Learning to Understand Questions on the Task History of a Service Robot

Vittorio Perera and Manuela Veloso
Carnegie Mellon University
Pittsburgh, PA USA
vdperera,mmv@cs.cmu.edu

***Abstract*— We present a novel approach to enable a mobile service robot to understand questions about the history of tasks it has executed. We frame the problem of understanding such questions as grounding an input sentence to a query that can be executed on the logs recorded by the robot during its runs. We define a query as an *operation* followed by a set of *filters*. In order to ground sentence to a query we introduce a joint probabilistic model. The model is composed by a shallow semantic parser and a knowledge base to store and re-use the groundings of a sentence. The Knowledge Base and its predicates are designed to match the structure of a query. Our results show that, by using such Knowledge Base, the approach proposed requires fewer and fewer corrections as users interact with the system.**

I. Introduction

We have been deploying the CoBot robots (see Figure 1) for over five yecars [1]. The CoBot robots are mobile service robots able to autonomously navigate in our multi-floor office environment, and to execute tasks for users, such as moving between locations, delivering objects, and escorting visitors. The robots have navigated over 1,000km [2] and have executed countless tasks.

The CoBot robots autonomously move performing tasks without any supervision. Hence, we can see these robots navigating in our environment, but people are unaware of what specific task they are executing or have executed. In this work, our aim is to enable the CoBot robots to autonomously answer questions about the history of the tasks they have performed. We address questions provided as natural language sentences, and aim at retrieving a specific bit of information from the long list of tasks executed by the robot. The problem we have to solve is therefore two fold. First, the robots need to be able to access the history of tasks they have performed. The CoBot robots are continuously recording logs of their execution, so we use these logs to access the history of tasks. Second, the robots need to ground a natural language sentence to an operation that can be performed on the logs recorded. To do so, we introduce parameterized operations and learn a Knowledge Base to store and to re-use the groundings of the questions in natural language to the log operations.

In previous work, we introduced the concept of verbalization [3], [4] to allow a mobile service robot to convert its route experiences into natural language. Here we move beyond the navigation aspect to the task level. We focus on learning the groundings of language to log queries, i.e., we focus on the understanding of the question itself.

Fig. 1: The CoBot robots.

We frame the problem of answering a question about the history of tasks executed as finding the best query to execute on the logs recorded by the robot during its runs. To do so, we define the structure of a query as a primitive *operation* to be performed on the record of the logs that match a set of filters defined by the user. We then enable the robot to learn the grounding of a question to a query by using a Knowledge Base that stores mappings from natural language expressions to log operations and filters. Finally, we validate our approach by evaluating it on a corpus of queries.

The rest of the paper is organized as follows. Section II discusses related work. Sections III-Aand III-B present the structure of the tasks our CoBot robots execute and the structure of the logs they record. Section III-C provides the formalization of the structure of the queries that can be executed on the logs. Section IV introduces the overall model to ground sentences to queries. Section IV-A presents our semantic parser, and Sections IV-B and IV-C present the Knowledge Base we designed and its use for grounding. Finally, Section V details our experimental setup and the results achieved.

II. Related Work

In the literature we identify three main categories of works relevant to our proposed approach: 1) works presenting robots that are persistently deployed in the environment, 2)

works pertaining to intelligibility or explanation of intelligent agents, and 3) works enabling autonomous robots to understand natural language.

The CoBot robots are not the only example of robots interacting and executing tasks for their user that are persistently deployed in the environment. The Kejia robot [5], initially designed to compete in the Robocup@Home[1], was recently deployed in a mall where it acted as a guide [6]. The DWI-Bots [7] are able to complete user requests and integrate probabilistic and symbolic reasoning. These capabilities are used to dialog with their user and resolve queries efficiently by using commonsense reasoning. Finally, the STRANDS project [8] aims at tackling the increasing demand from end-users for autonomous service robots that can operate in real environments for extended periods.

