response to the input from the EIS 230 by treating the input as a stand-alone utterance. For example, the response could be to perform a specific action or to start a new conversation in a particular state, where the initial state of the new conversation depends on the input sent from the EIS 230.

[0102] Another way in which skill bot invoker 240 can invoke a skill bot is through implicit invocation using the intent classifier 242. The intent classifier 242 can be trained, using machine-learning and/or rules-based training techniques, to determine a likelihood that an utterance is representative of a task that a particular skill bot is configured to perform. The intent classifier 242 is trained on different classes, one class for each skill bot. For instance, whenever a new skill bot is registered with the master bot, a list of example utterances associated with the new skill bot can be used to train the intent classifier 242 to determine a likelihood that a particular utterance is representative of a task that the new skill bot can perform. The parameters produced as result of this training (e.g., a set of values for parameters of a machine-learning model) can be stored as part of skill bot information 254.

[0103] In certain embodiments, the intent classifier 242 is implemented using a machine-learning model, as described in further detail herein. Training of the machine-learning model may involve inputting at least a subset of utterances from the example utterances associated with various skill bots to generate, as an output of the machine-learning model, inferences as to which bot is the correct bot for handling any particular training utterance. For each training utterance, an indication of the correct bot to use for the training utterance may be provided as ground truth information. The behavior of the machine-learning model can then be adapted (e.g., through back-propagation) to minimize the difference between the generated inferences and the ground truth information.

[0104] In certain embodiments, the intent classifier 242 determines, for each skill bot registered with the master bot, a confidence score indicating a likelihood that the skill bot can handle an utterance (e.g., the non-explicitly invoking utterance 234 received from EIS 230). The intent classifier 242 may also determine a confidence score for each system level intent

(e.g., help, exit) that has been configured. If a particular confidence score meets one or more conditions, then the skill bot invoker 240 will invoke the bot associated with the particular confidence score. For example, a threshold confidence score value may need to be met. Thus, an output 245 of the intent classifier 242 is either an identification of a system intent or an identification of a particular skill bot. In some embodiments, in addition to meeting a threshold confidence score value, the confidence score must exceed the next highest confidence score by a certain win margin. Imposing such a condition would enable routing to a particular skill bot when the confidence scores of multiple skill bots each exceed the threshold confidence score value.

10 [0105] After identifying a bot based on evaluation of confidence scores, the skill bot invoker 240 hands over processing to the identified bot. In the case of a system intent, the identified bot is the master bot. Otherwise, the identified bot is a skill bot. Further, the skill bot invoker 240 will determine what to provide as input 247 for the identified bot. As indicated above, in the case of an explicit invocation, the input 247 can be based on a part of
15 an utterance that is not associated with the invocation, or the input 247 can be nothing (e.g., an empty string). In the case of an implicit invocation, the input 247 can be the entire utterance.

[0106] Data store 250 comprises one or more computing devices that store data used by the various subsystems of the master bot system 200. As explained above, the data store 250
20 includes rules 252 and skill bot information 254. The rules 252 include, for example, rules for determining, by MIS 220, when an utterance represents multiple intents and how to split an utterance that represents multiple intents. The rules 252 further include rules for determining, by EIS 230, which parts of an utterance that explicitly invokes a skill bot to send to the skill bot. The skill bot information 254 includes invocation names of skill bots in the chatbot
25 system, e.g., a list of the invocation names of all skill bots registered with a particular master bot. The skill bot information 254 can also include information used by intent classifier 242 to determine a confidence score for each skill bot in the chatbot system, e.g., parameters of a machine-learning model.

[0107] FIG. 3 is a simplified block diagram of a skill bot system 300 according to certain
30 embodiments. Skill bot system 300 is a computing system that can be implemented in software only, hardware only, or a combination of hardware and software. In certain

embodiments such as the embodiment depicted in FIG. 1, skill bot system 300 can be used to implement one or more skill bots within a digital assistant.

[0108] Skill bot system 300 includes an MIS 310, an intent classifier 320, and a conversation manager 330. The MIS 310 is analogous to the MIS 220 in FIG. 2 and provides similar functionality, including being operable to determine, using rules 352 in a data store 350: (1) whether an utterance represents multiple intents and, if so, (2) how to split the utterance into a separate utterance for each intent of the multiple intents. In certain embodiments, the rules applied by MIS 310 for detecting multiple intents and for splitting an utterance are the same as those applied by MIS 220. The MIS 310 receives an utterance 302 and extracted information 304. The extracted information 304 is analogous to the extracted information 205 in FIG. 1 and can be generated using the language parser 214 or a language parser local to the skill bot system 300.

[0109] Intent classifier 320 can be trained in a similar manner to the intent classifier 242 discussed above in connection with the embodiment of FIG. 2 and as described in further detail herein. For instance, in certain embodiments, the intent classifier 320 is implemented using a machine-learning model. The machine-learning model of the intent classifier 320 is trained for a particular skill bot, using at least a subset of example utterances associated with that particular skill bot as training utterances. The ground truth for each training utterance would be the particular bot intent associated with the training utterance.

[0110] The utterance 302 can be received directly from the user or supplied through a master bot. When the utterance 302 is supplied through a master bot, e.g., as a result of processing through MIS 220 and EIS 230 in the embodiment depicted in FIG. 2, the MIS 310 can be bypassed so as to avoid repeating processing already performed by MIS 220. However, if the utterance 302 is received directly from the user, e.g., during a conversation that occurs after routing to a skill bot, then MIS 310 can process the utterance 302 to determine whether the utterance 302 represents multiple intents. If so, then MIS 310 applies one or more rules to split the utterance 302 into a separate utterance for each intent, e.g., an utterance “D” 306 and an utterance “E” 308. If utterance 302 does not represent multiple intents, then MIS 310 forwards the utterance 302 to intent classifier 320 for intent classification and without splitting the utterance 302.

[0111] Intent classifier 320 is configured to match a received utterance (e.g., utterance 306 or 308) to an intent associated with skill bot system 300. As explained above, a skill bot can

be configured with one or more intents, each intent including at least one example utterance that is associated with the intent and used for training a classifier. In the embodiment of FIG. 2, the intent classifier 242 of the master bot system 200 is trained to determine confidence scores for individual skill bots and confidence scores for system intents. Similarly, intent classifier 320 can be trained to determine a confidence score for each intent associated with the skill bot system 300. Whereas the classification performed by intent classifier 242 is at the bot level, the classification performed by intent classifier 320 is at the intent level and therefore finer grained. The intent classifier 320 has access to intents information 354. The intents information 354 includes, for each intent associated with the skill bot system 300, a list of utterances that are representative of and illustrate the meaning of the intent and are typically associated with a task performable by that intent. The intents information 354 can further include parameters produced as a result of training on this list of utterances.

[0112] Conversation manager 330 receives, as an output of intent classifier 320, an indication 322 of a particular intent, identified by the intent classifier 320, as best matching the utterance that was input to the intent classifier 320. In some instances, the intent classifier 320 is unable to determine any match. For example, the confidence scores computed by the intent classifier 320 could fall below a threshold confidence score value if the utterance is directed to a system intent or an intent of a different skill bot. When this occurs, the skill bot system 300 may refer the utterance to the master bot for handling, e.g., to route to a different skill bot. However, if the intent classifier 320 is successful in identifying an intent within the skill bot, then the conversation manager 330 will initiate a conversation with the user.

[0113] The conversation initiated by the conversation manager 330 is a conversation specific to the intent identified by the intent classifier 320. For instance, the conversation manager 330 may be implemented using a state machine configured to execute a dialog flow for the identified intent. The state machine can include a default starting state (e.g., for when the intent is invoked without any additional input) and one or more additional states, where each state has associated with it actions to be performed by the skill bot (e.g., executing a purchase transaction) and/or dialog (e.g., questions, responses) to be presented to the user. Thus, the conversation manager 330 can determine an action/dialog 335 upon receiving the indication 322 identifying the intent, and can determine additional actions or dialog in response to subsequent utterances received during the conversation.

[0114] Data store 350 comprises one or more computing devices that store data used by the various subsystems of the skill bot system 300. As depicted in FIG. 3, the data store 350 includes the rules 352 and the intents information 354. In certain embodiments, data store 350 can be integrated into a data store of a master bot or digital assistant, e.g., the data store 250 in FIG. 2.

Fine-Tuning

[0115] It has been discovered that fine-tuning a machine-learning model (e.g., an NER) for a new context by training an incomplete subset of layers implementing a multi-head attention technique can drastically improve the efficiency of training (e.g., in terms of training time, usage of computational resources for training, time to collect a data set for use in fine-tuning, and/or usage of computational resources for use in fine-tuning) as compared to the efficiency of a comparable training where all parameters were configurable during the fine-tuning. Further, the accuracy of the fine-tuned model can be can nonetheless positively compare to (e.g., may match, may be at least 90% of, may be at least 80% of, may be at least 70% of, may be at least 60% of, or may be at least 50% of) the comparable model that trained all parameters based on client data corresponding to a new context. The machine-learning model that is to be fine-tuned and/or the machine-learning model that is fine-tuned may be implemented in a chat system, as described with respect to FIGS. 1, 2 and 3.

[0116] FIG. 4 shows a block diagram illustrating aspects of a chatbot system 400 configured to train one or more models, e.g., implemented as a named entity recognizer, based on text data. As shown in FIG. 4, the entity recognition performed by the chatbot system 400 in this example includes various stages: a pre-training stage 402 to build and train a prediction model (such that each parameter within the prediction model is learned), a fine-tuning stage 404 to fine-tune the prediction model (such that each of at least some of the parameters of the prediction model are modified) and a chatbot implementation stage 406 for implementing one or more chatbots. Thus, at pre-training stage 402, a pre-trained machine-learning model 408 is generated; at fine-tuning stage 404, the pre-trained machine-learning model 408 is fine-tuned to generate a fine-tuned machine-learning model 410; and at chatbot implementation stage 406, the fine-tuned machine-learning model 410 is used to support one or more chatbots processing received text data 412.

[0117] Each of the pre-trained machine-learning model 408 and the fine-tuned machine-learning model 410 can be a named entity recognizer model configured and used to

(for example) recognize one or more entities in an utterance. At the chatbot implementation stage, the fine-tuned machine-learning model 410 may be used in combination with one or more other models, such as another model for determining a likelihood that an utterance is representative of a task that a particular skill bot is configured to perform, another model for predicting an intent from an utterance for a first type of skill bot, and/or another model for predicting an intent from an utterance for a second type of skill bot.

[0118] Each of the pre-trained machine-learning model 408 and the fine-tuned machine learning model 410 can be and/or can include: a transformer, model using a multi-head attention technique, convolutional neural network (“CNN”), *e.g.* an inception neural network, a residual neural network (“Resnet”), a recurrent neural network, *e.g.*, long short-term memory (“LSTM”) models or gated recurrent units (“GRUs”) models, and/or other variants of Deep Neural Networks (“DNN”) (*e.g.*, a multi-label n-binary DNN classifier or multi-class DNN classifier for single intent classification).

[0119] Each of the pre-trained machine-learning model 408 and the fine-tuned machine learning model 410 can be or can include any other suitable machine-learning model trained for natural language processing, such as a Naive Bayes Classifier, Linear Classifier, Support Vector Machine, Bagging Models such as Random Forest Model, Boosting Models, Shallow Neural Networks, or combinations of one or more of such techniques—*e.g.*, CNN-HMM or MCNN (Multi-Scale Convolutional Neural Network).

[0120] Each of the pre-trained machine-learning model 408 and the fine-tuned machine-learning model 410 can include multiple layers, multiple layers that use a multi-head attention technique, multiple layers that include a transformer model, multiple layers that use a multi-head self-attention technique. For a multi-head self-attention technique, each of the multiple heads performs a self-attention technique using different Query, Key, and Value linear projections.

[0121] Each of the pre-trained machine-learning model 408 and the fine-tuned machine-learning model 410 can include multiple layers that use a self-attention technique. A self-attention technique can transform a given input into each of a Query representation (generated by using a Query linear projection), a Key representation (generated by using a Key linear projection), and a Value representation (generated by using a Value linear projection). Multiplying the Query representation by the Key representation (and potentially applying an activation function, such as a softmax function) to generate attention scores that

indicate how much attention is to be paid to a value at a given position when evaluating a value at another position. The attention scores can then be multiplied by the Value representation to generate an output.

5 [0122] In some instances, an architecture and/or hyperparameters of pre-trained machine-learning model 408 are the same as an architecture and/or hyperparameters of fine-tuned machine-learning model 410.

10 [0123] The chatbot system 400 may employ the same type of prediction model or different types of prediction models for recognizing one or more entities in an utterance, determining a likelihood that an utterance is representative of a task that a particular skill bot is configured to perform, predicting an intent from an utterance for a first type of skill bot, and predicting an intent from an utterance for a second type of skill bot. Still other types of prediction models may be implemented in other examples according to this disclosure.

15 [0124] At the pre-training stage 402, pre-training data assets 414 are loaded, split into training and validation sets so that the system can train and test the model being built. The splitting the pre-training data assets 414 into training and validation sets may be performed randomly (e.g., a 90/10% or 70/30%) or the splitting may be performed in accordance with a more complex validation technique such as K-Fold Cross-Validation, Leave-one-out Cross-Validation, Leave-one-group-out Cross-Validation, Nested Cross-Validation, or the like to minimize sampling bias and overfitting.

