US009972307B2

(12) **United States Patent**
Stent et al.

(10) **Patent No.: US 9,972,307 B2**
(45) **Date of Patent: *May 15, 2018**

(54) **SYSTEM AND METHOD FOR DIALOG MODELING**

(71) Applicant: **AT&T Intellectual Property I, L.P.**, Atlanta, GA (US)

(72) Inventors: **Amanda Stent**, Chatham, NJ (US); **Srinivas Bangalore**, Morristown, NJ (US)

(73) Assignee: **AT&T Intellectual Property I, L.P.**, Atlanta, GA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days. days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **14/845,634**

(22) Filed: **Sep. 4, 2015**

(65) **Prior Publication Data**

US 2015/0379984 A1 Dec. 31, 2015

Related U.S. Application Data

(63) Continuation of application No. 12/324,340, filed on Nov. 26, 2008, now Pat. No. 9,129,601.

(51) **Int. Cl.**
G06F 17/22 (2006.01)
G10L 15/08 (2006.01)
(Continued)

(52) **U.S. Cl.**
CPC ***G10L 15/063*** (2013.01); ***G06F 17/22*** (2013.01); ***G06F 17/2241*** (2013.01);
(Continued)

(58) **Field of Classification Search**
CPC G10L 15/22; G10L 15/18; G10L 15/1822
(Continued)

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,974,191 A * 11/1990 Amirghodsi G06F 17/277 380/1
5,748,841 A * 5/1998 Morin G06F 3/16 704/257

(Continued)

OTHER PUBLICATIONS

Bangalore, Srinivas, and Amanda J. Stent. "Incremental parsing models for dialog task structure." Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics. Association for Computational Linguistics, 2009.*

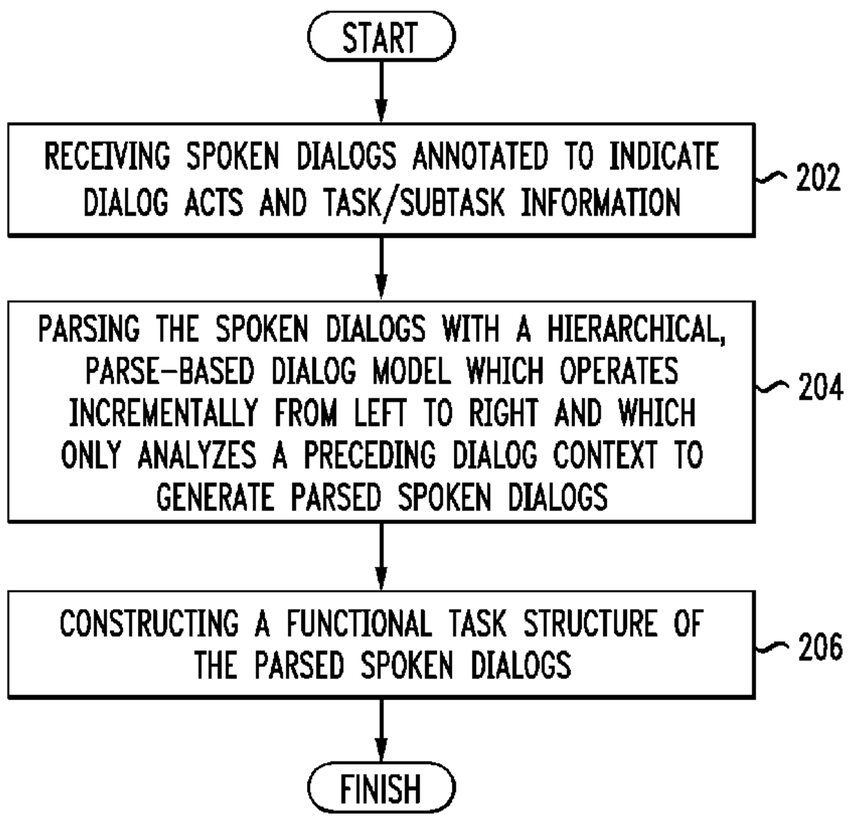
(Continued)

Primary Examiner — Jialong He

(57) **ABSTRACT**

Disclosed herein are systems, computer-implemented methods, and computer-readable media for dialog modeling. The method includes receiving spoken dialogs annotated to indicate dialog acts and task/subtask information, parsing the spoken dialogs with a hierarchical, parse-based dialog model which operates incrementally from left to right and which only analyzes a preceding dialog context to generate parsed spoken dialogs, and constructing a functional task structure of the parsed spoken dialogs. The method can further either interpret user utterances with the functional task structure of the parsed spoken dialogs or plan system responses to user utterances with the functional task structure of the parsed spoken dialogs. The parse-based dialog model can be a shift-reduce model, a start-complete model, or a connection path model.

5 Claims, 7 Drawing Sheets



(51) **Int. Cl.**

G10L 15/06	(2013.01)
G10L 15/183	(2013.01)
G10L 15/22	(2006.01)
G10L 15/00	(2013.01)
G10L 15/04	(2013.01)
G10L 25/12	(2013.01)
G10L 15/18	(2013.01)

(52) **U.S. Cl.**

CPC ***G10L 15/005*** (2013.01); ***G10L 15/04*** (2013.01); ***G10L 15/08*** (2013.01); ***G10L 15/183*** (2013.01); ***G10L 15/22*** (2013.01); ***G10L 25/12*** (2013.01); *G06F 17/227* (2013.01); *G10L 15/18* (2013.01); *G10L 2015/0638* (2013.01)

(58) **Field of Classification Search**

USPC 704/1–10, 204, 231–236, 250–257, 704/270–275, E15.04–E15.046, 704/E11.001–E11.007; 379/52, 379/67.1–88.28; 706/12–14, 45–61; 707/600–606, 790–812

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,799,268	A *	8/1998	Boguraev	G06F 17/2775 704/10
5,999,904	A *	12/1999	Brown	G10L 15/22 704/200
6,035,297	A *	3/2000	Van Huben	G06F 17/30557 707/695
6,223,150	B1 *	4/2001	Duan	G10L 15/26 704/277
6,266,642	B1 *	7/2001	Franz	G06F 17/271 704/257
6,442,524	B1 *	8/2002	Ecker	G06F 17/2755 704/257
6,510,411	B1 *	1/2003	Norton	G10L 15/26 379/76
6,601,055	B1 *	7/2003	Roberts	G06F 19/345 600/300
6,868,383	B1 *	3/2005	Bangalore	G06K 9/00355 704/251
7,031,908	B1 *	4/2006	Huang	G10L 15/1815 704/9
7,283,958	B2 *	10/2007	Azara	G06F 17/279 704/257
7,373,300	B1 *	5/2008	Bangalore	G06F 17/27 704/257
7,519,529	B1 *	4/2009	Horvitz	G06F 17/279 369/30.04
7,711,551	B2 *	5/2010	Lopez-Barquilla	G10L 15/18 704/270.1
2002/0040292	A1 *	4/2002	Marcu	G06F 17/271 704/4
2003/0061029	A1 *	3/2003	Shaket	G06F 17/279 704/9
2003/0191625	A1 *	10/2003	Gorin	G06F 17/278 704/1
2003/0216905	A1 *	11/2003	Chelba	G06F 17/271 704/9
2003/0233223	A1 *	12/2003	Wang	G06F 17/30896 704/4
2003/0233230	A1 *	12/2003	Ammicht	G10L 15/183 704/235
2004/0049391	A1 *	3/2004	Polanyi	G10L 15/1807 704/271
2005/0108001	A1 *	5/2005	Aarskog	G06F 17/271 704/10
2005/0154591	A1 *	7/2005	Lecoeuche	H04M 1/72561 704/270.1
2005/0171926	A1 *	8/2005	Thione	G10L 15/22
2005/0182618	A1 *	8/2005	Azara	G06F 17/279 704/9
2006/0009974	A1 *	1/2006	Junqua	G10L 15/083 704/257
2006/0074670	A1 *	4/2006	Weng	G10L 15/22 704/257
2006/0206332	A1 *	9/2006	Paek	G10L 15/063 704/257
2006/0271364	A1 *	11/2006	Mirkovic	G06F 17/28 704/239
2007/0050191	A1 *	3/2007	Weider	G06F 17/30864 704/275
2008/0134058	A1 *	6/2008	Shen	G10L 15/22 715/760
2008/0221902	A1 *	9/2008	Cerra	G10L 15/065 704/270.1
2010/0076765	A1 *	3/2010	Zweig	G10L 15/1822 704/255
2010/0131260	A1 *	5/2010	Bangalore	G06F 17/279 704/3
2010/0131274	A1 *	5/2010	Stent	G10L 15/183 704/257

OTHER PUBLICATIONS

Sagae, Kenji, and Alon Lavie. "A best-first probabilistic shift-reduce parser." Proceedings of the COLING/ACL on Main conference poster sessions. Association for Computational Linguistics, 2006.*

Ratnaparkhi, Adwait. "A linear observed time statistical parser based on maximum entropy models." arXiv preprint cmp-lg/9706014 (1997).*

Costa, Fabrizio, et al. "Wide coverage incremental parsing by learning attachment preferences." AI* IA 2001: Advances in Artificial Intelligence (2001): 297-307.*