In Human-Computer Interaction a large body of works focus on intelligibility of intelligent systems (e.g., for context-aware systems [9]). In [10], the authors present a study performed over 200 participants. The results shows that automatically providing explanation for a system's decision can improve users trust, satisfaction and acceptance level. In [11], the authors use a music recommender system to show how the mental model of a user affect the ability of the user to more effectively operate the system itself. In particular, they show that completeness in the explanation provided by the system helps the user build more useful mental models. Last, [12] uses a Clinical Decision Support System (CDSS) to show how automatically generated explanations can increase the users trust. Similarly we aim at improving the understanding and trust users have toward our CoBot robots. As a first step, in this work we enable our robots to understand question on the task they perform.

We expect users to ask questions to our robots using natural language. Enabling an agent to understand natural language has been a long standing goal for AI researchers [13] that today interests more and more scholars [14]. In the robotics literature we identify two main approaches to enable robots to understand natural language. The first is to map the input sentence to a logical form that the robot can evaluate and ac upon (e.g., λ-calculus [5], [6], [15], [16]). The second approach, that we also adopt, is to consider a probabilistic model to map language into robot actions as in [17], [18], [19], [20]. In the previous works, robots learned how to understand commands or instructions requiring them to execute physical actions. We move beyond these works by enabling robots to understand questions about the history of tasks they execute. Finally, to represents the tasks executed by the robots, we use semantic frames [21], a formalism extensively applied both to linguistics [22] and to robotics [23], [24].

III. Tasks, Logs, And Queries Structure

Our goal is to enable a mobile service robot to answer questions about its past experience, and in particular inquiries about the tasks it has executed. In this sections we first describe the tasks the robots are able to execute. We then presents the structure of the logs recorded. Finally, we formalize what a query to the log is and its structure.

Action:	GoTo
Arguments:	Destination
Action:	FindAndFetch
Arguments:	Object
	Source
	Destination
Action:	Escort
Arguments:	Person
	Destination

Fig. 2: Semantic frames and their arguments for each of the tasks the robot can execute.

A. Robot Tasks

Our CoBot robots are able to execute three tasks: going to a specific place, escorting a person to a location in the environment or delivering an object. Each of this tasks is represented by a semantic frame. Semantic frames are composed by an action (the task itself) and a set of task-specific arguments. Figure 2 shows the semantic frames and their argument for the three tasks our robot can perform.

B. Logs

Our CoBot robots run primarily on ROS[2]. ROS allows to write modular code where each module is called a node. Nodes can publish or subscribe to specific topics in order to send and receive messages from other nodes. Each topic specifies the structure of the messages exchanged. Using rosbag[3] it is possible to record and replay all the messages exchanged during each run of the robot. These recordings constitute the bulk of our logs. During each run the logging process records the messages exchanged over 40 topics but, to answer queries about the history of task executed, we will focus only on one specific topic (namely, `/Cobot/TaskPlannerStatus`). Figure 3 shows the structure of the messages exchanged on this topic. The message has many different attributes that are used for multiple purposes. To keep track of the current execution state of a task we use the *timeBlocked*, and *paused* arguments. To schedule requests for other tasks we need to check the *navigationTimeReamining* and *timeToDeadline* arguments. The argument of the message relevant to answer questions about the history of tasks performed are the one specifying the task type (*currentTask*) and the ones matching the argument of the semantic frames representing the tasks (*fromLocation*, *toLocation*, *objectToFind*).

All the relevant attributes in the messages are strings. The *currentTask* can only be one of the three actions shown in Figure 2 (GoTo, FindAndFetch or Escort). The *objectToFind*

[1] http://www.robocupathome.org/

[2] www.ros.org

[3] http://wiki.ros.org/rosbag

attribute matches directly the English expressions used to schedule the task. Tasks can be scheduled from a GUI on board, from a website, or through a spoken interface; in all three cases the *objectToFind* attribute contains the information as provided by the user. Last, the two arguments relating to location (*fromLocation*, and *toLocation*) are expressed as location-labels (e.g., "GHC7412") that the robot autonomously map to (x, y) coordinates in its map.