20 [0125] The pre-training data assets 414 may include at least a subset of utterances from example utterances associated with one or more skill bots. As indicated above, an utterance can be provided in various ways including audio or text. The utterance can be a sentence fragment, a complete sentence, multiple sentences, and the like. In some instances, the example utterances are provided by a previous or existing client or customer. In other
25 instances, the example utterances are automatically generated from prior libraries of utterances (e.g., identifying utterances from a library that are specific to a skill that a chatbot is designated to learn). The pre-training data assets 414 can include input text or audio (or input features of text or audio frames).

30 [0126] At the pre-training stages, pre-training labels 416 may also be obtained, where each of the pre-training label corresponds to an individual pre-training data asset. The pre-training labels may be stored in or stored as a matrix or table of values. For each training utterance, an

indication of the correct entities and classification thereof to be inferred by the may be provided as ground truth information for pre-training labels 416. The behavior of the model being pre-trained can then be adapted (e.g., through back-propagation) to minimize the difference between the generated inferences for various entities and the ground truth information.

[0127] The pre-training can include selecting hyperparameters for the pre-trained machine learning model 408 and performing iterative operations of inputting utterances from at least some of the pre-training data assets 414 into pre-trained machine learning model 408 to find a set of model parameters (e.g., weights and/or biases) that maximizes or minimizes an objective function, e.g., minimizes a loss function, for pre-trained machine learning model 408. The hyperparameters are settings that can be tuned or optimized to control the behavior of the pre-trained machine learning model 408. Most models explicitly define hyperparameters that control different aspects of the models such as memory or cost of execution. However, additional hyperparameters may be defined to adapt a model to a specific scenario. For example, the hyperparameters may include the number of hidden units or layers of a model, the learning rate of a model, the convolution kernel width, or the number of parameters for a model. Each iteration of training can involve finding a set of model parameters for the model 408 (configured with a defined set of hyperparameters) so that the value of the objective function using the set of model parameters is smaller than the value of the objective function using a different set of model parameters in a previous iteration. The objective function can be constructed to measure the difference between the outputs inferred using the model 408 and the ground truth annotated to the subset of augmented pre-training data assets 414 using the labels 416.

[0128] The pre-training can include processing each of one or more pre-training data assets using a current version of the model to generate a prediction, calculating a loss based on the prediction and a label corresponding to the data asset, and updating one or more parameters of the model based on the loss. In order to generate the prediction, multiple intermediate values may be generated for each pre-training data asset. For example, an intermediate value may include an output from a layer in the model. Each layer may generate multiple outputs (e.g., over 100, over 1,000, over 10,000, or over 100,000 outputs). In some instances, the model in its entirety is configured to generate a consistent output so long as an input is the same, the model's hyperparameters are the same, and the model's parameters are the same.

[0129] Fine-tuning the model may include fine-tuning some of the model's total parameters, while leaving other parameters of the model fixed to the values identified during training. For example, fine-tuning may include adjusting parameters associated with one or more particular layers, while keeping parameter values for one or more lower layers fixed. In this case, so long as the model is not using different hyperparameters (e.g., random-number seeds) across learning iterations, the outputs from the one or more lower layers should be the same for a given input data asset across learning iterations. Accordingly, once the model is pre-trained, one or more intermediate values corresponding to outputs by a layer below each layer that is to be fine-tuned can be stored in a cache data store 418 for each pre-training data assets (in association with an identifier of the data asset and/or corresponding label). In some instances, an architecture of at least one layer that is to be fine-tuned is the same as an architecture of at least one layer that is to be fixed.

[0130] Caching intermediate values has an advantage of facilitate fine-tuning of the model for each of multiple environments or of multiple contexts. For example, the model may be pre-trained to learn elements of a given language (e.g., English) and may be fine-tuned multiple times to learn how to respond to text or audio received in association with a given web site or a given merchant.

[0131] At fine-tuning stage 404, the cached intermediate values and fine-tuning labels 420 are used to fine-tune a subset of the parameters initially learned during pre-training stage 402. For example, a pre-training data asset may include the text: "I have not received my shipment". An output by a pre-trained model may correspond to a request for an update on a shipment status. An output from a lower layer in the model may represent the words "product" "not received". An output by a fine-tuned model may correspond to a request for a shipment to be completed as soon as possible.

[0132] In some instances, the fine-tuning can be performed without using underlying data assets 414, given that the cached data may be usable in lieu of the input data sets. Fine-tuning the model can include learning parameters for one or more layers using a self-attention technique and/or a multi-head attention technique.

[0133] A fine-tuned model 420 can be defined to be configured with at least some parameters learned during pre-training stage 402 and at least some parameters learned during fine-tuning stage 404.

[0134] Fine-tuned model 420 can be or can be part of an entity recognition model. Fine-tuned model 420 can be used in implementation stage 406 to process new text data 424,

[0135] At implementation stage 406, fine-tuned model 420 may be deployed and used as part of or as a trained entity recognition model implementing one or more chatbots. For example, one or more chatbots may be configured with fine-tuned model 420 to receive text data 424 from one or more users, and recognize and extract entities from various utterances received by the one or more chatbots. Text data 424 may include text data received in an environment or context corresponding to that for which fine-tune machine learning model 422 was fine-tuned. The entities may be part of the extracted information (e.g., extracted information 205; 304 described respectively in FIGS 2 and 3) obtained from the text data 460, and may be used in downstream processing such as intent classification. An output generated based on the extracted entities may then be transmitted to and/or presented at a device corresponding to a source of the utterance.

Techniques for Generating and Using a Fine-Tuned Model for a Given Environment

[0136] FIG. 5 is a flowchart illustrating a process 500 for generating and using a fine-tuned model for a given environment according to certain embodiments. The processing depicted in FIG. 5 may be implemented in software (e.g., code, instructions, program) executed by one or more processing units (e.g., processors, cores) of the respective systems, hardware, or combinations thereof. The software may be stored on a non-transitory storage medium (e.g., on a memory device). The method presented in FIG. 5 and described below is intended to be illustrative and non-limiting. Although FIG. 5 depicts the various processing steps occurring in a particular sequence or order, this is not intended to be limiting. In certain alternative embodiments, the steps may be performed in some different order or some steps may also be performed in parallel. In certain embodiments, such as in the embodiment depicted in FIGS. 1-4, the processing depicted in FIG. 5 may be performed by a pre-processing subsystem (e.g., pre-processing subsystem 210 or prediction model training stage 410) to generate keyword augmented data sets for training by one or more predictive models (e.g., the intent classifier 242 or 320 or prediction models 425).

[0137] At 505, a pre-trained version of a machine-learning model is accessed. The machine-learning model includes multiple layers configured to implement a self-attention technique (e.g., multiple layers configured to implement a multi-head attention technique).

[0138] At block 510, a client data set is accessed. The client data set may include a set of labels, where each label corresponds to an input data set used to pre-train the machine-learning model. Additionally or alternatively, the client data set includes pairs of input data sets and labels, where each of the input data sets and the labels are provided by, selected by, and/or identified by a client.

[0139] At block 515, an incomplete subset of layers in the machine-learning model is identified for fine-tuning the model using the client data set. The incomplete subset of layers may be identified based on stored information indicating which layer(s) are to be fine-tuned during a fine-tuning stage. In some instances, the incomplete subset of layers is identified based on a size of the client data set, a selection from a client device, a number of layers in the machine-learning model, and/or a preliminary comparison of a distribution corresponding to data and/or labels in client data set as compared to a distribution corresponding to data and/or labels used to pre-train the model. The incomplete subset of layers may include one or more layers that are above each other layer not in the subset. For example, the incomplete subset of layers may include layers 8 and above in the model, while layers 1-7 are not in the subset (where the model is configured such that output from layer 7 feeds to layer 8).

[0140] At block 520, the machine-learning model is fine-tuned using the client data set to generate an updated, fine-tuned version of the model. During the fine-tuning, parameter values associated with layers that are not in the incomplete subset of layers may be frozen (e.g., and retrieved from a cache for an initial model configuration), while the parameter values for the subset of layers can be learned.

[0141] At block 525, a use of the updated fine-tuned version of machine-learning model is facilitated. For example, the updated version of the machine-learning model can be used to transform an utterance into an intent representation and/or to identify a response for the utterance.

Illustrative Systems

[0142] FIG. 6 depicts a simplified diagram of a distributed system 600. In the illustrated example, distributed system 600 includes one or more client computing devices 602, 604, 606, and 608, coupled to a server 612 via one or more communication networks 610. Clients computing devices 602, 604, 606, and 608 may be configured to execute one or more applications.

[0143] In various examples, server 612 may be adapted to run one or more services or software applications that enable one or more embodiments described in this disclosure. In certain examples, server 612 may also provide other services or software applications that may include non-virtual and virtual environments. In some examples, these services may be offered as web-based or cloud services, such as under a Software as a Service (SaaS) model to the users of client computing devices 602, 604, 606, and/or 608. Users operating client computing devices 602, 604, 606, and/or 608 may in turn utilize one or more client applications to interact with server 612 to utilize the services provided by these components.

[0144] In the configuration depicted in FIG. 6, server 612 may include one or more components 618, 620 and 622 that implement the functions performed by server 612. These components may include software components that may be executed by one or more processors, hardware components, or combinations thereof. It should be appreciated that various different system configurations are possible, which may be different from distributed system 600. The example shown in FIG. 6 is thus one example of a distributed system for implementing an example system and is not intended to be limiting.

[0145] Users may use client computing devices 602, 604, 606, and/or 608 to execute one or more applications, models or chatbots, which may generate one or more events or models that may then be implemented or serviced in accordance with the teachings of this disclosure. A client device may provide an interface that enables a user of the client device to interact with the client device. The client device may also output information to the user via this interface. Although FIG. 6 depicts only four client computing devices, any number of client computing devices may be supported.

[0146] The client devices may include various types of computing systems such as portable handheld devices, general purpose computers such as personal computers and laptops, workstation computers, wearable devices, gaming systems, thin clients, various messaging devices, sensors or other sensing devices, and the like. These computing devices may run various types and versions of software applications and operating systems (e.g., Microsoft Windows®, Apple Macintosh®, UNIX® or UNIX-like operating systems, Linux or Linux-like operating systems such as Google Chrome™ OS) including various mobile operating systems (e.g., Microsoft Windows Mobile®, iOS®, Windows Phone®, Android™, BlackBerry®, Palm OS®). Portable handheld devices may include cellular phones, smartphones, (e.g., an iPhone®), tablets (e.g., iPad®), personal digital assistants (PDAs), and the like. Wearable

devices may include Google Glass® head mounted display, and other devices. Gaming systems may include various handheld gaming devices, Internet-enabled gaming devices (e.g., a Microsoft Xbox® gaming console with or without a Kinect® gesture input device, Sony PlayStation® system, various gaming systems provided by Nintendo®, and others), and the like. The client devices may be capable of executing various different applications such as various Internet-related apps, communication applications (e.g., E-mail applications, short message service (SMS) applications) and may use various communication protocols.

[0147] Network(s) 610 may be any type of network familiar to those skilled in the art that may support data communications using any of a variety of available protocols, including without limitation TCP/IP (transmission control protocol/Internet protocol), SNA (systems network architecture), IPX (Internet packet exchange), AppleTalk®, and the like. Merely by way of example, network(s) 610 may be a local area network (LAN), networks based on Ethernet, Token-Ring, a wide-area network (WAN), the Internet, a virtual network, a virtual private network (VPN), an intranet, an extranet, a public switched telephone network (PSTN), an infra-red network, a wireless network (e.g., a network operating under any of the Institute of Electrical and Electronics (IEEE) 1002.11 suite of protocols, Bluetooth®, and/or any other wireless protocol), and/or any combination of these and/or other networks.

[0148] Server 612 may be composed of one or more general purpose computers, specialized server computers (including, by way of example, PC (personal computer) servers, UNIX® servers, mid-range servers, mainframe computers, rack-mounted servers, etc.), server farms, server clusters, or any other appropriate arrangement and/or combination. Server 612 may include one or more virtual machines running virtual operating systems, or other computing architectures involving virtualization such as one or more flexible pools of logical storage devices that may be virtualized to maintain virtual storage devices for the server. In various examples, server 612 may be adapted to run one or more services or software applications that provide the functionality described in the foregoing disclosure.

[0149] The computing systems in server 612 may run one or more operating systems including any of those discussed above, as well as any commercially available server operating system. Server 612 may also run any of a variety of additional server applications and/or mid-tier applications, including HTTP (hypertext transport protocol) servers, FTP (file transfer protocol) servers, CGI (common gateway interface) servers, JAVA® servers, database servers, and the like. Exemplary database servers include without limitation those

commercially available from Oracle®, Microsoft®, Sybase®, IBM® (International Business Machines), and the like.