* cited by examiner

FIG. 1

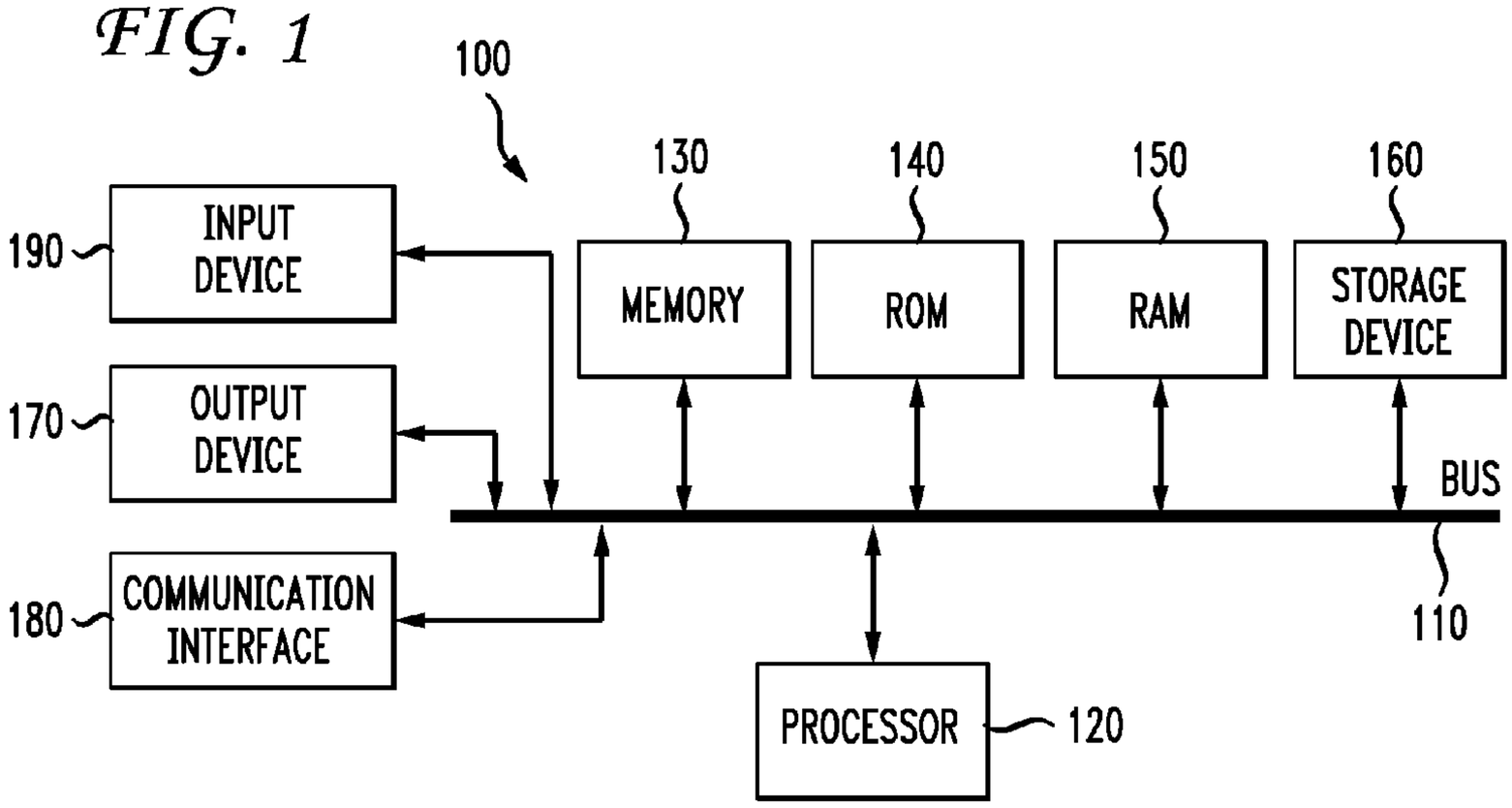

100
130
140
150
160
190
INPUT DEVICE
MEMORY
ROM
RAM
STORAGE DEVICE
170
OUTPUT DEVICE
BUS
180
COMMUNICATION INTERFACE
110
PROCESSOR
120

FIG. 2

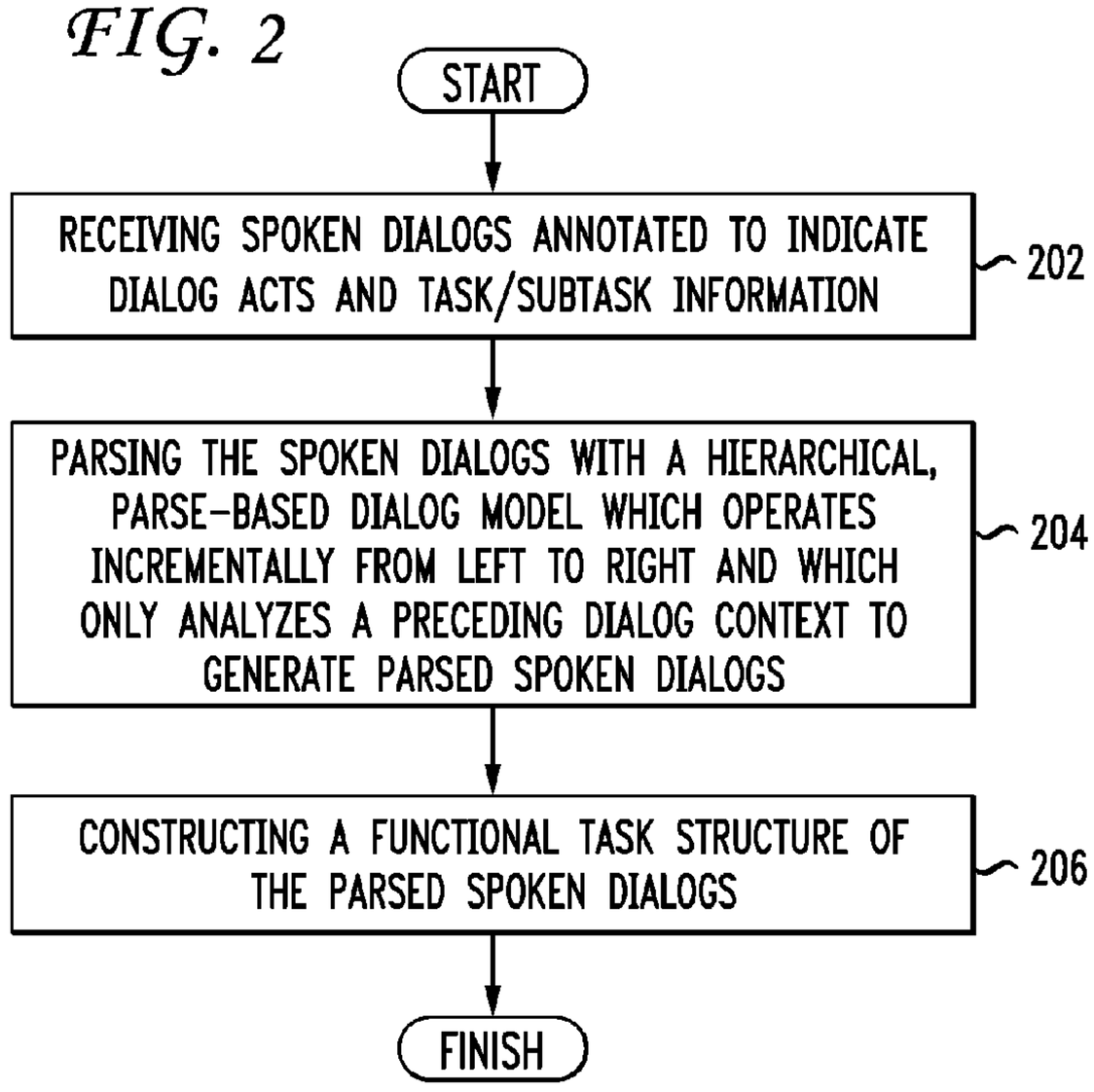
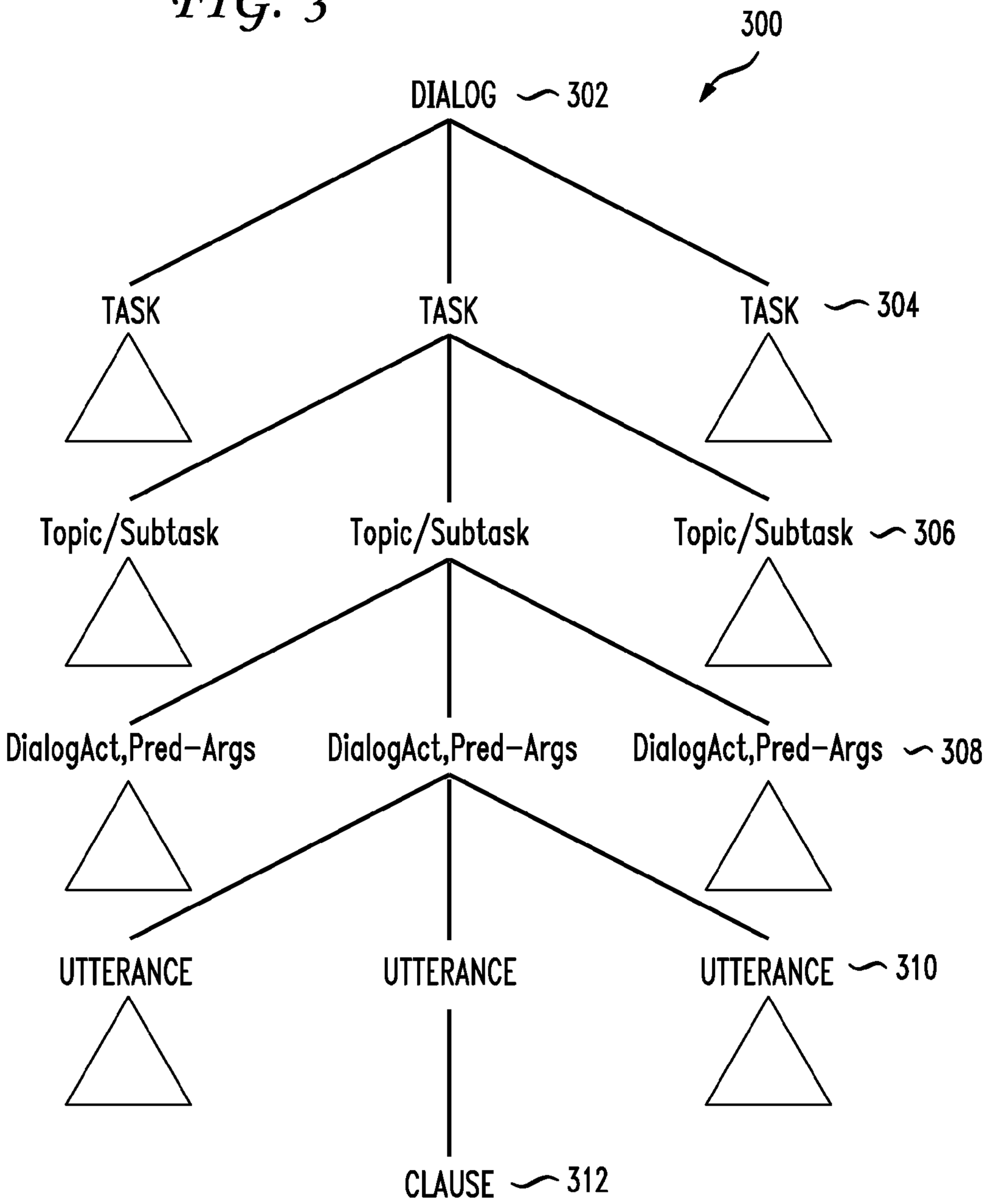