`int32`	*mission_id*
`string`	*currentTask*
`string`	*currentSubTask*
`string`	*currentNavigationSubTask*
`float32`	*timeBlocked*
`float32`	*taskDuration*
`float32`	*subTaskDuration*
`float32`	*navigationSubTaskDuration*
`float32`	*navigationTimeRemaining*
`float32`	*timeToDeadline*
`CobotLocalizationMsg`	*currentDestination*
`string`	*fromLocation*
`string`	*toLocation*
`string`	*objectToFind*
`bool`	*carryingObject*
`bool`	*navigating*
`bool`	*taskCompleted*
`int32`	*successValue*
`bool`	*paused*
`string`	*owner*

Fig. 3: Structure of the message exchanged on the topic `/Cobot/TaskPlannerStatus`. On the left the argument type and on the right the argument name.

C. Queries

We define a query as an *operation* and a set of *filters*. The operation defines which function we apply to the relevant records of the logs. We identify three different operations:

- **CHECK:** this operation returns true if the logs have at least a record matching all the filters specified by the query, otherwise it returns false (e.g., "Did you go to Chris office last Tuesday?").
- **COUNT:** this operation returns the number of records matching the filters specified in the query (e.g., "How many times did you escort someone to the lab last week?").
- **SELECT:** this operation returns all the records matching the filters specified in the query (e.g., "what where you doing on March 3rd at 3pm?").

The filters, as the name suggest, reduce the number of records and select only the relevant ones for the query. We consider two different types of filters namely, task-related and the time-related. Since the queries we consider refer to the history of tasks executed, it is natural for the user to specify a time to frame their question. We formalize the time reference of a query as time-related filters. Specifically, we identify two different type of such filters, the ones referring to a specific time (e.g., "What were you doing *yesterday at 4:30pm*?") and the ones specifying a time interval (e.g. "how many times did you go to the lab *last week*?").

For our CoBot robots, we identify a total of five task-related filters. The first task-related filter is the task ID. This filter specifies what kind of task the query refers to. We derive the remaining four filters directly from the arguments of the semantic frame for each task. These filters are: destination, source, object, and person. Not all the filters are applicable to all tasks, Figure 4 shows which filter is applicable to each task.

	GoTo	FindAndFetch	Escort
Time	✓	✓	✓
Task ID	✓	✓	✓
Destination	✓	✓	✓
Source		✓	
Object		✓	
Person			✓

Fig. 4: Filters and their applicability

IV. Grounding Sentences To Queries

We frame the problem of understanding a question about the tasks the robot executed as inference in a joint probabilistic model over the possible groundings Q, a parse of the question P, and given a Knowledge Base KB. The goal is to find the query Q^* that maximize our joint probability. Formally:

$$\arg\max_{Q} p(Q, P|KB)$$

Using Bayes Theorem we rewrite our joint distribution in terms of the grounding and parsing probabilities:

$$p(Q, P|KB) = p(Q|P, KB)p(P|KB)$$

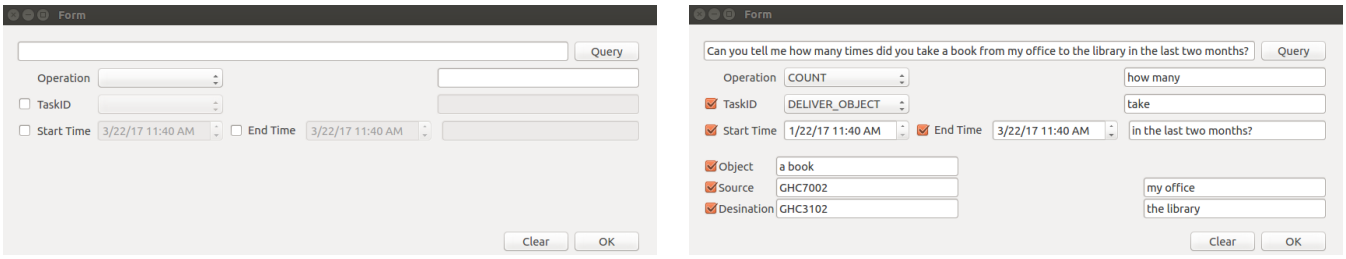
Finally, since the parser is independent from the Knowledge Base, we can further simplify the equation for the probability of a query as:

$$p(Q, P|KB) = p(Q|P, KB)p(P)$$

In the next three subsections we first describe our parsing model, we then detail the Knowledge Base used for the grounding model, and finally we introduce our model to compute the grounding probability.