[0150] In some implementations, server 612 may include one or more applications to analyze and consolidate data feeds and/or event updates received from users of client computing devices 602, 604, 606, and 608. As an example, data feeds and/or event updates may include, but are not limited to, Twitter® feeds, Facebook® updates or real-time updates received from one or more third party information sources and continuous data streams, which may include real-time events related to sensor data applications, financial tickers, network performance measuring tools (e.g., network monitoring and traffic management applications), clickstream analysis tools, automobile traffic monitoring, and the like. Server 612 may also include one or more applications to display the data feeds and/or real-time events via one or more display devices of client computing devices 602, 604, 606, and 608.

[0151] Distributed system 600 may also include one or more data repositories 614, 616. These data repositories may be used to store data and other information in certain examples. For example, one or more of the data repositories 614, 616 may be used to store information such as information related to chatbot performance or generated models for use by chatbots used by server 612 when performing various functions in accordance with various embodiments. Data repositories 614, 616 may reside in a variety of locations. For example, a data repository used by server 612 may be local to server 612 or may be remote from server 612 and in communication with server 612 via a network-based or dedicated connection. Data repositories 614, 616 may be of different types. In certain examples, a data repository used by server 612 may be a database, for example, a relational database, such as databases provided by Oracle Corporation® and other vendors. One or more of these databases may be adapted to enable storage, update, and retrieval of data to and from the database in response to SQL-formatted commands.

[0152] In certain examples, one or more of data repositories 614, 616 may also be used by applications to store application data. The data repositories used by applications may be of different types such as, for example, a key-value store repository, an object store repository, or a general storage repository supported by a file system.

[0153] In certain examples, the functionalities described in this disclosure may be offered as services via a cloud environment. FIG. 7 is a simplified block diagram of a cloud-based system environment in which various services may be offered as cloud services in accordance

with certain examples. In the example depicted in FIG. 7, cloud infrastructure system 702 may provide one or more cloud services that may be requested by users using one or more client computing devices 704, 706, and 708. Cloud infrastructure system 702 may comprise one or more computers and/or servers that may include those described above for server 612.

- 5 The computers in cloud infrastructure system 702 may be organized as general purpose computers, specialized server computers, server farms, server clusters, or any other appropriate arrangement and/or combination.

[0154] Network(s) 710 may facilitate communication and exchange of data between clients 704, 706, and 708 and cloud infrastructure system 702. Network(s) 710 may include one or
10 more networks. The networks may be of the same or different types. Network(s) 710 may support one or more communication protocols, including wired and/or wireless protocols, for facilitating the communications.

[0155] The example depicted in FIG. 7 is only one example of a cloud infrastructure system and is not intended to be limiting. It should be appreciated that, in some other
15 examples, cloud infrastructure system 702 may have more or fewer components than those depicted in FIG. 7, may combine two or more components, or may have a different configuration or arrangement of components. For example, although FIG. 7 depicts three client computing devices, any number of client computing devices may be supported in alternative examples.

20 [0156] The term cloud service is generally used to refer to a service that is made available to users on demand and via a communication network such as the Internet by systems (e.g., cloud infrastructure system 702) of a service provider. Typically, in a public cloud environment, servers and systems that make up the cloud service provider's system are different from the customer's own on-premise servers and systems. The cloud service
25 provider's systems are managed by the cloud service provider. Customers may thus avail themselves of cloud services provided by a cloud service provider without having to purchase separate licenses, support, or hardware and software resources for the services. For example, a cloud service provider's system may host an application, and a user may, via the Internet, on demand, order and use the application without the user having to buy infrastructure resources
30 for executing the application. Cloud services are designed to provide easy, scalable access to applications, resources and services. Several providers offer cloud services. For example,

several cloud services are offered by Oracle Corporation® of Redwood Shores, California, such as middleware services, database services, Java cloud services, and others.

[0157] In certain examples, cloud infrastructure system 702 may provide one or more cloud services using different models such as under a Software as a Service (SaaS) model, a
5 Platform as a Service (PaaS) model, an Infrastructure as a Service (IaaS) model, and others, including hybrid service models. Cloud infrastructure system 702 may include a suite of applications, middleware, databases, and other resources that enable provision of the various cloud services.

[0158] A SaaS model enables an application or software to be delivered to a customer over
10 a communication network like the Internet, as a service, without the customer having to buy the hardware or software for the underlying application. For example, a SaaS model may be used to provide customers access to on-demand applications that are hosted by cloud infrastructure system 702. Examples of SaaS services provided by Oracle Corporation® include, without limitation, various services for human resources/capital management,
15 customer relationship management (CRM), enterprise resource planning (ERP), supply chain management (SCM), enterprise performance management (EPM), analytics services, social applications, and others.

[0159] An IaaS model is generally used to provide infrastructure resources (e.g., servers, storage, hardware and networking resources) to a customer as a cloud service to provide
20 elastic compute and storage capabilities. Various IaaS services are provided by Oracle Corporation®.

[0160] A PaaS model is generally used to provide, as a service, platform and environment resources that enable customers to develop, run, and manage applications and services without the customer having to procure, build, or maintain such resources. Examples of PaaS
25 services provided by Oracle Corporation® include, without limitation, Oracle Java Cloud Service (JCS), Oracle Database Cloud Service (DBCS), data management cloud service, various application development solutions services, and others.

[0161] Cloud services are generally provided on an on-demand self-service basis, subscription-based, elastically scalable, reliable, highly available, and secure manner. For
30 example, a customer, via a subscription order, may order one or more services provided by cloud infrastructure system 702. Cloud infrastructure system 702 then performs processing to

provide the services requested in the customer's subscription order. For example, a user may use utterances to request the cloud infrastructure system to take a certain action (e.g., an intent), as described above, and/or provide services for a chatbot system as described herein. Cloud infrastructure system 702 may be configured to provide one or even multiple cloud services.

[0162] Cloud infrastructure system 702 may provide the cloud services via different deployment models. In a public cloud model, cloud infrastructure system 702 may be owned by a third party cloud services provider and the cloud services are offered to any general public customer, where the customer may be an individual or an enterprise. In certain other examples, under a private cloud model, cloud infrastructure system 702 may be operated within an organization (e.g., within an enterprise organization) and services provided to customers that are within the organization. For example, the customers may be various departments of an enterprise such as the Human Resources department, the Payroll department, etc. or even individuals within the enterprise. In certain other examples, under a community cloud model, the cloud infrastructure system 702 and the services provided may be shared by several organizations in a related community. Various other models such as hybrids of the above mentioned models may also be used.

[0163] Client computing devices 704, 706, and 708 may be of different types (such as client computing devices 602, 604, 606, and 608 depicted in FIG. 6) and may be capable of operating one or more client applications. A user may use a client device to interact with cloud infrastructure system 702, such as to request a service provided by cloud infrastructure system 702. For example, a user may use a client device to request information or action from a chatbot as described in this disclosure.

[0164] In some examples, the processing performed by cloud infrastructure system 702 for providing services may involve model training and deployment. This analysis may involve using, analyzing, and manipulating data sets to train and deploy one or more models. This analysis may be performed by one or more processors, possibly processing the data in parallel, performing simulations using the data, and the like. For example, big data analysis may be performed by cloud infrastructure system 702 for generating and training one or more models for a chatbot system. The data used for this analysis may include structured data (e.g., data stored in a database or structured according to a structured model) and/or unstructured data (e.g., data blobs (binary large objects)).

[0165] As depicted in the example in FIG. 7, cloud infrastructure system 702 may include infrastructure resources 730 that are utilized for facilitating the provision of various cloud services offered by cloud infrastructure system 702. Infrastructure resources 730 may include, for example, processing resources, storage or memory resources, networking
5 resources, and the like. In certain examples, the storage virtual machines that are available for servicing storage requested from applications may be part of cloud infrastructure system 702. In other examples, the storage virtual machines may be part of different systems.

[0166] In certain examples, to facilitate efficient provisioning of these resources for supporting the various cloud services provided by cloud infrastructure system 702 for
10 different customers, the resources may be bundled into sets of resources or resource modules (also referred to as "pods"). Each resource module or pod may comprise a pre-integrated and optimized combination of resources of one or more types. In certain examples, different pods may be pre-provisioned for different types of cloud services. For example, a first set of pods may be provisioned for a database service, a second set of pods, which may include a
15 different combination of resources than a pod in the first set of pods, may be provisioned for Java service, and the like. For some services, the resources allocated for provisioning the services may be shared between the services.

[0167] Cloud infrastructure system 702 may itself internally use services 732 that are shared by different components of cloud infrastructure system 702 and which facilitate the
20 provisioning of services by cloud infrastructure system 702. These internal shared services may include, without limitation, a security and identity service, an integration service, an enterprise repository service, an enterprise manager service, a virus scanning and white list service, a high availability, backup and recovery service, service for enabling cloud support, an email service, a notification service, a file transfer service, and the like.

[0168] Cloud infrastructure system 702 may comprise multiple subsystems. These subsystems may be implemented in software, or hardware, or combinations thereof. As depicted in FIG. 7, the subsystems may include a user interface subsystem 712 that enables users or customers of cloud infrastructure system 702 to interact with cloud infrastructure system 702. User interface subsystem 712 may include various different interfaces such as a
30 web interface 714, an online store interface 716 where cloud services provided by cloud infrastructure system 702 are advertised and are purchasable by a consumer, and other interfaces 718. For example, a customer may, using a client device, request (service request

734) one or more services provided by cloud infrastructure system 702 using one or more of interfaces 714, 716, and 718. For example, a customer may access the online store, browse cloud services offered by cloud infrastructure system 702, and place a subscription order for one or more services offered by cloud infrastructure system 702 that the customer wishes to
5 subscribe to. The service request may include information identifying the customer and one or more services that the customer desires to subscribe to. For example, a customer may place a subscription order for a service offered by cloud infrastructure system 702. As part of the order, the customer may provide information identifying a chatbot system for which the service is to be provided and optionally one or more credentials for the chatbot system.

10 **[0169]** In certain examples, such as the example depicted in FIG. 7, cloud infrastructure system 702 may comprise an order management subsystem (OMS) 720 that is configured to process the new order. As part of this processing, OMS 720 may be configured to: create an account for the customer, if not done already; receive billing and/or accounting information from the customer that is to be used for billing the customer for providing the requested
15 service to the customer; verify the customer information; upon verification, book the order for the customer; and orchestrate various workflows to prepare the order for provisioning.

[0170] Once properly validated, OMS 720 may then invoke the order provisioning subsystem (OPS) 724 that is configured to provision resources for the order including processing, memory, and networking resources. The provisioning may include allocating
20 resources for the order and configuring the resources to facilitate the service requested by the customer order. The manner in which resources are provisioned for an order and the type of the provisioned resources may depend upon the type of cloud service that has been ordered by the customer. For example, according to one workflow, OPS 724 may be configured to determine the particular cloud service being requested and identify a number of pods that
25 may have been pre-configured for that particular cloud service. The number of pods that are allocated for an order may depend upon the size/amount/level/scope of the requested service. For example, the number of pods to be allocated may be determined based upon the number of users to be supported by the service, the duration of time for which the service is being requested, and the like. The allocated pods may then be customized for the particular
30 requesting customer for providing the requested service.

[0171] In certain examples, setup phase processing, as described above, may be performed by cloud infrastructure system 702 as part of the provisioning process. Cloud infrastructure

system 702 may generate an application ID and select a storage virtual machine for an application from among storage virtual machines provided by cloud infrastructure system 702 itself or from storage virtual machines provided by other systems other than cloud infrastructure system 702.

5 **[0172]** Cloud infrastructure system 702 may send a response or notification 744 to the requesting customer to indicate when the requested service is now ready for use. In some instances, information (e.g., a link) may be sent to the customer that enables the customer to start using and availing the benefits of the requested services. In certain examples, for a customer requesting the service, the response may include a chatbot system ID generated by
10 cloud infrastructure system 702 and information identifying a chatbot system selected by cloud infrastructure system 702 for the chatbot system corresponding to the chatbot system ID.

[0173] Cloud infrastructure system 702 may provide services to multiple customers. For each customer, cloud infrastructure system 702 is responsible for managing information
15 related to one or more subscription orders received from the customer, maintaining customer data related to the orders, and providing the requested services to the customer. Cloud infrastructure system 702 may also collect usage statistics regarding a customer's use of subscribed services. For example, statistics may be collected for the amount of storage used, the amount of data transferred, the number of users, and the amount of system up time and
20 system down time, and the like. This usage information may be used to bill the customer. Billing may be done, for example, on a monthly cycle.

[0174] Cloud infrastructure system 702 may provide services to multiple customers in parallel. Cloud infrastructure system 702 may store information for these customers, including possibly proprietary information. In certain examples, cloud infrastructure system
25 702 comprises an identity management subsystem (IMS) 728 that is configured to manage customer information and provide the separation of the managed information such that information related to one customer is not accessible by another customer. IMS 728 may be configured to provide various security-related services such as identity services, such as information access management, authentication and authorization services, services for
30 managing customer identities and roles and related capabilities, and the like.

[0175] FIG. 8 illustrates an example of computer system 800. In some examples, computer system 800 may be used to implement any of the digital assistant or chatbot systems within a

distributed environment, and various servers and computer systems described above. As shown in FIG. 8, computer system 800 includes various subsystems including a processing subsystem 804 that communicates with a number of other subsystems via a bus subsystem 802. These other subsystems may include a processing acceleration unit 806, an I/O subsystem 808, a storage subsystem 818, and a communications subsystem 824. Storage subsystem 818 may include non-transitory computer-readable storage media including storage media 822 and a system memory 810.