START
RECEIVING SPOKEN DIALOGS ANNOTATED TO INDICATE DIALOG ACTS AND TASK/SUBTASK INFORMATION
202
PARSING THE SPOKEN DIALOGS WITH A HIERARCHICAL, PARSE-BASED DIALOG MODEL WHICH OPERATES INCREMENTALLY FROM LEFT TO RIGHT AND WHICH ONLY ANALYZES A PRECEDING DIALOG CONTEXT TO GENERATE PARSED SPOKEN DIALOGS
204
CONSTRUCTING A FUNCTIONAL TASK STRUCTURE OF THE PARSED SPOKEN DIALOGS
206
FINISH

FIG. 3


300
DIALOG
302
TASK
TASK
TASK
304
Topic/Subtask
Topic/Subtask
Topic/Subtask
306
DialogAct,Pred-Args
DialogAct,Pred-Args
DialogAct,Pred-Args
308
UTTERANCE
UTTERANCE
UTTERANCE
310
CLAUSE
312

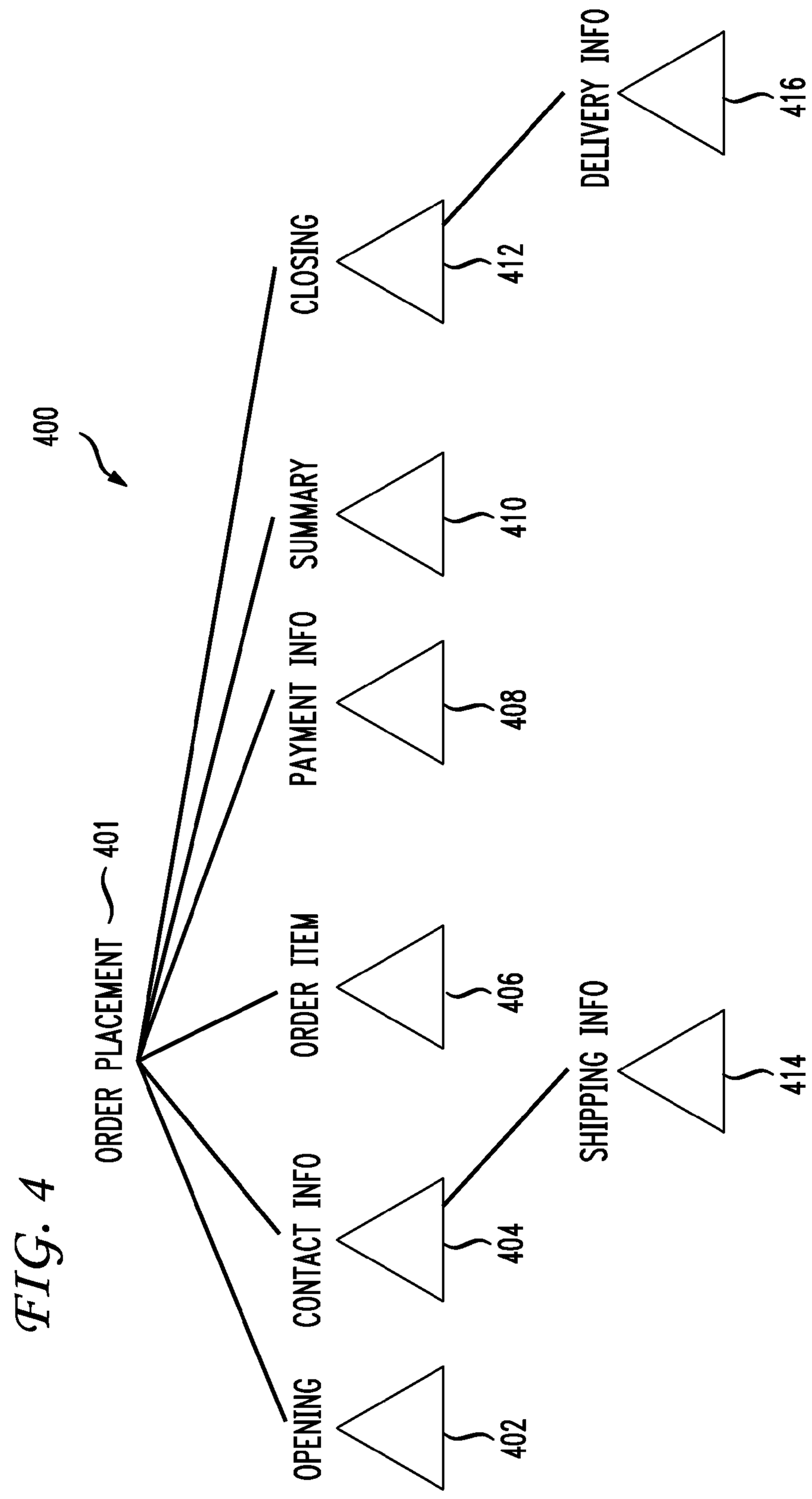

400
ORDER PLACEMENT
401
OPENING
402
CONTACT INFO
404
ORDER ITEM
406
PAYMENT INFO
408
SUMMARY
410
SHIPPING INFO
414
CLOSING
412
DELIVERY INFO
416

FIG. 4

FIG. 5A

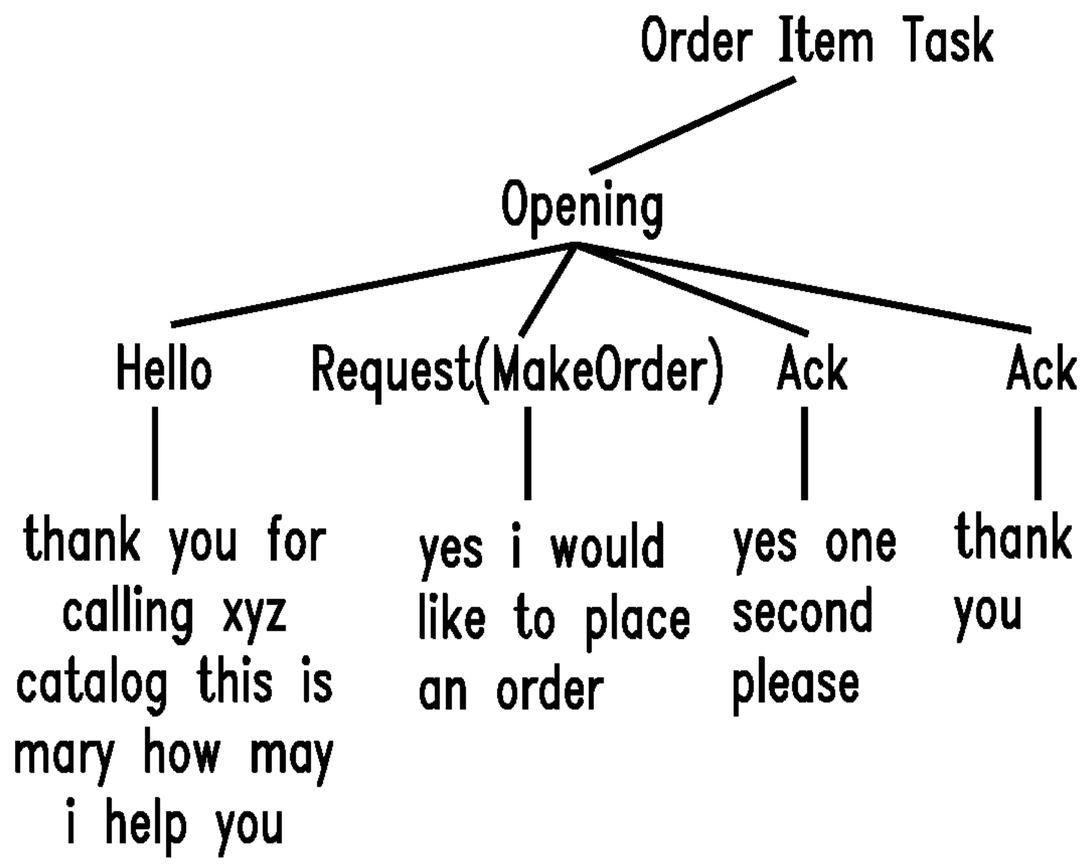

Order Item Task
Opening
Hello
Request(MakeOrder)
Ack
Ack
thank you for calling xyz catalog this is mary how may i help you
yes i would like to place an order
yes one second please
thank you