A. Parsing

We adopt a shallow semantic parsing. We label each word in a sentence and then we chunk together words with the same labels. The labels we have chosen closely match the structure of a query as explained in Section III-C. The set of label L we used is: $\{$*Operation*, *TaskID*, *Time*, *Destination*, *Source*, *Object*, *Person*, *None*$\}$. We use the label *Operations* for the words referring to one of the three operation we can perform on the logs (CHECK, COUNT, and SELECT). The *TaskID*, *Destination*, *Source*, *Object*, and *Person* labels are used to select the words pertaining to one of the task-related filters. The *Time* label matches words identifying time-related filters. Finally the label *None* is used for words that can be ignored. Figure 6 shows an example of an input sentence and its parse.



(a) (b)

Fig. 5: The form used for our experiments. (a) Shows the form as initially presented to the user, and (b) shows all the fields completed.

Given a sentence S of length N we need to label each words s_i as l_i, where $l_i \in L$. We model the parsing problem as a function of pre-learned weights w and observed features ϕ. Formally:

$$p(P) = p(l_1, ..., l_N | s_1, ..., s_N)$$
$$= \frac{1}{Z} \exp(\sum_{1}^{N} w \cdot \phi(l_i, s_{i-1}, s_i, s_{i+1}))$$

We learned this model using a conditional random field, where ϕ is a function producing binary features based on the part-of-speech tags of the current, next, and previous words, as well as the current, next, and previous words themselves.

"Can you tell me how many times did you take a book from my office to the library in the last two months?"

(a)

Can you tell me [how many times]$_{\text{Operation}}$ did you [take]$_{\text{TaskID}}$ [a book]$_{\text{Object}}$ from [my office]$_{\text{Source}}$ to [the library]$_{\text{Destination}}$ [in the last two months?]$_{\text{Time}}$

(b)

Fig. 6: (a) Example sentence and (b) its parse. The words without label are actually labeled as *None*, the label has been omitted for clarity.

B. Knowledge Base

The goal of the knowledge base is to store and re-use grounding of natural language expression. The Knowledge Base stores groundings in the form of three different binary predicates. The first argument of each predicate is always a natural language expression e. The second argument of a predicate represents the grounding γ of the expression e. To each predicate we attach a confidence score that we indicate as $C_{(e,\gamma)}$. The three predicates we use are *OperationGroundsTo*, *TaskGroundsTo*, and *LocationGroundsTo*.

The predicate *OperationGroundsTo* is used to store the mapping from the words labeled by the parser as *taskID* to one of the possible operation on the logs (e.g., *OperationGroundsTo*("How many times", COUNT)). Similarly, *TaskGroundsTo* saves the mapping from words labeled as *taskID*, to the semantic frame actions (e.g., *TaskGroundsTo*("take", FindAndFetch)). Finally the predicate *LocationGroundsTo* maps the expression labeled as *Source* or *Destination* to one of th location-labels used by the ROS message (e.g., *LocationGroundsTo*("my ofice", GHC7002).