[0176] Bus subsystem 802 provides a mechanism for letting the various components and subsystems of computer system 800 communicate with each other as intended. Although bus subsystem 802 is shown schematically as a single bus, alternative examples of the bus subsystem may utilize multiple buses. Bus subsystem 802 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, a local bus using any of a variety of bus architectures, and the like. For example, such architectures may include an Industry Standard Architecture (ISA) bus, Micro Channel Architecture (MCA) bus, Enhanced ISA (EISA) bus, Video Electronics Standards Association (VESA) local bus, and Peripheral Component Interconnect (PCI) bus, which may be implemented as a Mezzanine bus manufactured to the IEEE P1386.1 standard, and the like.

[0177] Processing subsystem 804 controls the operation of computer system 800 and may comprise one or more processors, application specific integrated circuits (ASICs), or field programmable gate arrays (FPGAs). The processors may include be single core or multicore processors. The processing resources of computer system 800 may be organized into one or more processing units 832, 834, etc. A processing unit may include one or more processors, one or more cores from the same or different processors, a combination of cores and processors, or other combinations of cores and processors. In some examples, processing subsystem 804 may include one or more special purpose co-processors such as graphics processors, digital signal processors (DSPs), or the like. In some examples, some or all of the processing units of processing subsystem 804 may be implemented using customized circuits, such as application specific integrated circuits (ASICs), or field programmable gate arrays (FPGAs).

[0178] In some examples, the processing units in processing subsystem 804 may execute instructions stored in system memory 810 or on computer readable storage media 822. In various examples, the processing units may execute a variety of programs or code

instructions and may maintain multiple concurrently executing programs or processes. At any given time, some or all of the program code to be executed may be resident in system memory 810 and/or on computer-readable storage media 822 including potentially on one or more storage devices. Through suitable programming, processing subsystem 804 may
5 provide various functionalities described above. In instances where computer system 800 is executing one or more virtual machines, one or more processing units may be allocated to each virtual machine.

[0179] In certain examples, a processing acceleration unit 806 may optionally be provided for performing customized processing or for off-loading some of the processing performed by
10 processing subsystem 804 so as to accelerate the overall processing performed by computer system 800.

[0180] I/O subsystem 808 may include devices and mechanisms for inputting information to computer system 800 and/or for outputting information from or via computer system 800. In general, use of the term input device is intended to include all possible types of devices
15 and mechanisms for inputting information to computer system 800. User interface input devices may include, for example, a keyboard, pointing devices such as a mouse or trackball, a touchpad or touch screen incorporated into a display, a scroll wheel, a click wheel, a dial, a button, a switch, a keypad, audio input devices with voice command recognition systems, microphones, and other types of input devices. User interface input devices may also include
20 motion sensing and/or gesture recognition devices such as the Microsoft Kinect[®] motion sensor that enables users to control and interact with an input device, the Microsoft Xbox[®] 360 game controller, devices that provide an interface for receiving input using gestures and spoken commands. User interface input devices may also include eye gesture recognition devices such as the Google Glass[®] blink detector that detects eye activity (e.g., "blinking"
25 while taking pictures and/or making a menu selection) from users and transforms the eye gestures as inputs to an input device (e.g., Google Glass[®]). Additionally, user interface input devices may include voice recognition sensing devices that enable users to interact with voice recognition systems (e.g., Siri[®] navigator) through voice commands.

[0181] Other examples of user interface input devices include, without limitation, three
30 dimensional (3D) mice, joysticks or pointing sticks, gamepads and graphic tablets, and audio/visual devices such as speakers, digital cameras, digital camcorders, portable media players, webcams, image scanners, fingerprint scanners, barcode reader 3D scanners, 3D

printers, laser rangefinders, and eye gaze tracking devices. Additionally, user interface input devices may include, for example, medical imaging input devices such as computed tomography, magnetic resonance imaging, position emission tomography, and medical ultrasonography devices. User interface input devices may also include, for example, audio input devices such as MIDI keyboards, digital musical instruments and the like.

[0182] In general, use of the term output device is intended to include all possible types of devices and mechanisms for outputting information from computer system 800 to a user or other computer. User interface output devices may include a display subsystem, indicator lights, or non-visual displays such as audio output devices, etc. The display subsystem may be a cathode ray tube (CRT), a flat-panel device, such as that using a liquid crystal display (LCD) or plasma display, a projection device, a touch screen, and the like. For example, user interface output devices may include, without limitation, a variety of display devices that visually convey text, graphics and audio/video information such as monitors, printers, speakers, headphones, automotive navigation systems, plotters, voice output devices, and modems.

[0183] Storage subsystem 818 provides a repository or data store for storing information and data that is used by computer system 800. Storage subsystem 818 provides a tangible non-transitory computer-readable storage medium for storing the basic programming and data constructs that provide the functionality of some examples. Storage subsystem 818 may store software (e.g., programs, code modules, instructions) that when executed by processing subsystem 804 provides the functionality described above. The software may be executed by one or more processing units of processing subsystem 804. Storage subsystem 818 may also provide authentication in accordance with the teachings of this disclosure.

[0184] Storage subsystem 818 may include one or more non-transitory memory devices, including volatile and non-volatile memory devices. As shown in FIG. 8, storage subsystem 818 includes a system memory 810 and a computer-readable storage media 822. System memory 810 may include a number of memories including a volatile main random access memory (RAM) for storage of instructions and data during program execution and a non-volatile read only memory (ROM) or flash memory in which fixed instructions are stored. In some implementations, a basic input/output system (BIOS), containing the basic routines that help to transfer information between elements within computer system 800, such as during start-up, may typically be stored in the ROM. The RAM typically contains data and/or

program modules that are presently being operated and executed by processing subsystem 804. In some implementations, system memory 810 may include multiple different types of memory, such as static random access memory (SRAM), dynamic random access memory (DRAM), and the like.

- 5 **[0185]** By way of example, and not limitation, as depicted in FIG. 8, system memory 810 may load application programs 812 that are being executed, which may include various applications such as Web browsers, mid-tier applications, relational database management systems (RDBMS), etc., program data 814, and an operating system 816. By way of example, operating system 816 may include various versions of Microsoft Windows[®], Apple
- 10 Macintosh[®], and/or Linux operating systems, a variety of commercially-available UNIX[®] or UNIX-like operating systems (including without limitation the variety of GNU/Linux operating systems, the Google Chrome[®] OS, and the like) and/or mobile operating systems such as iOS, Windows[®] Phone, Android[®] OS, BlackBerry[®] OS, Palm[®] OS operating systems, and others.
- 15 **[0186]** Computer-readable storage media 822 may store programming and data constructs that provide the functionality of some examples. Computer-readable media 822 may provide storage of computer-readable instructions, data structures, program modules, and other data for computer system 800. Software (programs, code modules, instructions) that, when executed by processing subsystem 804 provides the functionality described above, may be
- 20 stored in storage subsystem 818. By way of example, computer-readable storage media 822 may include non-volatile memory such as a hard disk drive, a magnetic disk drive, an optical disk drive such as a CD ROM, DVD, a Blu-Ray[®] disk, or other optical media. Computer-readable storage media 822 may include, but is not limited to, Zip[®] drives, flash memory cards, universal serial bus (USB) flash drives, secure digital (SD) cards, DVD disks, digital
- 25 video tape, and the like. Computer-readable storage media 822 may also include, solid-state drives (SSD) based on non-volatile memory such as flash-memory based SSDs, enterprise flash drives, solid state ROM, and the like, SSDs based on volatile memory such as solid state RAM, dynamic RAM, static RAM, DRAM-based SSDs, magnetoresistive RAM (MRAM) SSDs, and hybrid SSDs that use a combination of DRAM and flash memory based
- 30 SSDs.

[0187] In certain examples, storage subsystem 818 may also include a computer-readable storage media reader 820 that may further be connected to computer-readable storage media

822. Reader 820 may receive and be configured to read data from a memory device such as a disk, a flash drive, etc.

[0188] In certain examples, computer system 800 may support virtualization technologies, including but not limited to virtualization of processing and memory resources. For example, computer system 800 may provide support for executing one or more virtual machines. In certain examples, computer system 800 may execute a program such as a hypervisor that facilitated the configuring and managing of the virtual machines. Each virtual machine may be allocated memory, compute (e.g., processors, cores), I/O, and networking resources. Each virtual machine generally runs independently of the other virtual machines. A virtual machine typically runs its own operating system, which may be the same as or different from the operating systems executed by other virtual machines executed by computer system 800. Accordingly, multiple operating systems may potentially be run concurrently by computer system 800.

[0189] Communications subsystem 824 provides an interface to other computer systems and networks. Communications subsystem 824 serves as an interface for receiving data from and transmitting data to other systems from computer system 800. For example, communications subsystem 824 may enable computer system 800 to establish a communication channel to one or more client devices via the Internet for receiving and sending information from and to the client devices. For example, when computer system 800 is used to implement bot system 120 depicted in FIG. 1, the communication subsystem may be used to communicate with a chatbot system selected for an application.

[0190] Communication subsystem 824 may support both wired and/or wireless communication protocols. In certain examples, communications subsystem 824 may include radio frequency (RF) transceiver components for accessing wireless voice and/or data networks (e.g., using cellular telephone technology, advanced data network technology, such as 3G, 4G or EDGE (enhanced data rates for global evolution), WiFi (IEEE 802.XX family standards, or other mobile communication technologies, or any combination thereof), global positioning system (GPS) receiver components, and/or other components. In some examples, communications subsystem 824 may provide wired network connectivity (e.g., Ethernet) in addition to or instead of a wireless interface.

[0191] Communication subsystem 824 may receive and transmit data in various forms. In some examples, in addition to other forms, communications subsystem 824 may receive input

communications in the form of structured and/or unstructured data feeds 826, event streams 828, event updates 830, and the like. For example, communications subsystem 824 may be configured to receive (or send) data feeds 826 in real-time from users of social media networks and/or other communication services such as Twitter[®] feeds, Facebook[®] updates, web feeds such as Rich Site Summary (RSS) feeds, and/or real-time updates from one or more third party information sources.

[0192] In certain examples, communications subsystem 824 may be configured to receive data in the form of continuous data streams, which may include event streams 828 of real-time events and/or event updates 830, that may be continuous or unbounded in nature with no explicit end. Examples of applications that generate continuous data may include, for example, sensor data applications, financial tickers, network performance measuring tools (e.g. network monitoring and traffic management applications), clickstream analysis tools, automobile traffic monitoring, and the like.

[0193] Communications subsystem 824 may also be configured to communicate data from computer system 800 to other computer systems or networks. The data may be communicated in various different forms such as structured and/or unstructured data feeds 826, event streams 828, event updates 830, and the like to one or more databases that may be in communication with one or more streaming data source computers coupled to computer system 800.

[0194] Computer system 800 may be one of various types, including a handheld portable device (e.g., an iPhone[®] cellular phone, an iPad[®] computing tablet, a PDA), a wearable device (e.g., a Google Glass[®] head mounted display), a personal computer, a workstation, a mainframe, a kiosk, a server rack, or any other data processing system. Due to the ever-changing nature of computers and networks, the description of computer system 800 depicted in FIG. 8 is intended only as a specific example. Many other configurations having more or fewer components than the system depicted in FIG. 8 are possible. Based on the disclosure and teachings provided herein, it should be appreciated there are other ways and/or methods to implement the various examples.

Example

Example 1

[0195] A machine-learning model was constructed that includes a BERT model and a decoder model. The BERT model included a 12-layer bidirectional transformer encoder (as illustrated in FIG. 9A) with 12 bidirectional self-attention heads. In each layer, the transformer includes an encoder. As indicated in FIG. 9B, the encoder of the transformer includes a multi-head attention layer, which uses values (V), keys (K), and queries (Q) as described herein to determine how to transform inputs based not only on the individual values in the input but also based on other values and the attention paid to the other values. (See FIG. 9C.)

[0196] The BERT model (e.g., a multilingual BERT model) was trained using a training data set. During this training, values of parameters across all twelve layers were learned. Each of multiple fine-tuning processes were then performed to fine-tune the model. Each process corresponds to a particular data set to use in the fine-tuning and an identification of one or more layers for which parameter values in the multi-head self-attention networks are to be fine-tuned. The trained BERT model may include one as identified at <https://github.com/google-research/bert/blob/master/multilingual.md>, which is hereby incorporated by reference for all purposes. Across the fine-tuning processes, each of eight data sets were used for the fine-tuning. Each of these eight data sets is a public data set that is identified in a heading in the table shown in FIG. 10. The different rows in the table shown in FIG. 10 indicate for how many layers the multi-head self-attention networks were fine-tuned. For each combination of the data set used for the fine-tuning and the number of layers for which the multi-head self-attention networks are fine-tuned, the table shown in FIG. 10 identifies micro F1 scores generated by using the fine-tuned model predicted labels and the test data of the corresponding publicly available dataset. The last column shows the average micro F1 scores for fine-tuning the indicated number of layers across data sets.