FIG. 5B

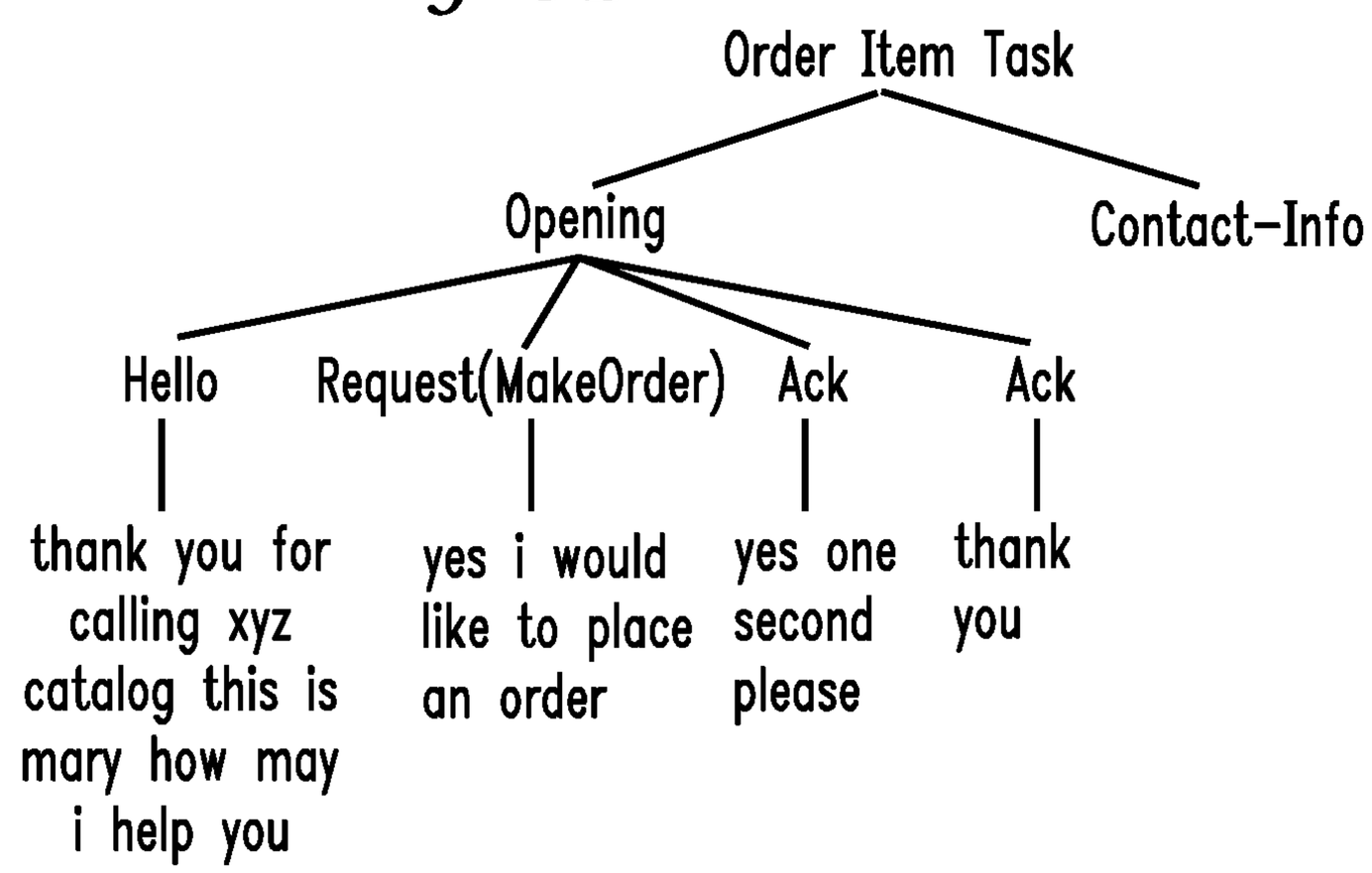

Order Item Task
Opening
Contact-Info
Hello
Request(MakeOrder)
Ack
Ack
thank you for calling xyz catalog this is mary how may i help you
yes i would like to place an order
yes one second please
thank you

FIG. 5C

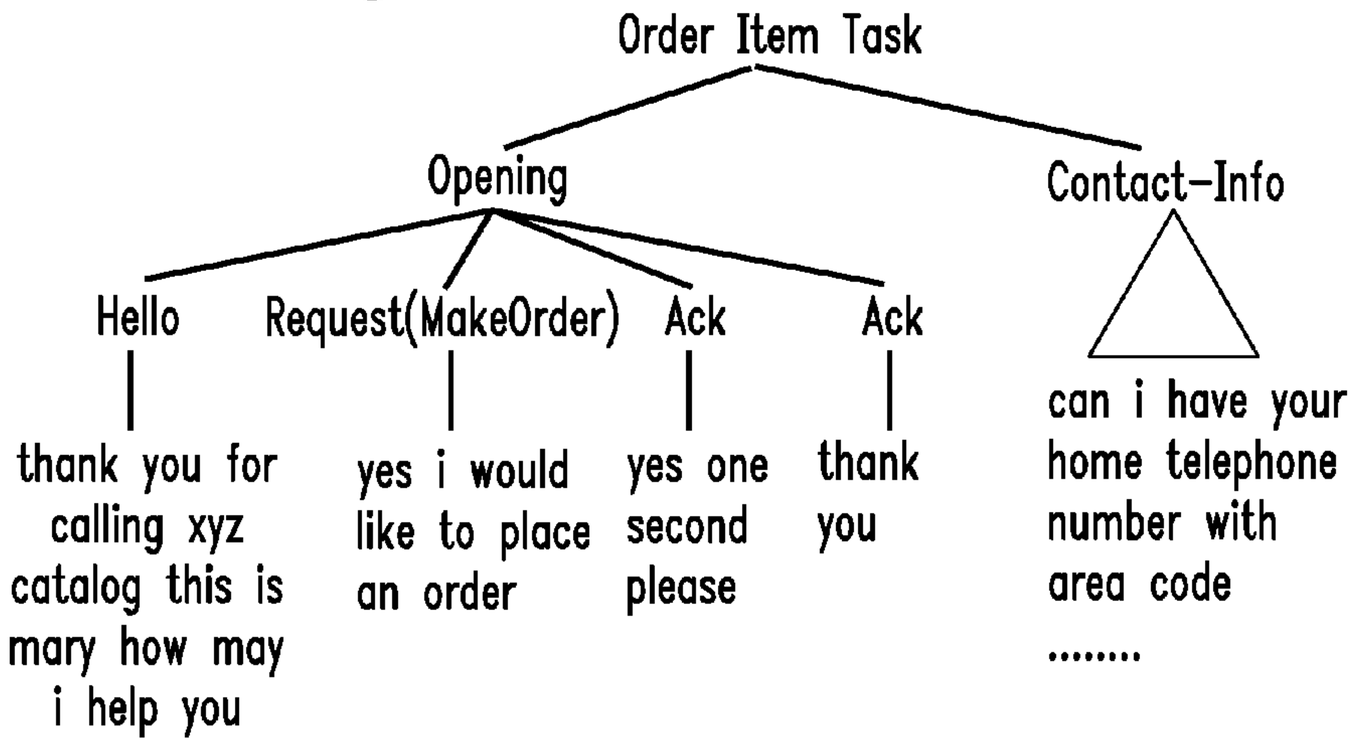

Order Item Task
Opening
Hello
Request(MakeOrder)
Ack
Ack
thank you for calling xyz catalog this is mary how may i help you
yes i would like to place an order
yes one second please
thank you
Contact-Info
can i have your home telephone number with area code
.....