C. Query Grounding

At the beginning of this section we showed how our model is composed by a grounding probability and a parsing probability. Here we detail how the grounding probability is computed. A query Q is composed by an operation op and a set of i filters f. We assume that an operation and its filter are independent and can therefore be grounded separately. Accordingly, we rewrite our initial grounding probability as:

$$p(Q|P, KB) = p(op|P, KB) \prod^{i} (f_i|P, KB)$$

The probability of an operation, or a filter, is computed using the confidence score of the Knowledge base predicates. To compute the probability of a specific grounding γ^* for an expression e we use the following formula:

$$p(\gamma^*|P, KB) = \frac{C_{\gamma^*,e}}{\sum^{\gamma} C_{\gamma,e}}$$

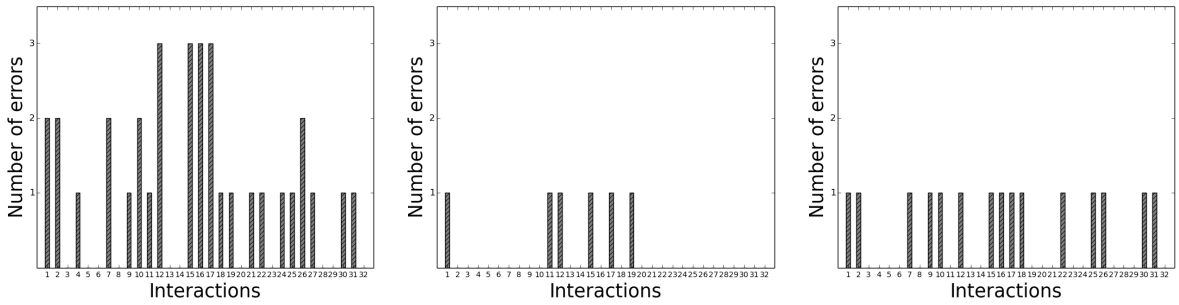
The knowledge base contains predicates that allow for the grounding of the operation and the filters about the task, the source and the destination of a task. The filters about the person and the object do not require grounding as, in the logs, they are already saved as natural language expressions. Finally for the time filters we opted to use a third party library [25].

V. Experimental Evaluation

A. Experimental Setup

In order to test our approach we designed a form that users can fill to ask questions about the history of tasks executed by our CoBot robots. Figure 5 shows the form before and after being completely filled. The form is designed to match the structure of a query and it has three main components: a large text box used for user input, a first grounding sections for filters common to most queries, and a second grounding section for the task-related filters.

The first grounding section of the form has two drop-down menus and two date/time edit box. These four elements are



(a) (b) (c)

Fig. 7: On the Y axis the error of the approach proposed in grounding sentences to query, on the X axis the number of queries as time progress. (a) Shows the overall error, (b) The error in grounding the operation only and, (c) the error in grounding the task.

all placed on the left-hand side of the form. The drop-down menus allow the user to select, respectively, the grounding for the operation of the query and, if needed, for the taskID. The two date/time edit box allow the user to set time-related filters. Time interval are specified by using both boxes, while specific time filters (e.g., "yesterday at 4:30pm") can be set by disabling the second date/time box. For each element in this section the form has, on the left, a small text-box showing what part of the sentence matches the selected operation or filter. This text-box are initially compiled based on the structure extracted by the semantic parser.

The second section of the form allows the user to specify task-specific filters such as the object for a FindAndFetch task or the person for an Escort task. This section is initially hidden, it only appears once a task has been selected and, since each task has different arguments, is dynamically generated. For the filters that require a grounding (i.e., Source, and Destination) the form provides two text boxes, on the left the ones grounding and, on the right, the ones for the matching structure extracted by the parser. For the remaining filters (i.e. Person, and Object) the form only shows the part of the sentence that is going to be used to search the logs.

B. Experimental Results

The "Ok" button at the end of the form allows the user to confirm that all the information entered is correct. When the button is clicked two things happen. First, we update the Knowledge Base, by reading the content of the elements on the right-hand side of the form we generate new predicates and add them to the Knowledge Base. If a predicate is already present in the Knowledge Base we simply update its weight. Second, we save the content of the text boxes on the right-hand side. The fragment of the sentence constitute a parse that has been validated by the user and that can be used to improve the parser itself.