[0197] As shown in FIG. 10, the highest accuracy was achieved when the self-attention networks of between 5 and 9 of the layers were fine-tuned – not all 12. On average, the highest accuracy was achieved when only the multihead self attention network of the layer 8 was fine-tuned. Additionally, fine-tuning only the multi-head attention networks within the transformer block in the selected layer reduced the time for fine-tuning by 75%.

Example 2

[0198] A machine-learning model (with layers as illustrated in FIG. 11) is constructed that includes the same architecture as identified in Example 1. This architecture includes a BERT

model 1105 with a 12-layer bidirectional transformer encoder and activation layer 1110. An initial set of layers 1115 (including all layers in the BERT model and the decoder model / activation layer) are trained during an initial training stage using an initial training data set. A fine-tuning set of layers 1120 are identified as being an incomplete subset of the initial set of layers 1115. Only part of a single layer (e.g., each multi-head attention component) in the fine-tuning layer 1120 is the fine-tuned using a different (e.g., context-specific) training set. The fine-tuned model is then configured to process data corresponding to the context, while using less context-specific data than would be required if the whole machine-learning model was to be trained using context-specific data.

10 **Additional Considerations:**

[0199] Although specific details and examples have been described, various modifications, alterations, alternative constructions, and equivalents are possible. Examples are not restricted to operation within certain specific data processing environments, but are free to operate within a plurality of data processing environments. Additionally, although certain examples have been described using a particular series of transactions and steps, it should be apparent to those skilled in the art that this is not intended to be limiting. Although some flowcharts describe operations as a sequential process, many of the operations may be performed in parallel or concurrently. In addition, the order of the operations may be rearranged. A process may have additional steps not included in the figure. Various features and aspects of the above-described examples may be used individually or jointly.

[0200] Further, while certain examples have been described using a particular combination of hardware and software, it should be recognized that other combinations of hardware and software are also possible. Certain examples may be implemented only in hardware, or only in software, or using combinations thereof. The various processes described herein may be implemented on the same processor or different processors in any combination.

[0201] Where devices, systems, components or modules are described as being configured to perform certain operations or functions, such configuration may be accomplished, for example, by designing electronic circuits to perform the operation, by programming programmable electronic circuits (such as microprocessors) to perform the operation such as by executing computer instructions or code, or processors or cores programmed to execute code or instructions stored on a non-transitory memory medium, or any combination thereof. Processes may communicate using a variety of techniques including but not limited to

conventional techniques for inter-process communications, and different pairs of processes may use different techniques, or the same pair of processes may use different techniques at different times.

[0202] Specific details are given in this disclosure to provide a thorough understanding of the examples. However, examples may be practiced without these specific details. For example, well-known circuits, processes, algorithms, structures, and techniques have been shown without unnecessary detail in order to avoid obscuring the examples. This description provides example examples only, and is not intended to limit the scope, applicability, or configuration of other examples. Rather, the preceding description of the examples will provide those skilled in the art with an enabling description for implementing various examples. Various changes may be made in the function and arrangement of elements.

[0203] The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense. It will, however, be evident that additions, subtractions, deletions, and other modifications and changes may be made thereunto without departing from the broader spirit and scope as set forth in the claims. Thus, although specific examples have been described, these are not intended to be limiting. Various modifications and equivalents are within the scope of the following claims.

[0204] In the foregoing specification, aspects of the disclosure are described with reference to specific examples thereof, but those skilled in the art will recognize that the disclosure is not limited thereto. Various features and aspects of the above-described disclosure may be used individually or jointly. Further, examples may be utilized in any number of environments and applications beyond those described herein without departing from the broader spirit and scope of the specification. The specification and drawings are, accordingly, to be regarded as illustrative rather than restrictive.

[0205] In the foregoing description, for the purposes of illustration, methods were described in a particular order. It should be appreciated that in alternate examples, the methods may be performed in a different order than that described. It should also be appreciated that the methods described above may be performed by hardware components or may be embodied in sequences of machine-executable instructions, which may be used to cause a machine, such as a general-purpose or special-purpose processor or logic circuits programmed with the instructions to perform the methods. These machine-executable instructions may be stored on one or more machine readable mediums, such as CD-ROMs or

other type of optical disks, floppy diskettes, ROMs, RAMs, EPROMs, EEPROMs, magnetic or optical cards, flash memory, or other types of machine-readable mediums suitable for storing electronic instructions. Alternatively, the methods may be performed by a combination of hardware and software.

- 5 **[0206]** Where components are described as being configured to perform certain operations, such configuration may be accomplished, for example, by designing electronic circuits or other hardware to perform the operation, by programming programmable electronic circuits (e.g., microprocessors, or other suitable electronic circuits) to perform the operation, or any combination thereof.
- 10 **[0207]** While illustrative examples of the application have been described in detail herein, it is to be understood that the inventive concepts may be otherwise variously embodied and employed, and that the appended claims are intended to be construed to include such variations, except as limited by the prior art.

WHAT IS CLAIMED IS:

1. A method, comprising:

accessing a pre-trained version of a machine-learning model that includes multiple layers and is configured to process audio or textual language input, each of the multiple
5 layers being configured with a plurality of layer-specific pre-trained parameter values corresponding to a plurality of parameters, and each of the multiple layers being configured to implement a multi-head attention technique;

accessing a client data set;

10 identifying an incomplete subset of the multiple layers for which corresponding layer-specific pre-trained parameter values are to be fine-tuned using the client data set,

fine-tuning the machine-learning model using the client data set to generate an updated version of the machine-learning model, wherein the layer-specific pre-trained parameter values configured for each layer of one or more of the multiple layers not included in the incomplete subset are frozen during the fine-tuning; and

15 facilitating use of the updated version of the machine-learning model.

2. The method of claim 1, wherein the multi-head attention technique includes a self-attention technique.

20 3. The method of claim 1, wherein the incomplete subset of the multiple layers includes two layers or one layer, and wherein the pre-trained machine-learning model includes at least five layers.

4. The method of claim 1, wherein the plurality of parameters includes at least
25 500,000 parameters, and wherein the updating the plurality of parameter values is performed using one or more Central Processing Units.

5. The method of claim 1, wherein the updating of the parameter values includes:
accessing a set of cached values corresponding to the one or more of the multiple layers not
30 included in the incomplete subsets.

6. The method of claim 1, wherein a quantity of the plurality of parameter values configured for the incomplete subset of the multiple layers is 30% or less as compared to a quantity parameters configured for the machine-learning model.

5 7. The method of claim 1, wherein facilitating use of the updated version includes translating linguistic input into a command or query.

8. The method of claim 1, wherein the multiple layers includes at least 4 layers, and wherein each of the incomplete subset of the multiple layers is positioned at Layer 2 or
10 higher in the multiple layers.

9. A system comprising:
one or more processors; and
a memory coupled to the one or more processors, the memory storing a plurality of
15 instructions executable by the one or more processors, the plurality of instructions comprising instructions that when executed by the one or more processors cause the one or more processors to perform a set of actions including:

accessing a pre-trained version of a machine-learning model that includes multiple layers and is configured to process audio or textual language input, each of
20 the multiple layers being configured with a plurality of layer-specific pre-trained parameter values corresponding to a plurality of parameters, and each of the multiple layers being configured to implement a multi-head attention technique;

accessing a client data set;

25 identifying an incomplete subset of the multiple layers for which corresponding layer-specific pre-trained parameter values are to be fine-tuned using the client data set,

fine-tuning the machine-learning model using the client data set to generate an updated version of the machine-learning model, wherein the layer-specific pre-trained parameter values configured for each layer of one of more of the multiple layers not
30 included in the incomplete subset are frozen during the fine-tuning; and

facilitating use of the updated version of the machine-learning model.

10. The system of claim 9, wherein the multi-head attention technique includes a self-attention technique.

11. The system of claim 9, wherein the incomplete subset of the multiple layers
5 includes two layers or one layer, and wherein the pre-trained machine-learning model includes at least five layers.

12. The system of claim 9, wherein the plurality of parameters includes at least 500,000 parameters, and wherein the updating the plurality of parameter values is performed
10 using one or more Central Processing Units.

13. The system of claim 9, wherein the updating of the parameter values includes: accessing a set of cached values corresponding to the one or more of the multiple layers not included in the incomplete subsets.
15

14. The system of claim 9, wherein a quantity of the plurality of parameter values configured for the incomplete subset of the multiple layers is 30% or less as compared to a quantity parameters configured for the machine-learning model.

15. The system of claim 9, wherein facilitating use of the updated version includes translating linguistic input into a command or query.
20

16. The system of claim 9, wherein the multiple layers includes at least 4 layers, and wherein each of the incomplete subset of the multiple layers is positioned at Layer 2 or
25 higher in the multiple layers.

17. A non-transitory computer-readable memory storing a plurality of instructions executable by one or more processors, the plurality of instructions comprising instructions that when executed by the one or more processors cause the one or more processors to
30 perform a set of actions including:

accessing a pre-trained version of a machine-learning model that includes multiple layers and is configured to process audio or textual language input, each of the multiple layers being configured with a plurality of layer-specific pre-trained parameter values

corresponding to a plurality of parameters, and each of the multiple layers being configured to implement a multi-head attention technique;

accessing a client data set;

5 identifying an incomplete subset of the multiple layers for which corresponding layer-specific pre-trained parameter values are to be fine-tuned using the client data set,

fine-tuning the machine-learning model using the client data set to generate an updated version of the machine-learning model, wherein the layer-specific pre-trained parameter values configured for each layer of one of more of the multiple layers not included in the incomplete subset are frozen during the fine-tuning; and

10 facilitating use of the updated version of the machine-learning model.

18. The non-transitory computer-readable memory of claim 17, wherein the multi-head attention technique includes a self-attention technique.

15 19. The non-transitory computer-readable memory of claim 17, wherein the incomplete subset of the multiple layers includes two layers or one layer, and wherein the pre-trained machine-learning model includes at least five layers.

20 20. The non-transitory computer-readable memory of claim 17, wherein the plurality of parameters includes at least 500,000 parameters, and wherein the updating the plurality of parameter values is performed using one or more Central Processing Units.