FIG. 5D

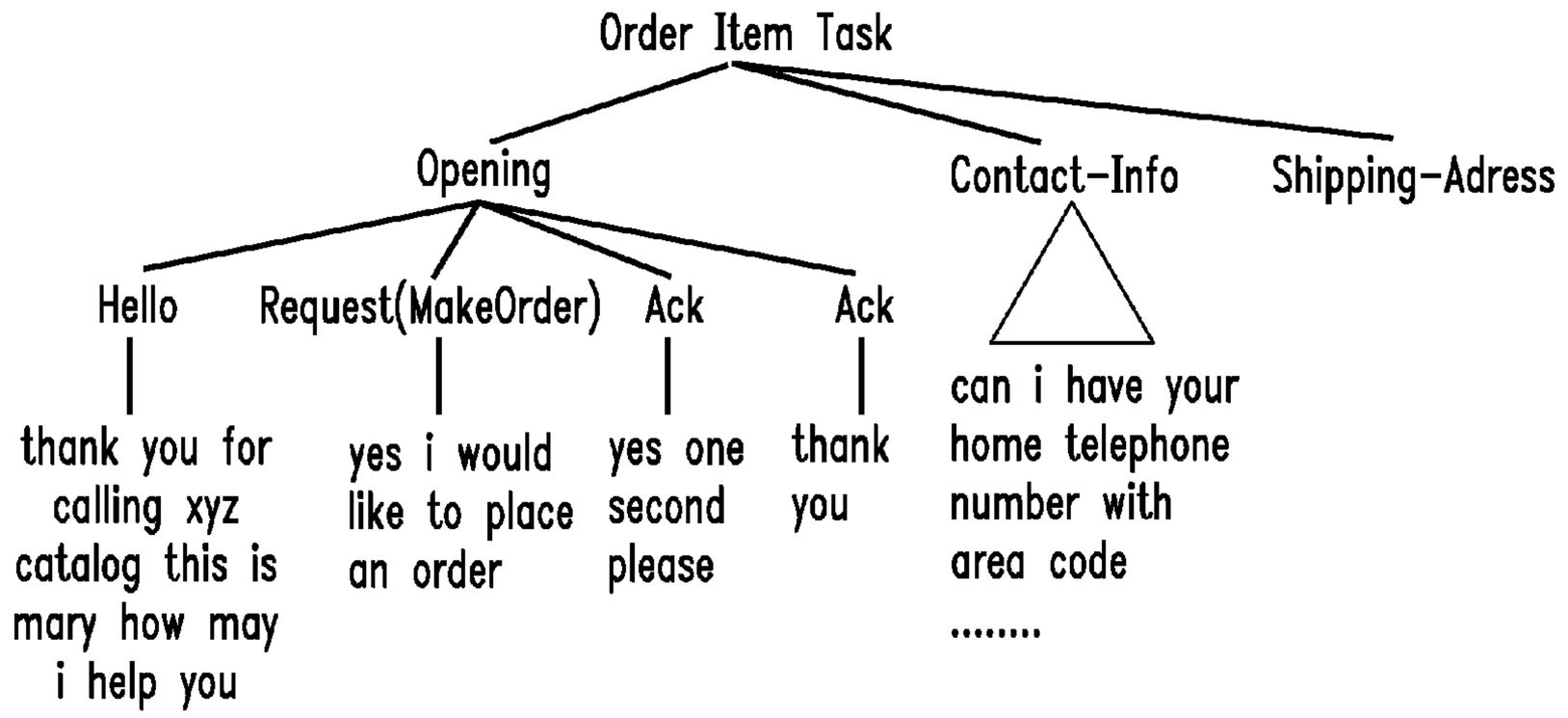
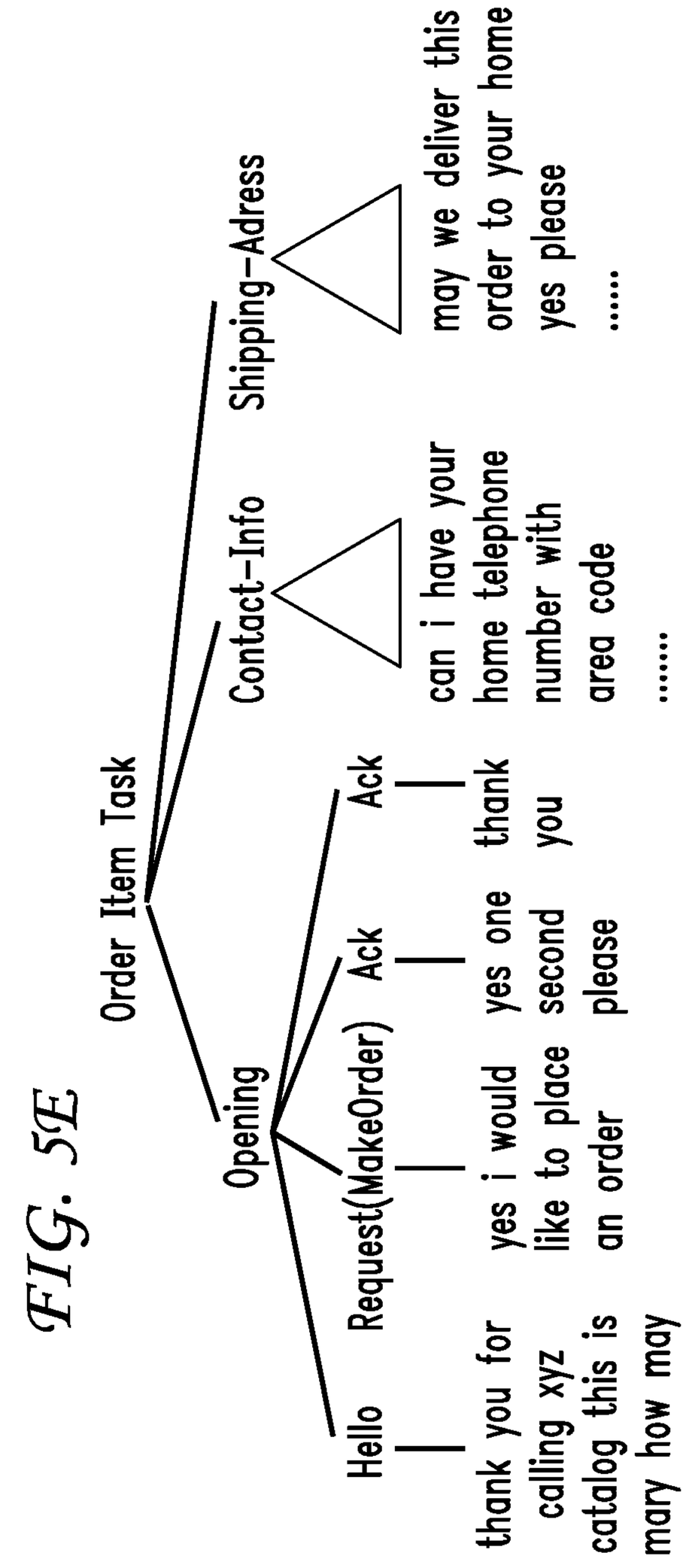

Order Item Task
Opening
Hello
Request(MakeOrder)
Ack
Ack
thank you for calling xyz catalog this is mary how may i help you
yes i would like to place an order
yes one second please
thank you
Contact-Info
Shipping-Adress
can i have your home telephone number with area code
.....

FIG. 5E


Order Item Task
Opening
Hello
thank you for calling xyz catalog this is mary how may i help you
Request(MakeOrder)
Ack
yes i would like to place an order
Ack
yes one second please
thank you
Contact-Info
can i have your home telephone number with area code
.....
Shipping-Adress
may we deliver this order to your home
yes please
.....

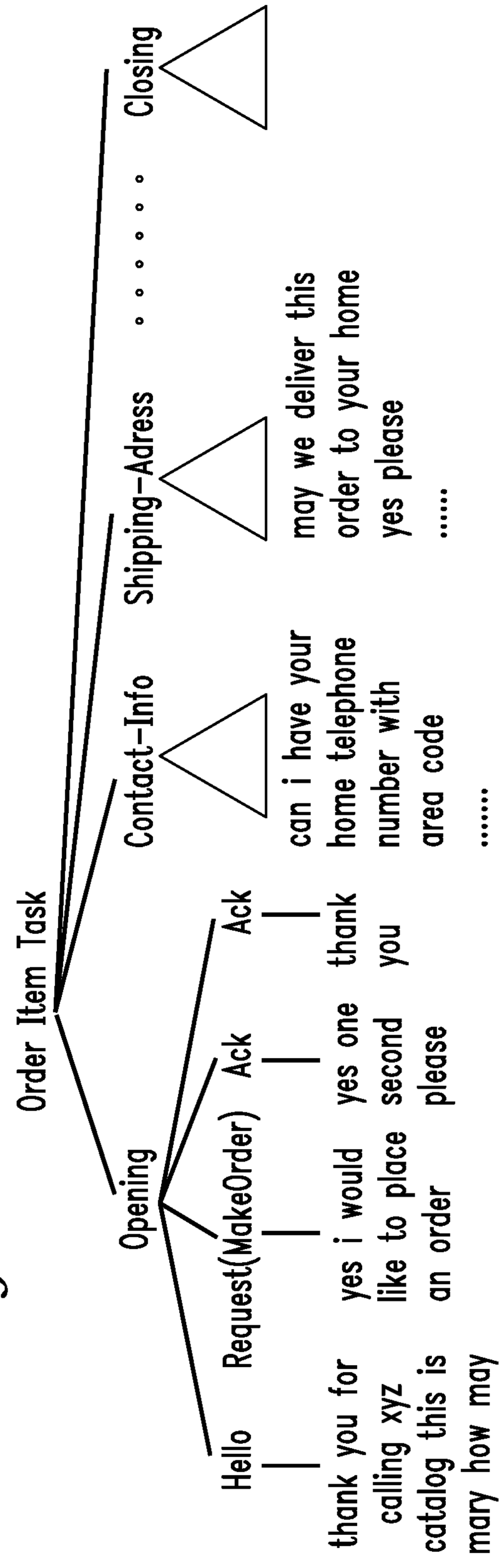

Order Item Task
Opening
Hello
thank you for calling xyz catalog this is mary how may i help you
Request(MakeOrder)
yes i would like to place an order
Ack
yes one second please
Contact-Info
can i have your home telephone number with area code
.....
Ack
thank you
Shipping-Adress
may we deliver this order to your home
yes please
.....
Closing

FIG. 5F

SYSTEM AND METHOD FOR DIALOG MODELING

PRIORITY INFORMATION

The present application is a continuation of U.S. patent application Ser. No. 12/324,340, filed Nov. 26, 2008, the contents of which is incorporated herein by reference in its entirety.

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to recognizing speech and more specifically to dialog modeling.