1/11

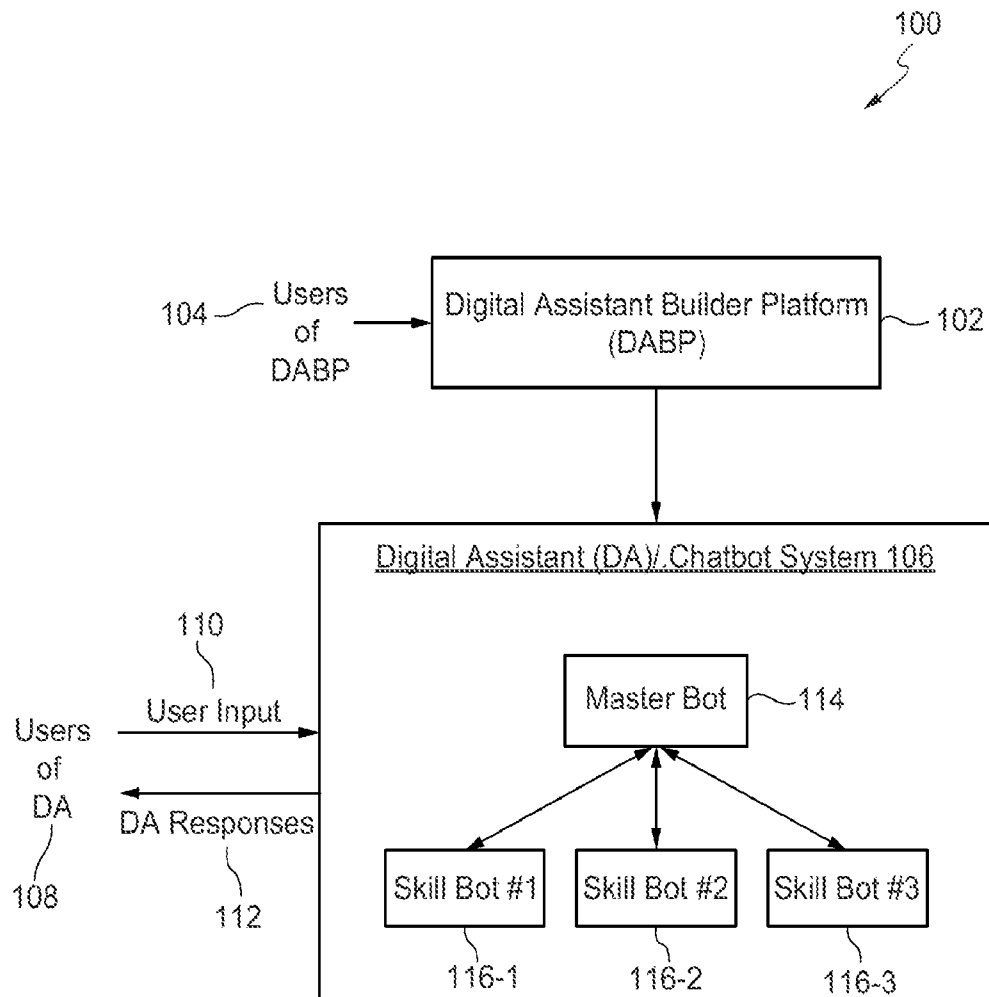


FIG. 1

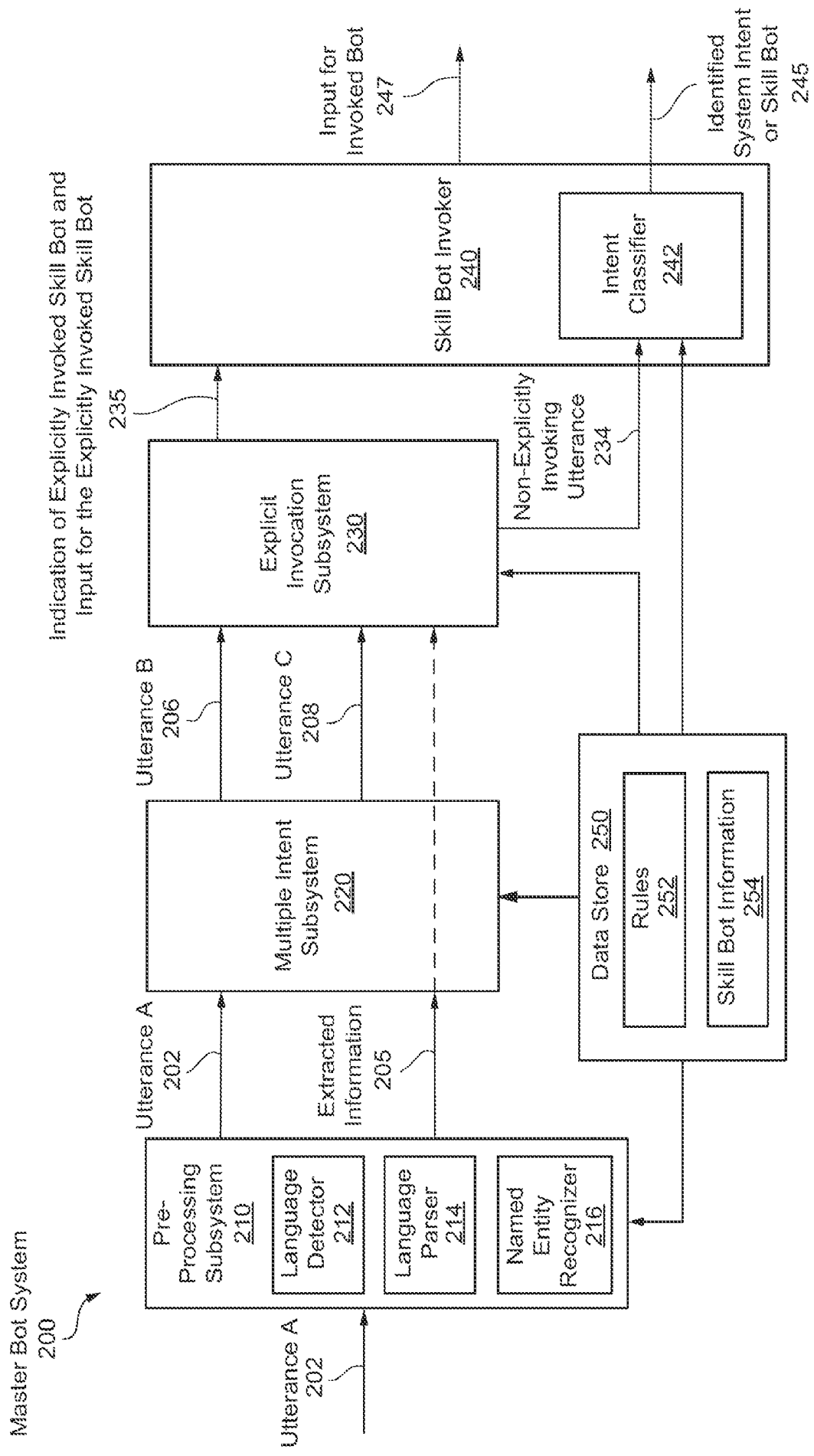


FIG. 2

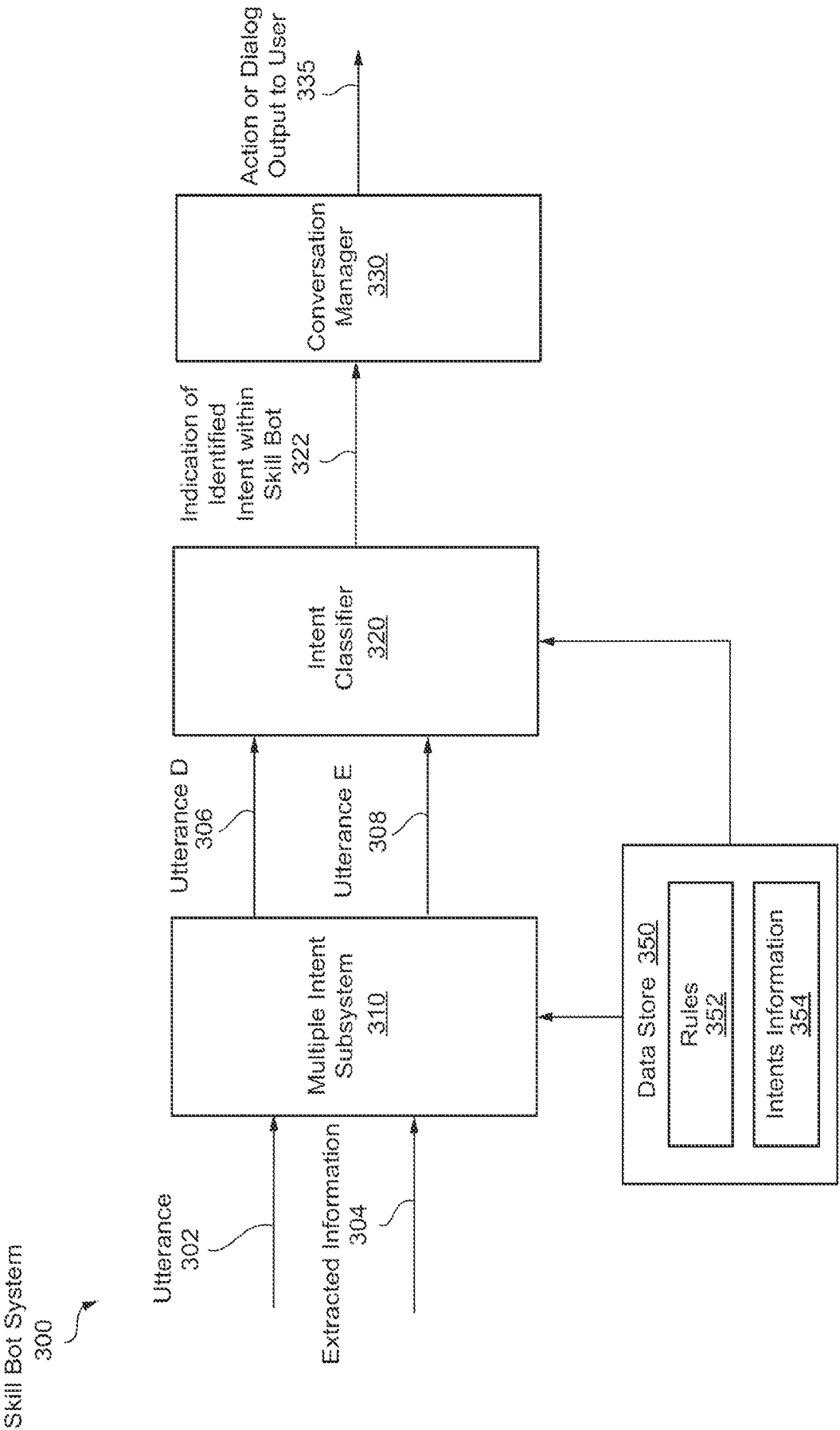


FIG. 3

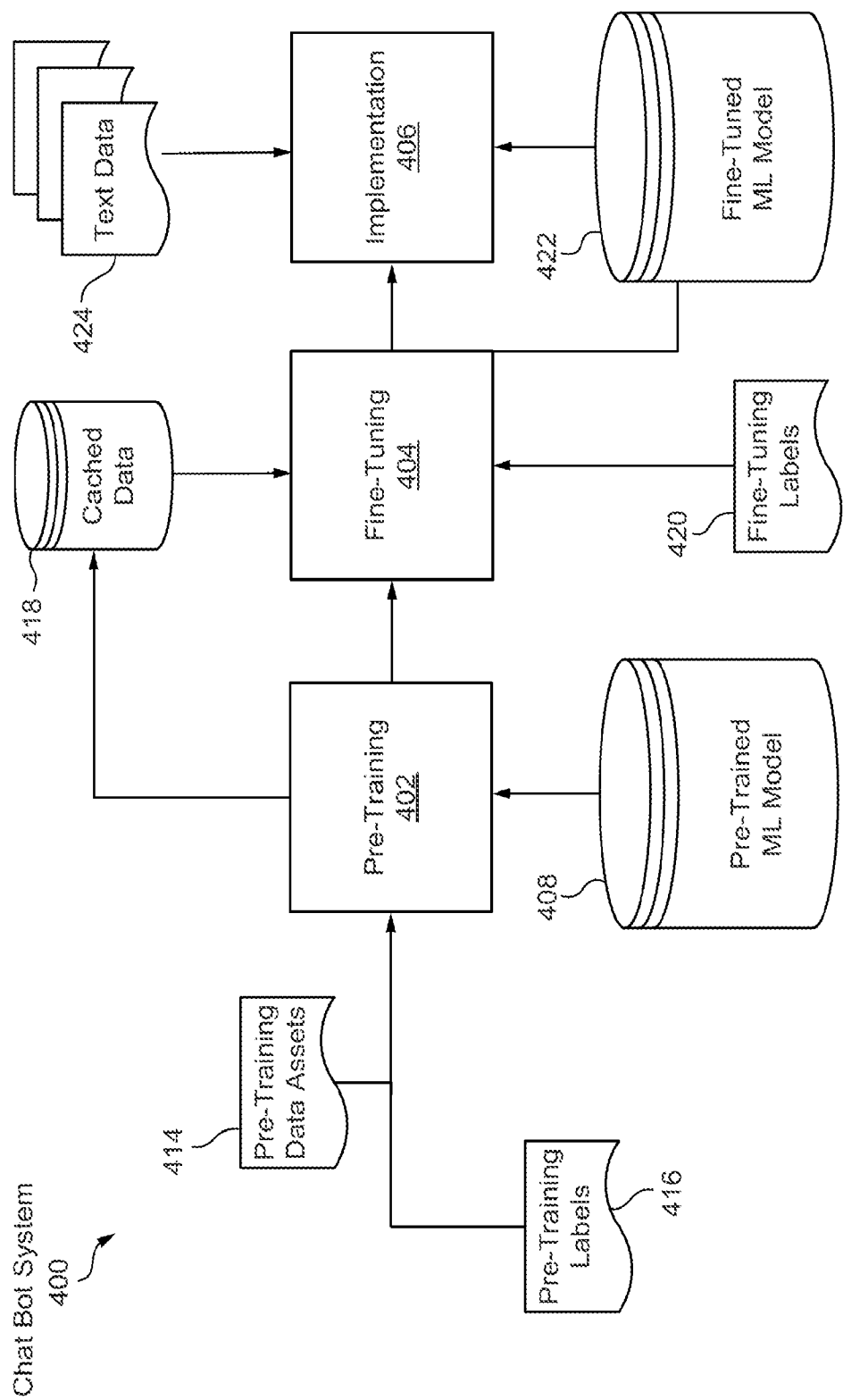


FIG. 4

5/11

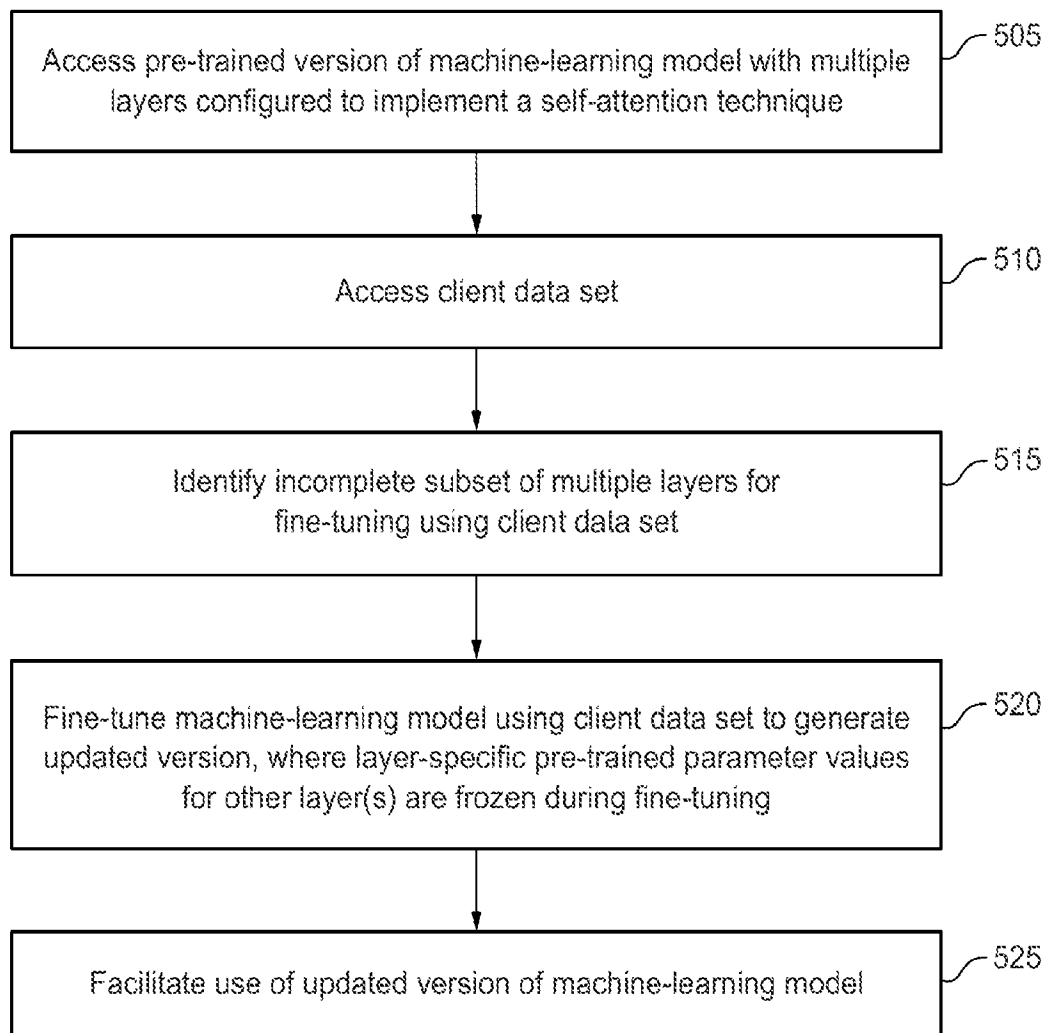


FIG. 5

6/11

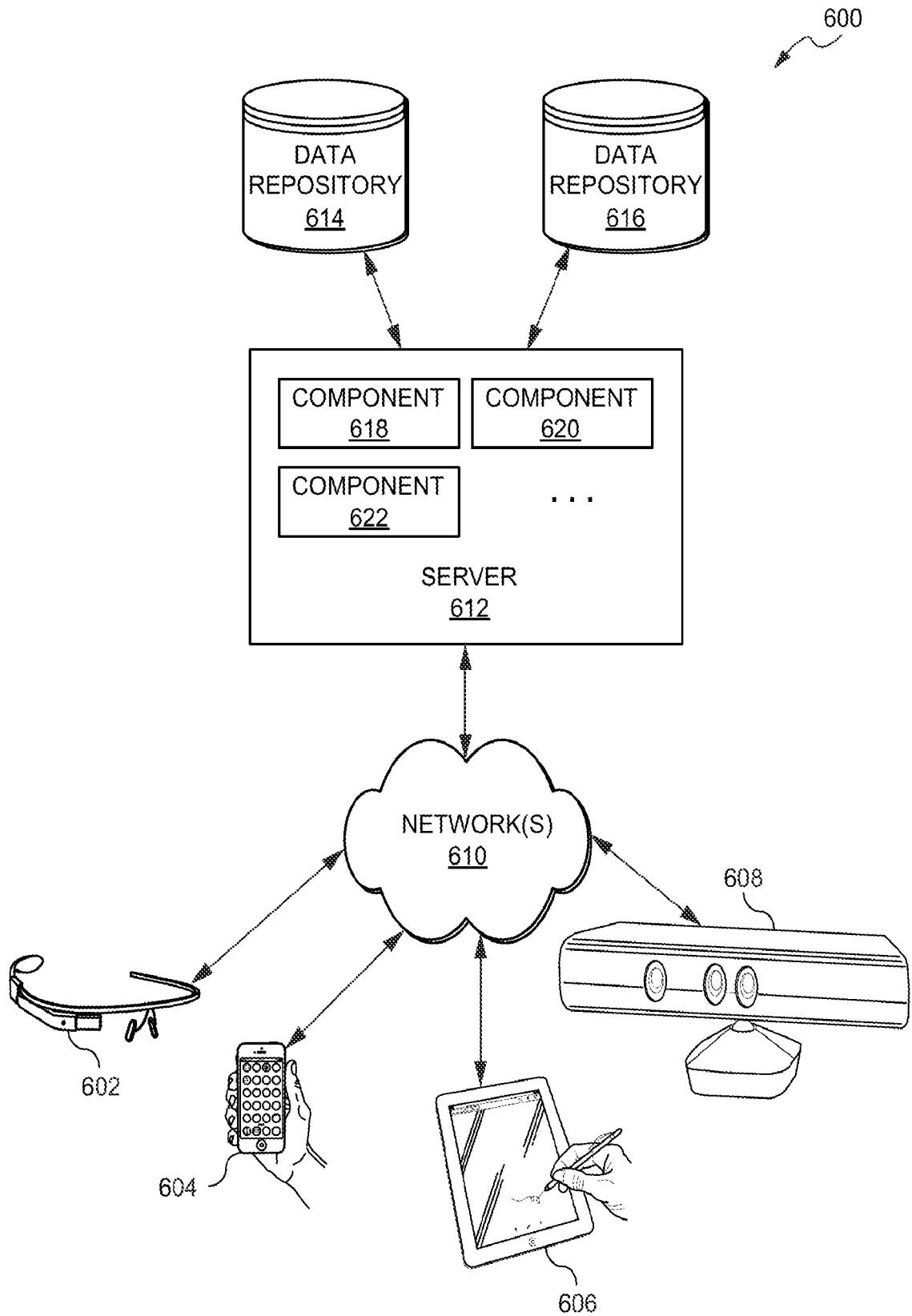


FIG. 6

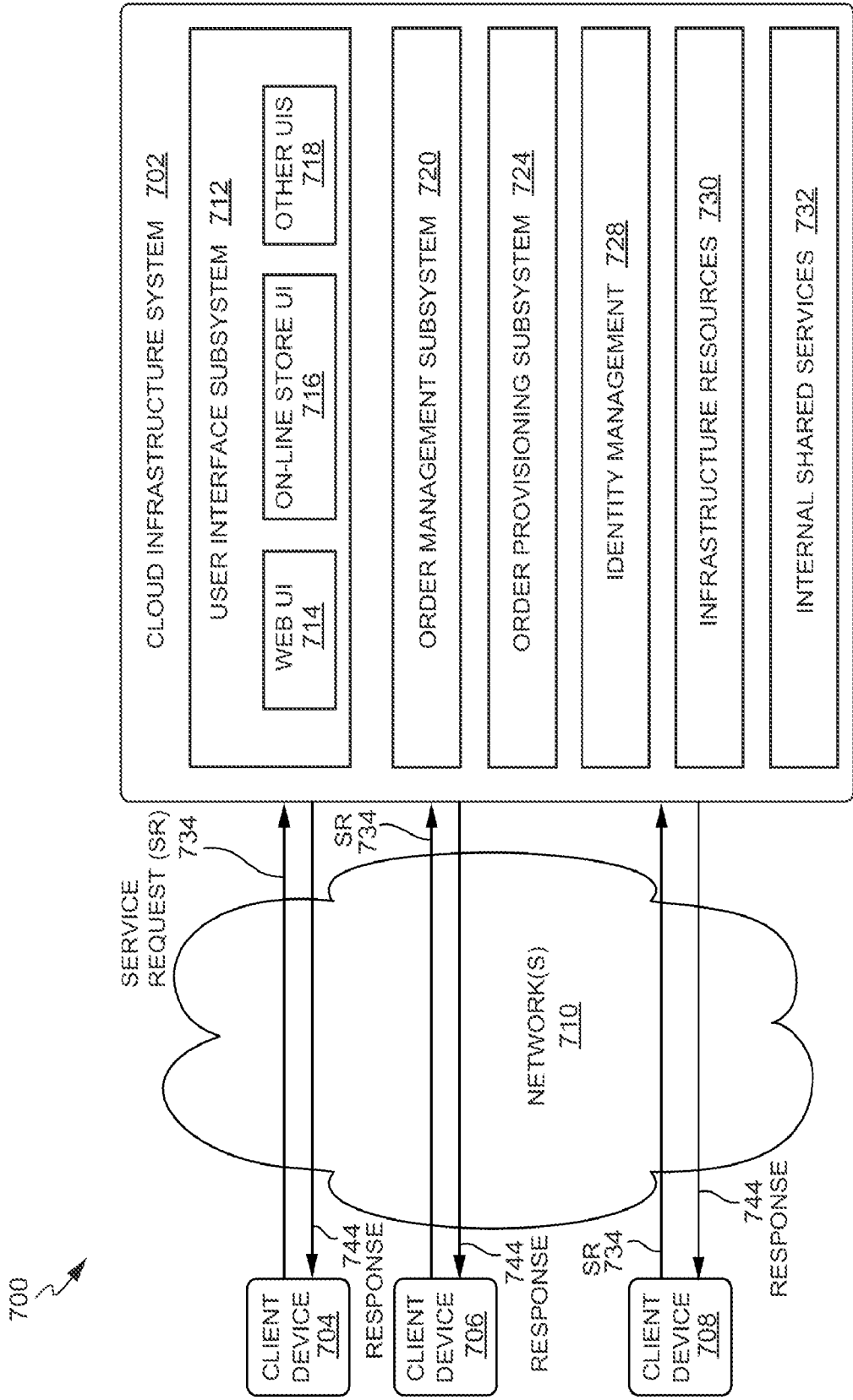


FIG. 7

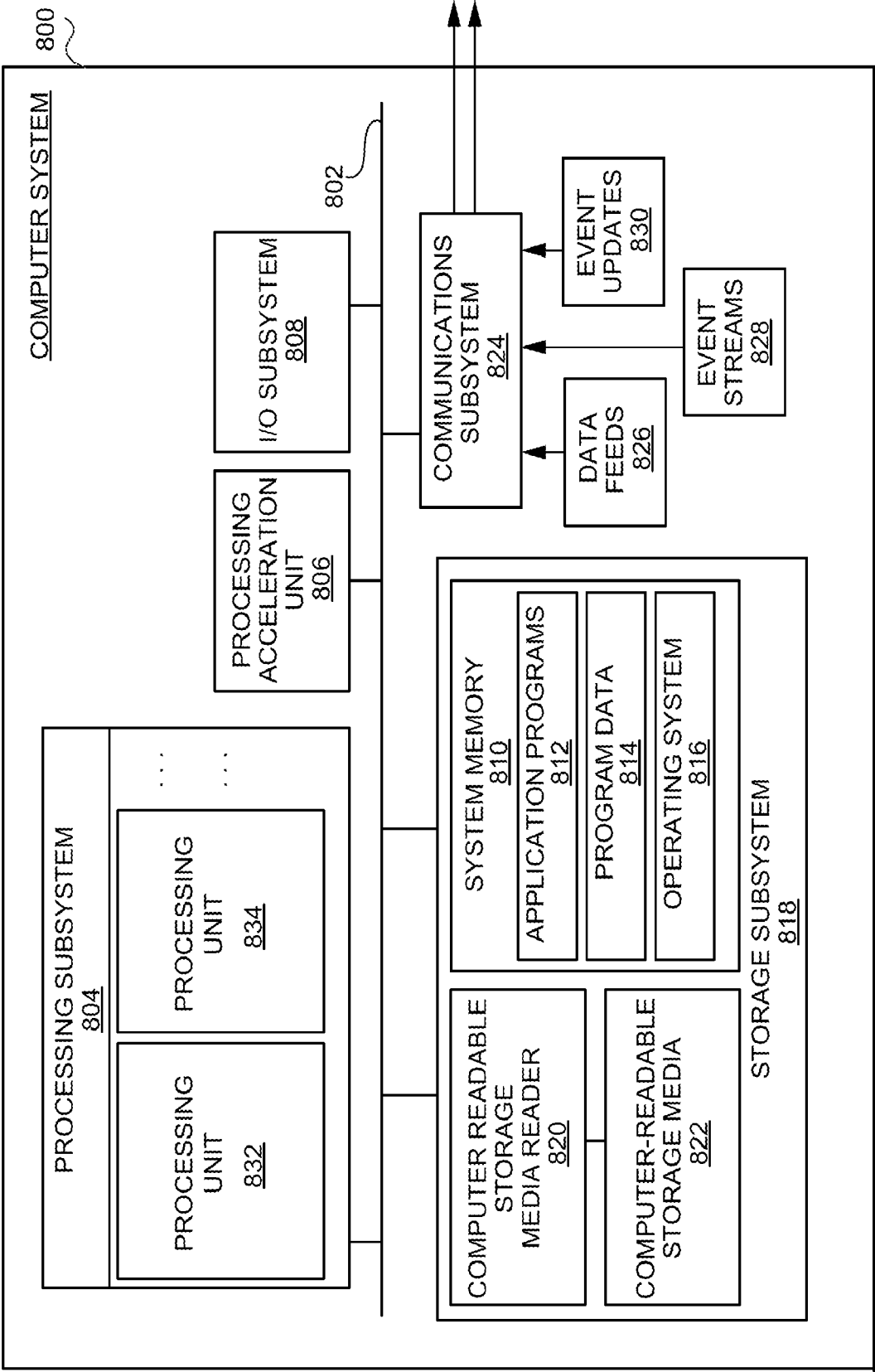


FIG. 8

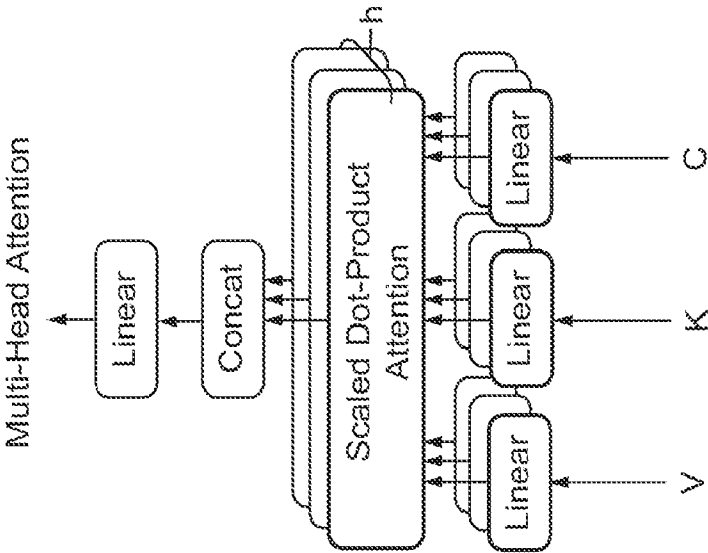


FIG. 9C

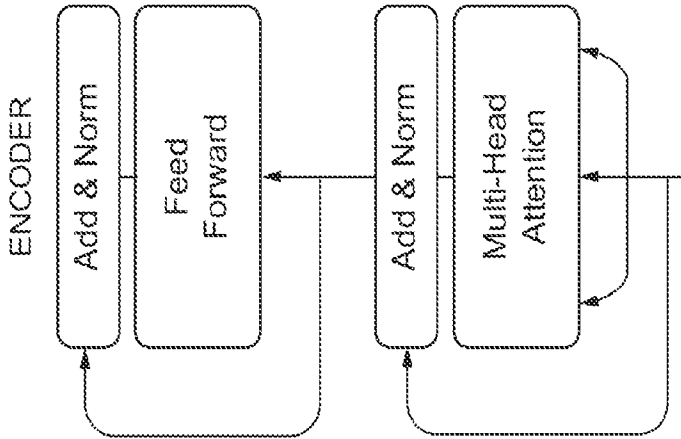


FIG. 9B

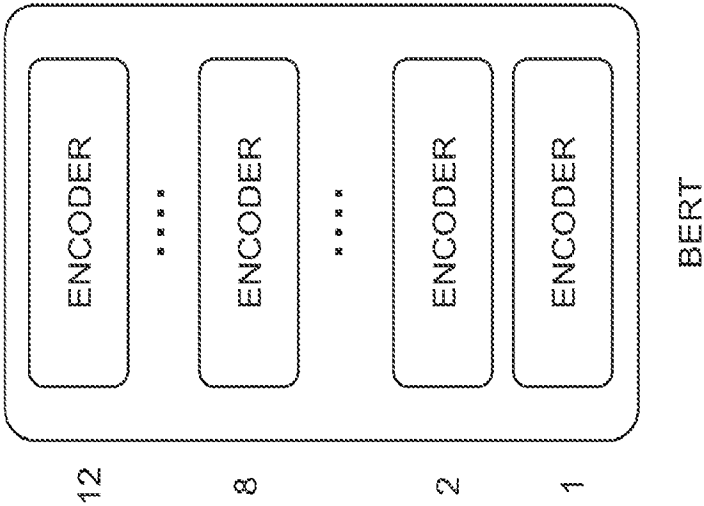


FIG. 9A

10/11

	CoNLL2003	MITMovie	MIT Restaurant	ATIS	MultiWOZ	SNIPS	OntoNotes _en	WikiGold	Average
n (MLP-CRF) + finetune multihead self attention only (1.7M trainable parameters)									
12	0.878	0.692	0.743	0.939	0.823	0.897	0.776	0.744	0.811
11	0.885	0.697	0.757	0.948	0.860	0.910	0.802	0.767	0.828
10	0.885	0.705	0.762	0.949	0.870	0.912	0.810	0.755	0.831
9	0.891	0.707	0.769	0.952	0.874	0.923	0.816	0.783	0.839
8	0.892	0.705	0.769	0.955	0.887	0.927	0.817	0.779	0.841
7	0.887	0.700	0.771	0.956	0.891	0.923	0.813	0.758	0.837
6	0.883	0.697	0.771	0.954	0.894	0.920	0.806	0.729	0.832
5	0.876	0.693	0.771	0.952	0.882	0.923	0.797	0.725	0.827
4	0.859	0.678	0.756	0.948	0.885	0.911	0.779	0.664	0.810
3	0.852	0.670	0.763	0.946	0.885	0.920	0.772	0.651	0.807
2	0.832	0.654	0.739	0.945	0.884	0.907	0.746	0.594	0.788
1	0.792	0.635	0.713	0.927	0.861	0.876	0.692	0.534	0.754
0 (Embedding Layer)	0.595	0.483	0.596	0.714	0.449	0.568	0.478	0.344	0.528

FIG. 10

11/11

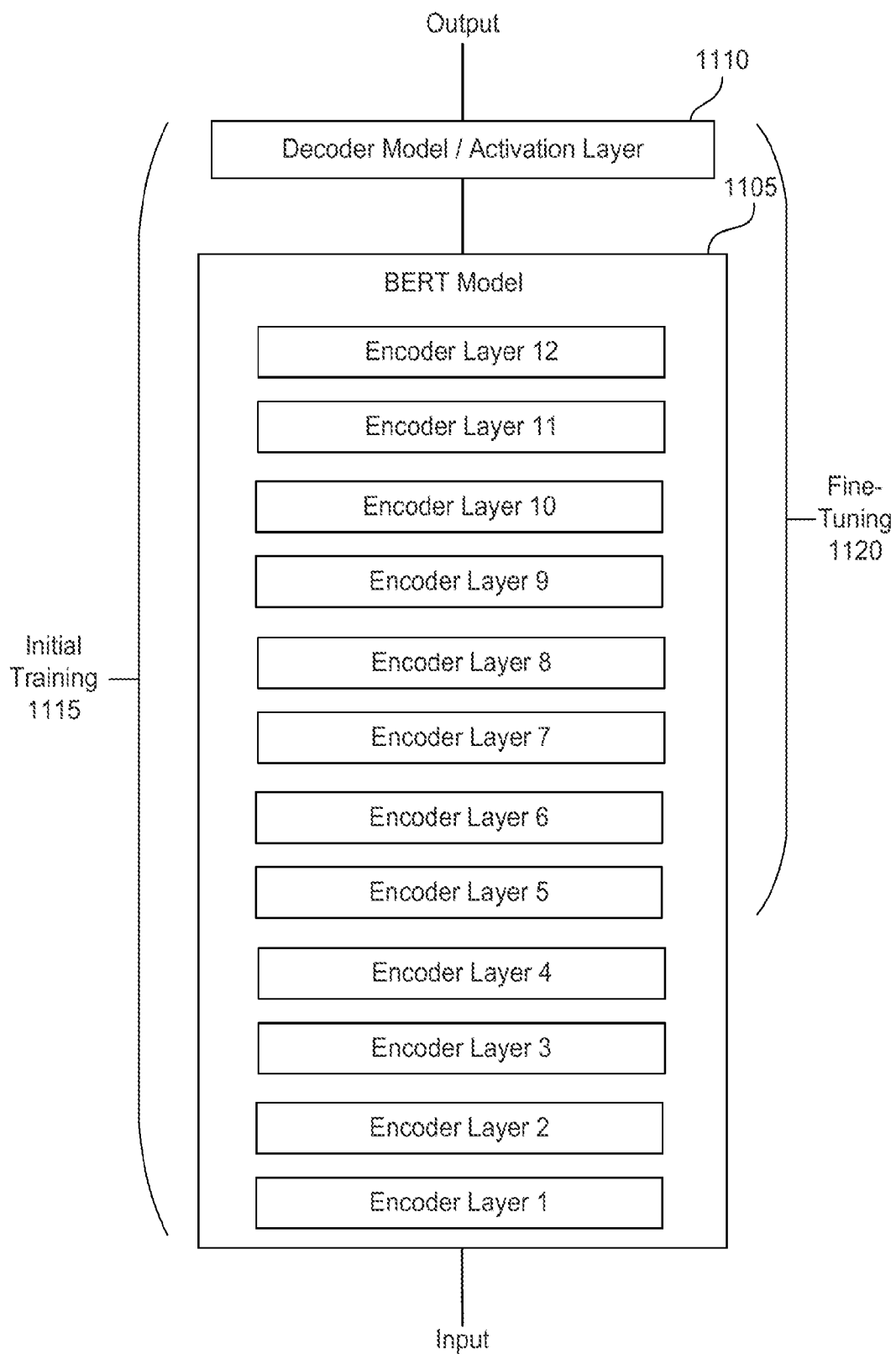


FIG. 11

INTERNATIONAL SEARCH REPORT

International application No.

PCT/US2022/040530

A. CLASSIFICATION OF SUBJECT MATTER

G06N 3/08(2006.01)i; G06N 20/00(2019.01)i

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06N 3/08(2006.01); G06N 20/00(2019.01); G06N 3/04(2006.01); G06Q 10/00(2006.01)

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

Korean utility models and applications for utility models

Japanese utility models and applications for utility models

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

eKOMPASS(KIPO internal) & Keywords: machine learning mode, fine-tuning, layer-specific, frozen layer, multi-head attention, language translation

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	YUNHUI GUO et al., 'SpotTune: Transfer Learning Through Adaptive Fine-Tuning', 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 09 January 2020 pages 4800-4807; and figures 1-2	1-20
Y	US 11138392 B2 (GOOGLE LLC) 05 October 2021 (2021-10-05) column 3, lines 56-59; column 4, lines 37-40; and claim 1	1-20
A	US 2020-0042864 A1 (VERITONE, INC.) 06 February 2020 (2020-02-06) paragraphs [0030]-[0032]; and figure 4	1-20
A	US 2020-0184327 A1 (MICROSOFT TECHNOLOGY LICENSING, LLC) 11 June 2020 (2020-06-11) paragraphs [0030]-[0036]; and figures 2-3	1-20
A	US 2021-0279596 A1 (HITACHI, LTD.) 09 September 2021 (2021-09-09) claim 1	1-20

☐ Further documents are listed in the continuation of Box C.☒ See patent family annex.

* Special categories of cited documents:

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

"D" document cited by the applicant in the international application

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

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

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

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

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

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

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

"&" document member of the same patent family

Date of the actual completion of the international search

30 November 2022

Date of mailing of the international search report