2. Introduction

Dialog modeling is a way to represent a dialog in an understandable, usable structure. Considerable work in the art has been devoted to discourse parsing, including dialog parsing. Much of this work has been focused on offline parsing of the rhetorical and/or semantic structure of a text monologue. Dialog models can be built online for dialog management, or offline for dialog mining and summarization. Corpora of spoken dialog are now widely and increasingly available, and frequently come with annotations for tasks (such as games), dialog acts, named entities, and elements of syntactic structure. These types of information can provide rich clues for building dialog models which current approaches to dialog modeling frequently underutilize or ignore.

Accordingly, what is needed in the art is an improved way to model dialog structure using annotations so that the model can be used to understand and reason about the dialog.

SUMMARY

Additional features and advantages of the invention will be set forth in the description which follows, and in part will be obvious from the description, or may be learned by practice of the invention. The features and advantages of the invention may be realized and obtained by means of the instruments and combinations particularly pointed out in the appended claims. These and other features of the present invention will become more fully apparent from the following description and appended claims, or may be learned by the practice of the invention as set forth herein.

Disclosed are systems, computer-implemented methods, and tangible computer-readable media for dialog modeling. The method includes receiving spoken dialogs annotated to indicate dialog acts and task/subtask information, parsing the spoken dialogs with a hierarchical, parse-based dialog model which operates incrementally from left to right and which only analyzes the preceding dialog context to generate dialog parses, and constructing a functional task structure of the parsed spoken dialogs. The method can further: interpret user utterances within the functional task structure of the parsed spoken dialogs; plan system responses to user utterances using the functional task structure of the parsed spoken dialogs; and predict aspects of the next user utterances using the functional task structure of the parsed spoken dialogs. The parse-based dialog model can be a shift-reduce model having a stack and a tree which (a) shifts each utterance onto the stack, (b) inspects the stack, and (c) based on the stack inspection, performs a reduce action that creates subtrees in a binary version of the tree. The parse-based dialog model can be a start-complete model which uses a stack to maintain a global parse state and produces a dialog task structure directly without producing an equivalent binary tree. The parse-based dialog model can be a connection path model which does not use a stack to maintain a global parse state, and which (a) directly predicts a connection path from a root to a terminal for each received spoken dialog and (b) creates a parse tree representing the connection path for each received spoken dialog.

BRIEF DESCRIPTION OF THE DRAWINGS

In order to describe the manner in which the above-recited and other advantages and features of the invention can be obtained, a more particular description of the invention briefly described above will be rendered by reference to specific embodiments thereof which are illustrated in the appended drawings. Understanding that these drawings depict only exemplary embodiments of the invention and are not therefore to be considered to be limiting of its scope, the invention will be described and explained with additional specificity and detail through the use of the accompanying drawings in which:

FIG. **1** illustrates an example system embodiment;

FIG. **2** illustrates an example method embodiment;

FIG. **3** illustrates a shared plan tree for a dialog;

FIG. **4** illustrates a sample output from a parse-based model for dialogs in a catalog ordering domain; and

FIGS. **5**A-**5**F illustrate steps in an incremental evolution of a dialog structure.

DETAILED DESCRIPTION

Various embodiments of the invention are discussed in detail below. While specific implementations are discussed, it should be understood that this is done for illustration purposes only. A person skilled in the relevant art will recognize that other components and configurations may be used without parting from the spirit and scope of the invention.

With reference to FIG. **1**, an exemplary system includes a general-purpose computing device **100**, including a processing unit (CPU) **120** and a system bus **110** that couples various system components including the system memory such as read only memory (ROM) **140** and random access memory (RAM) **150** to the processing unit **120**. Other system memory **130** may be available for use as well. It can be appreciated that the invention may operate on a computing device with more than one CPU **120** or on a group or cluster of computing devices networked together to provide greater processing capability. A processing unit **120** can include a general purpose CPU controlled by software as well as a special-purpose processor. An Intel Xeon LV L7345 processor is an example of a general purpose CPU which is controlled by software. Particular functionality may also be built into the design of a separate computer chip. An STMicroelectronics STA013 processor is an example of a special-purpose processor which decodes MP3 audio files. Of course, a processing unit includes any general purpose CPU and a module configured to control the CPU as well as a special-purpose processor where software is effectively incorporated into the actual processor design. A processing unit may essentially be a completely self-contained computing system, containing multiple cores or CPUs, a bus, memory controller, cache, etc. A multi-core processing unit may be symmetric or asymmetric.

The system bus **110** may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus

architectures. A basic input/output (BIOS) stored in ROM **140** or the like, may provide the basic routine that helps to transfer information between elements within the computing device **100**, such as during start-up. The computing device **100** further includes storage devices such as a hard disk drive **160**, a magnetic disk drive, an optical disk drive, tape drive or the like. The storage device **160** is connected to the system bus **110** by a drive interface. The drives and the associated computer readable media provide nonvolatile storage of computer readable instructions, data structures, program modules and other data for the computing device **100**. In one aspect, a hardware module that performs a particular function includes the software component stored in a tangible computer-readable medium in connection with the necessary hardware components, such as the CPU, bus, display, and so forth, to carry out the function. The basic components are known to those of skill in the art and appropriate variations are contemplated depending on the type of device, such as whether the device is a small, handheld computing device, a desktop computer, or a computer server.

Although the exemplary environment described herein employs the hard disk, it should be appreciated by those skilled in the art that other types of computer readable media which can store data that are accessible by a computer, such as magnetic cassettes, flash memory cards, digital versatile disks, cartridges, random access memories (RAMs), read only memory (ROM), a cable or wireless signal containing a bit stream and the like, may also be used in the exemplary operating environment.

To enable user interaction with the computing device **100**, an input device **190** represents any number of input mechanisms, such as a microphone for speech, a touch-sensitive screen for gesture or graphical input, keyboard, mouse, motion input, speech and so forth. The input may be used by the presenter to indicate the beginning of a speech search query. The device output **170** can also be one or more of a number of output mechanisms known to those of skill in the art. In some instances, multimodal systems enable a user to provide multiple types of input to communicate with the computing device **100**. The communications interface **180** generally governs and manages the user input and system output. There is no restriction on the invention operating on any particular hardware arrangement and therefore the basic features here may easily be substituted for improved hardware or firmware arrangements as they are developed.

For clarity of explanation, the illustrative system embodiment is presented as comprising individual functional blocks (including functional blocks labeled as a "processor"). The functions these blocks represent may be provided through the use of either shared or dedicated hardware, including, but not limited to, hardware capable of executing software and hardware, such as a processor, that is purpose-built to operate as an equivalent to software executing on a general purpose processor. For example the functions of one or more processors presented in FIG. **1** may be provided by a single shared processor or multiple processors. (Use of the term "processor" should not be construed to refer exclusively to hardware capable of executing software.) Illustrative embodiments may comprise microprocessor and/or digital signal processor (DSP) hardware, read-only memory (ROM) for storing software performing the operations discussed below, and random access memory (RAM) for storing results. Very large scale integration (VLSI) hardware embodiments, as well as custom VLSI circuitry in combination with a general purpose DSP circuit, may also be provided.

The logical operations of the various embodiments are implemented as: (1) a sequence of computer implemented steps, operations, or procedures running on a programmable circuit within a general use computer, (2) a sequence of computer implemented steps, operations, or procedures running on a specific-use programmable circuit; and/or (3) interconnected machine modules or program engines within the programmable circuits.

Having disclosed some fundamental system components, the disclosure turns to the example method embodiment as shown in FIG. **2**. For the sake of simplicity, the example method embodiment is discussed in terms of a system configured to practice the method of dialog modeling. This approach is unique in that it builds the task structure of the dialog (rather than the rhetorical structure), it is incremental (so it can be used for dialog management), and it can be applied to interpreting user utterances as well as planning system responses. The system is intended to use parsing directly for dialog management in addition to for dialog mining and information extraction. The system receives spoken dialogs annotated to indicate dialog acts and task/subtask information (**202**). Spoken dialogs can be annotated manually and/or automatically.

The system parses the spoken dialogs with a hierarchical, parse-based dialog model which operates incrementally from left to right and which only analyzes the preceding dialog context to generate dialog parses (**204**). The system constructs a functional task structure of the parsed spoken dialogs (**206**). The system uses a functional task structure rather than a rhetorical structure.

The functional task structure can serve at least three potential purposes. First, the system can interpret user utterances within the functional task structure of the parsed spoken dialogs. Second, the system can plan system responses to user utterances using the functional task structure of the parsed spoken dialogs. Third, the system can predict the type of the next user utterance using the functional task structure of the parsed spoken dialogs.

The parse-based dialog model can be one of three variations, a shift-reduce model, a start-complete model, or a connection path model. The shift-reduce model uses a stack and a tree and (a) shifts each utterance onto the stack, (b) inspects the stack, and (c) based on the stack inspection, performs a reduce action that creates subtrees in a binary-branching version of the tree. In a shift-reduce model, the system recovers the subtask structure through a right-branching, shift-reduce parsing process. The parser shifts each utterance on to the stack. It then inspects the stack and decides whether to do one or more reduce actions that result in the creation of subtrees in the task structure. Possible parser actions include unary-reduce-X, binary-reduce-X, and shift. X represents each of the non-terminal functions (or nodes) in the tree. A non-terminal function is a function (or node) in a parse tree which is either a root or a branch in the parse tree. Unary-reduce-X pops one token off the stack and pushes the non-terminal function X on the stack. Binary-reduce-X pops two tokens off the stack and pushes the non-terminal function X on the stack. Shift pushes a token representing the utterance onto the stack. At the end of the dialog, the output is a binary branching tree corresponding to the subtask structure.