01 December 2022

Name and mailing address of the ISA/KR

Korean Intellectual Property Office
189 Cheongsa-ro, Seo-gu, Daejeon
35208, Republic of Korea

Facsimile No. +82-42-481-8578

Authorized officer

YANG, JEONG ROK

Telephone No. +82-42-481-5709

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.

PCT/US2022/040530

Patent document cited in search report			Publication date (day/month/year)	Patent family member(s)			Publication date (day/month/year)
US	11138392	B2	05 October 2021	US	2020-0034436	A1	30 January 2020
				US	2022-0083746	A1	17 March 2022
US	2020-0042864	A1	06 February 2020	EP	3652683	A1	20 May 2020
				US	11017780	B2	25 May 2021
				US	11043209	B2	22 June 2021
				US	11048982	B2	29 June 2021
				US	11176947	B2	16 November 2021
				US	2019-0115028	A1	18 April 2019
				US	2019-0139551	A1	09 May 2019
				US	2019-0385610	A1	19 December 2019
				US	2020-0042825	A1	06 February 2020
				US	2020-0043474	A1	06 February 2020
				US	2020-0043475	A1	06 February 2020
				US	2020-0058307	A1	20 February 2020
				US	2020-0066278	A1	27 February 2020
				US	2020-0075019	A1	05 March 2020
				US	2020-0285449	A1	10 September 2020
				US	2020-0285910	A1	10 September 2020
				US	2020-0286485	A1	10 September 2020
				WO	2019-028282	A1	07 February 2019
				WO	2020-028760	A1	06 February 2020
				WO	2020-028762	A1	06 February 2020
				WO	2020-068864	A1	02 April 2020
				WO	2020-068864	A9	28 May 2020
				WO	2020-068868	A1	02 April 2020
				WO	2020-176813	A1	03 September 2020
US	2020-0184327	A1	11 June 2020	AU	2019-394750	A1	11 June 2020
				BR	112021009042	A2	10 August 2021
				CA	3119027	A1	11 June 2020
				CN	113168559	A	23 July 2021
				EP	3891662	A1	13 October 2021
				JP	2022-510782	A	28 January 2022
				KR	10-2021-0093931	A	28 July 2021
				SG	11202105300	A	29 June 2021
				WO	2020-117411	A1	11 June 2020
US	2021-0279596	A1	09 September 2021	None			