The parse-based dialog model in the system can be a start-complete model which uses a stack to maintain a global parse state and produces a dialog task structure directly without producing an equivalent binary-branching tree. In the start-complete model, the system directly encodes the local instructions to build the tree in the parser actions. The

system uses a stack to maintain the global parse state. For each terminal, the parser decides whether to join that terminal onto the existing left-hand edge of the tree, or start a new subtree. The actions for the parser include start-X, n-start-X, complete-X, u-complete-X and b-complete-X. X represents each of the non-terminals in the tree. Start-X pushes a token representing the current utterance onto the stack. N-start-X pushes non-terminal X onto the stack. Complete-X pushes a token representing the current utterance onto the stack then pops the top two tokens off the stack and pushes the non-terminal X. U-complete-X pops the top token off the stack and pushes the non-terminal X. B-complete-X pops the top two tokens off the stack and pushes the non-terminal X. This model produces a dialog task structure directly, rather than producing an equivalent binary-branching tree.

The parse-based dialog model in the system can be a connection path model which does not use a stack to maintain a global parse state, and which (a) directly predicts a connection path from a root to a terminal for each received spoken dialog and (b) creates a parse tree representing the connection path for each received spoken dialog. In contrast to the shift-reduce and the start-complete models described above, the connection path model does not use a stack to track the global state of the parse. Instead, the parser directly predicts the connection path from the root to the terminal for each utterance. The collection of connection paths for all the utterances in a dialog defines the parse tree. With this model, the parser typically makes many more types of decisions. For example, on one sample set of training data, at each parsing step the connection path model could choose from 195 decision types, whereas the shift-reduce model chose from 32 decision types and the start-complete model chose from 82 decision types.

A task-oriented dialog is the result of incremental creation of a shared plan by dialog participants. FIG. **3** illustrates a shared plan tree for a dialog. The tree **300** representing the shared plan incorporates the task structure **304**, the subtask structure **306**, dialog acts **308**, syntactic structure and lexical content of utterances **310** and clauses **312** in the dialog. A task **304** can include sequences of subtasks **306**. A subtask **306** can include sequences of dialog acts **308**. Each dialog act **308** corresponds to one clause **312** spoken by one speaker.

FIG. **4** illustrates a sample output from a parse-based model for a sample dialog in a catalog ordering domain. The output **400** is a task/subtask structure. An order placement **402** task can include typical sequences of subtasks such as opening **402**, contact information **404**, order item **406**, payment information **408**, summary **410**, closing **412**, and related-offers. The output can nest subtasks multiple levels deep, even though FIG. **4** only shows one level deeper than the typical sequences of subtasks. Shipping information **414** is a nested subtask from contact information **404**. Delivery information is a subtask from summary **410**. Depending on the dialog, nesting can occur more frequently at the leftmost or rightmost frontier of the subtask tree.

One feature of an incremental sentence parser is that it can accommodate utterances from a participant into the tree as the parser receives them. FIGS. **5**A-**5**F illustrate steps in an incremental evolution of a dialog structure. FIGS. **5**A-**5**F depict an evolving dialog structure of a dialog to order an item. The dialog can be a dialog between two humans or between a human and an interactive voice response (IVR) system. FIG. **5**A illustrates the beginning of the dialog from the salutation to the second Ack. FIG. **5**B illustrates the formation of a contact information subtask. FIG. **5**C shows text of a sample dialog turn to solicit contact information. After the subtask of gathering contact information is complete, FIG. **5**D illustrates the formation of the next subtask, shipping address. FIG. **5**E shows text of a sample dialog to solicit the shipping address. One or more additional subtasks can follow after the shipping address subtask completes. After each desired subtask is complete or the user indicates they are finished, FIG. **5**F shows the formation of the final subtask, closing. After all these incremental steps, the result is a task-based dialog structure which is trained for use in interpreting user utterances and planning system responses to user utterances.

In contrast to an incremental sentence parser, the steps taken to incorporate an utterance into the tree by a dialog parser depend on whether the utterance was produced by the system or the user. The system splits each user turn into clauses which are supertagged and labeled with named entities. The system determines the dialog act of each clause from information about the clause and the immediately preceding dialog context. The system can then incorporate the clause into the task structure.

By contrast, a dialog system starts planning a system utterance by identifying the next subtask based on the task structure to that point. Then, the system chooses the dialog act of the utterance based on the task structure. Finally, the system generates an utterance to realize its communicative intent represented as a subtask and dialog act pair, with associated named entities. The system can accomplish utterance realization in any of the numerous ways known to those of skill in the art.

A dialog system predicts the type of the next user utterance by identifying the likely next subtask based on the task structure following the system utterance. Then, the system predicts the likely dialog act of the next user utterance based on the task structure. Finally, the system uses this information to select a language model for recognizing the next user utterance.

Equation 1 illustrates the probability of an incrementally evolving dialog tree after i exchanges between an agent and a user.

$$P(T_i) = \sum_{t \in T_{i-1}} P(t) * \sum_{s^a \in S} \sum_{d^a \in D} \sum_{d^u \in D} \sum_{s^u \in S} P(s^a \mid t) * P(d^a \mid s^a, t) * P(d^u \mid c_i^u, t) * P(s^u \mid d^u, t) \quad (1)$$

A parse for a dialog can be a shallow or deep tree structure. A shallow parse, or a chunk-based dialog model, is one in which utterances are grouped together into subtasks, but the dominance relations among subtasks are not tracked. The term parse-based dialog model refers to deep parsing models for dialog. This disclosure discusses three alternative parse-based models: shift-reduce, start-complete and connection path. Each of these models operates on the subtask structure for the dialog incrementally, from left-to-right, with access only to the preceding dialog context. These three models differ in the parsing actions and data structures internal to the parser. The instructions to reconstruct the parse are either entirely encoded in the stack, as in the shift reduce model, or entirely encoded in the parsing actions, as in the start-complete and connection path models. For each parsing action required to build the parse tree, the system constructs a feature vector containing contextual information for the parsing action. The system can use these

feature vectors and the associated parser actions to train a maximum entropy model. The system then uses this model to incrementally incorporate the utterances for a new dialog into that dialog's task structure as the dialog progresses.

One example data set of spoken dialogs is the CHILD corpus of 817 telephone-based dialogs in a catalog-purchasing domain. Each dialog in the CHILD corpus was transcribed by hand and all numbers (telephone, credit card, etc.) were removed for privacy reasons. The average dialog in this data set has 60 turns. Prior to processing, an automatic process segmented the dialogs into utterances and annotated them with part-of-speech tag and supertag information, named entities, dialog acts, and tasks/subtasks. For example, for the call-level type, some task/subtask labels are call-forward, closing, misc-other, opening, out-of-domain, sub-call. For the task-level type, some task/subtask labels are check-availability, contact-info, delivery-info, discount, order-change, order-item, order-problem, payment-info, related-offer, shipping-address, special-offer, and summary. For the dialog act type "Explain", some subtypes are catalog, CC_related, discount, order_info, order_problem, payment_rel, product_info, promotions, related_offer, and shipping.

The system can use the following features for each utterance: (a) the speaker ID; (b) unigrams, bigrams, and trigrams of the words; (c) unigrams, bigrams, and trigrams of the part of speech tags; (d) unigrams, bigrams and trigrams of the supertags; (e) binary features indicating the presence or absence of particular types of named entity; (f) the dialog act determined by the parser; (g) the task/subtask label determined by the parser; and (h) the parser stack at the current utterance determined by the parser.

An input feature vector has all of these features for up to three utterances of left-hand context, plus the current utterance for interpretation. The label is the parsing action or the dialog act label. If more than one parsing action takes place on a particular utterance, such as a shift operation followed by a reduce operation, the system repeats the feature vector twice with different stack contents.

The system can train models to determine dialog acts for user utterances, determine subtask labels and parsing actions for user utterances, determine subtask labels and parsing actions for system utterances, and determine dialog acts for system utterances. The system can choose a Maximum Entropy distribution that properly estimates the average of each feature over the training data. A machine learning toolkit can encode multiclass classification problems using binary Maximum Entropy classifiers in order to increase the speed of training and to scale the approach to large data sets. Each of the V classes is encoded as a bit vector such that, in the vector for class i, the i-th bit is one and all other bits are zero. Then, V one-vs-other binary classifiers are used as follows:

$$P(y \mid \Phi) = 1 - P(\bar{y} \mid \Phi) = \frac{e^{\lambda_y^{*}\Phi}}{e^{\lambda_y^{*}\Phi} + e^{\lambda_{\bar{y}}^{*}\Phi}} = \frac{1}{1 + e^{-\lambda'_y{}^{*}\Phi}} \tag{2}$$

where $\lambda_{\bar{y}}$ is the parameter vector for the anti-label $\bar{y}$ and $\lambda'_y=\lambda_y-\lambda_{\bar{y}}$. Based on the class independence assumption, the system can require that $y_i=1$ and for all $j\neq i$, $y_j=0$:

$$P(st_i \mid \Phi) = P(y_i \mid \Phi)\prod_{j\neq 1}^{V} P(y_j \mid \Phi) \tag{3}$$

The three parsing models' decoding processes have similar schema of proceeding through the following six tasks: (1) assign-user-dialog-act assigns the dialog act for the user's current utterance; (2) assign-user-subtask predicts the parser action for the user's current utterance; (3) predict-system-subtask predicts the parser action for the system's utterance to be generated; (4) predict-system-dialog-act predicts the dialog act for the system's utterance to be generated; (5) predict-user-subtask predicts the parser action for the user's next utterance; and (6) predict-user-dialog-act predicts the dialog act for the user's next utterance. The system models each of these steps in the decoding process as a classification problem with different feature vectors and label sets (e.g. parse action labels, dialog act labels). The decoder constructs a vector of features depending on the amount of context being used. The system uses this feature vector to query the appropriate classifier model to obtain a vector of labels with weights. The system uses parser action labels to extend the parse data structure. For example, in the shift-reduce model, a shift operation results in a push action on the stack and a reduce-X operation results in popping the top two elements of the stack and pushing X on to the stack. The decoder can use n-best results from the classifier to enlarge the search space. For example, in order to manage the search space effectively, the decoder can use a beam pruning strategy. The decoding process proceeds until it reaches the end of the dialog.

Given that the classifiers are sometimes error-prone in their assignment of labels, the parsing step of the decoder should robustly respond to such errors. The system can rule our incompatible parser actions based on the state of the stack in the different models (e.g. reduce-X action when the stack has one element, shift action on an already shifted utterance). The system can also use n-best results to alleviate the impact of classification errors. Finally, if the stack contains unattached constituents at the end of the dialog, the decoder attaches them as sibling constituents to produce a rooted tree structure. These constraints contribute to robustness, but are not applicable to the connection path model because any parsing action suggested by the classifier can be incorporated into the incremental parse tree. Consequently, when the system uses a connection path model, it has fewer opportunities to correct classifier errors.

The parse structure of dialogs as described herein can be used in online and offline applications. Some example applications include speech dialog mining, monitoring products purchases, monitoring the efficiency of different dialog stages, and measuring effectiveness of different promotional coupons or offers.

In one aspect, the system can use the functional task structure of the parsed spoken dialogs to predict a next user dialog act and subtask label for a user dialog turn and/or a system dialog turn. To accomplish this, the system can incrementally receive user utterances as a dialog progresses, assign a dialog act to a current user utterance based on the functional task structure of the parsed spoken dialogs, assign a subtask label to the current user utterance based on the functional task structure of the parsed spoken dialogs, predict a system subtask label for a next system utterance based on the functional task structure of the parsed spoken dialogs, predict a system dialog act for a next system utterance based on the functional task structure of the parsed spoken dialogs, predict a next subtask label for a next user utterance based on the functional task structure of the parsed spoken dialogs, and predict a next dialog act for a next user utterance based on the functional task structure of the parsed spoken dialogs.

Embodiments within the scope of the present invention may also include computer-readable media for carrying or having computer-executable instructions or data structures

stored thereon. Such computer-readable media can be any available media that can be accessed by a general purpose or special purpose computer, including the functional design of any special purpose processor as discussed above. By way of example, and not limitation, such computer-readable media can comprise RAM, ROM, EEPROM, CD-ROM or other optical disk storage, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to carry or store desired program code means in the form of computer-executable instructions, data structures, or processor chip design. When information is transferred or provided over a network or another communications connection (either hardwired, wireless, or combination thereof) to a computer, the computer properly views the connection as a computer-readable medium. Thus, any such connection is properly termed a computer-readable medium. Combinations of the above should also be included within the scope of the computer-readable media.

Computer-executable instructions include, for example, instructions and data which cause a general purpose computer, special purpose computer, or special purpose processing device to perform a certain function or group of functions. Computer-executable instructions also include program modules that are executed by computers in stand-alone or network environments. Generally, program modules include routines, programs, objects, components, data structures, and the functions inherent in the design of special-purpose processors, etc. that perform particular tasks or implement particular abstract data types. Computer-executable instructions, associated data structures, and program modules represent examples of the program code means for executing steps of the methods disclosed herein. The particular sequence of such executable instructions or associated data structures represents examples of corresponding acts for implementing the functions described in such steps.

Those of skill in the art will appreciate that other embodiments of the invention may be practiced in network computing environments with many types of computer system configurations, including personal computers, hand-held devices, multi-processor systems, microprocessor-based or programmable consumer electronics, network PCs, minicomputers, mainframe computers, and the like. Embodiments may also be practiced in distributed computing environments where tasks are performed by local and remote processing devices that are linked (either by hardwired links, wireless links, or by a combination thereof) through a communications network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

The various embodiments described above are provided by way of illustration only and should not be construed to limit the invention. For example, the principles herein are applicable to train nearly any spoken dialog system, measure dialog efficiency, and predict user or system dialog acts as well as dialog tasks. Those skilled in the art will readily recognize various modifications and changes that may be made to the present invention without following the example embodiments and applications illustrated and described herein, and without departing from the true spirit and scope of the present invention.

We claim:

1. A method comprising:

training a plurality of hierarchical, parsed-based dialog models, wherein each of the plurality of hierarchical, parsed-based dialog models operates incrementally from left to right and only analyzes an immediately preceding dialog context and wherein the plurality of hierarchical, parsed-based dialog models comprises one of a shift-reduce model, a start-complete model or a connection path model, and wherein:

when the plurality of hierarchical, parsed-based dialog models comprises a shift-reduce model, the shift-reduce model has a stack and a tree which (a) shifts each utterance onto the stack, (b) inspects the stack, and (c) based on a stack inspection, performs a reduce action that creates subtrees in the tree;

when the plurality of hierarchical, parsed-based dialog models comprises a start-complete model, the start-complete model uses a stack to maintain a global parse state and produces a dialog task structure directly without producing an equivalent tree; and

when the plurality of hierarchical, parsed-based dialog models comprises a connection path model, the connection path model does not use a stack to maintain a global parse state, and wherein the connection path model (a) directly predicts a connection path from a root to a terminal for each received spoken dialog, and (b) creates a parse tree representing the connection path for each received spoken dialog;

parsing, via a processor, spoken dialogs with a hierarchical, parse-based dialog model from the plurality of hierarchical, parsed-based dialog models, to yield parsed spoken dialogs;

constructing a functional task structure of the parsed spoken dialogs;

predicting a likely next dialog act in a spoken dialog using the functional task structure and the hierarchical, parsed-based dialog model, the likely next dialog act corresponding to a next utterance comprising a clause to be spoken by a speaker, wherein the predicting occurs prior to receiving the next utterance; and

selecting a language model for the next utterance based on the likely next dialog act.

2. The method of claim **1**, further comprising measuring a dialog efficiency at different dialog stages based on the language model selected.

3. A system comprising:

a processor; and

a computer-readable storage medium having instructions stored which, when executed by the processor, cause the processor to perform operations comprising:

training a plurality of hierarchical, parsed-based dialog models wherein the plurality of hierarchical, parsed-based dialog models operate incrementally from left to right and only analyze an immediately preceding dialog context and wherein the plurality of hierarchical, parsed-based dialog models comprises one of a shift-reduce model, a start-complete model or a connection path model, and wherein:

when the plurality of hierarchical, parsed-based dialog models comprises a shift-reduce model, the shift-reduce model has a stack and a tree which (a) shifts each utterance onto the stack, (b) inspects the stack, and (c) based on a stack inspection, performs a reduce action that creates subtrees in the tree;

when the plurality of hierarchical, parsed-based dialog models comprises a start-complete model, the start-complete model uses a stack to maintain a global parse state and produces a dialog task structure directly without producing an equivalent tree; and

when the plurality of hierarchical, parsed-based dialog models comprises a connection path model, the

connection path model does not use a stack to maintain a global parse state, and wherein the connection path model (a) directly predicts a connection path from a root to a terminal for each received spoken dialog, and (b) creates a parse tree representing the connection path for each received spoken dialog;

parsing spoken dialogs with a hierarchical, parse-based dialog model from the plurality of hierarchical, parsed-based dialog models, to yield parsed spoken dialogs;

constructing a functional task structure of the parsed spoken dialogs;

predicting a likely next dialog act in a spoken dialog using the functional task structure and the hierarchical, parsed-based dialog model, the likely next dialog act corresponding to a next utterance comprising a clause to be spoken by a speaker, wherein the predicting occurs prior to receiving the next utterance; and

selecting a language model for the next utterance based on the likely next dialog act.

4. The system of claim **3**, the computer-readable storage medium having additional instructions stored which, when executed by the processor, cause the processor to perform operations comprising measuring a dialog efficiency at different dialog stages based on the language model selected.

5. A computer-readable storage device having instructions stored which, when executed by a computing device, cause the computing device to perform operations comprising:

training a plurality of hierarchical, parsed-based dialog models wherein the plurality of hierarchical, parsed-based dialog models operate incrementally from left to right and only analyze an immediately preceding dialog context and wherein the plurality of hierarchical, parsed-based dialog models comprises one of a shift-reduce model, a start-complete model or a connection path model, and wherein:

when the plurality of hierarchical, parsed-based dialog models comprises a shift-reduce model, the shift-reduce model has a stack and a tree which (a) shifts each utterance onto the stack, (b) inspects the stack, and (c) based on a stack inspection, performs a reduce action that creates subtrees in the tree;

when the plurality of hierarchical, parsed-based dialog models comprises a start-complete model, the start-complete model uses a stack to maintain a global parse state and produces a dialog task structure directly without producing an equivalent tree; and

when the plurality of hierarchical, parsed-based dialog models comprises a connection path model, the connection path model does not use a stack to maintain a global parse state, and wherein the connection path model (a) directly predicts a connection path from a root to a terminal for each received spoken dialog, and (b) creates a parse tree representing the connection path for each received spoken dialog;

parsing spoken dialogs with a hierarchical, parse-based dialog model from the plurality of hierarchical, parsed-based dialog models, to yield parsed spoken dialogs;

constructing a functional task structure of the parsed spoken dialogs;

predicting a likely next dialog act in a spoken dialog using the functional task structure and the hierarchical, parsed-based dialog model, the likely next dialog act corresponding to a next utterance comprising a clause to be spoken by a speaker, wherein the predicting occurs prior to receiving the next utterance; and

selecting a language model for the next utterance based on the likely next dialog act.

* * * * *