In order to test the approach proposed we asked a user to enter the 32 sentences we used to train the semantic parser in the form described above. Given the small size of the corpus, we decided to use the same sentences to train the parser and to test our proposed approach. Doing so, we aim at minimizing the error induced by the parser itself and properly assess the grounding process. The user was asked to enter the sentence and, if the from returned an incorrect or missing grounding, to provide the right one. The initial Knowledge Base used for grounding was completely empty.

We evaluate our approach in terms of errors made when grounding a sentence to a query. The error is measured as the number of edits the user made to the field of the form before confirming the query. Figure 7a shows the total number of errors made for each of the 32 sentences used.

Figure 7b shows the errors made in grounding the part of the sentence referring to the operation to be performed on the logs. The Knowledge Base is unable to provide the grounding for the operations only on 6 occasion. This suggests that the language used to refer to operations is consistent across the 32 sentences we gathered. Moreover, after the 19th sentence, our approach always grounds the operation correctly.

Figure 7c shows the error made in grounding the taskID filter. We expected this error to have a comparable trend to the one recorded for the operation as the number of possible grounding is the same for the operation and the tasks. By looking at the Knowledge Base after the experiment we find two possible explanations. First, it appears that there is less variety in the expressions used to refer to operation compared to the ones used for task. For the COUNT operation the we only have to expressions, "how many" while for a FindAndFetch task e have at least three: "take", "bring", "deliver". Second, the expressions used to refer to a task sometimes overlaps. The verb "take" can be used to refer to a FindAndFetch task as well as to an Escort task, creating more uncertainty and therefore potential for error.

Finally, Figure 8 shows the number of facts recorded in the Knowledge Base as we go through the sequence of sentences. The number of facts initially increases quickly but after the 19th sentences the slope of the curve reduces. This matches what we show in Figure 7b where we highlighted that the approach proposed stops to ask questions about the operation after the 19th sentence. Last, we note that the plot seems to be converging toward a maximum number of facts; this suggests that, in the long run, our approach will no longer require corrections from the users.

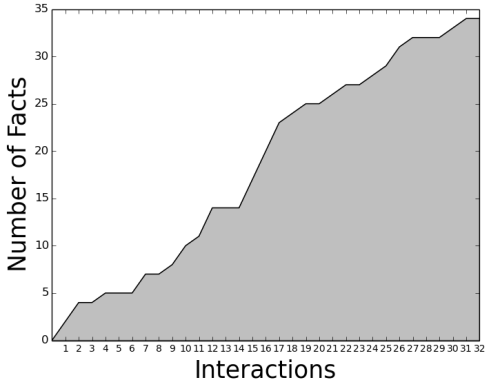


Fig. 8: The number of facts stored in the Knowledge Base as the number of interaction increases.

VI. Conclusions And Future Work

In an effort to make service robots more transparent to their user we presented a novel approach to enable a robot to understand questions about the history of tasks it previously executed. We formalize this problem as mapping an input sentence to a query to be executed against the logs recorded by the robot during its runs. We characterize a query about the history of the tasks executed by a service robot as an operation to be executed on the records of the logs that match the filters defined by the user. In this work we consider both task-related and time-related filers.

The problem of mapping an input sentence to a query is framed as probabilistic inference over a shallow semantic parse of the sentence and a Knowledge Base that the robot builds, iteratively, through the interaction with its users. The predicates used in our Knowledge Base are designed to allow to store and re-use groundings for different parts of each query. Our results, on a corpus of 32 sentences, show that the approach proposed improves in grounding sentences to queries. The number of corrections by the user decrease as the system goes through successive queries.

In the future we plan on deploying the approach proposed on the real CoBot robots. This will allow us to gather more example of sentences relating to queries about the task the robot executed in the past and further validate both our grounding and parsing approach. In order to deploy the approach presented in this paper in the wild we will also investigate the best way for the robot to provide answer to the users once the query has been executed.

Acknowledgments

This research is partly sponsored by DARPA under agreements FA87501620042 and FA87501720152, and by the National Science Foundation under award NSF IIS1012733 and IIS1637927. The views and conclusions contained in this document are solely of the authors.

References

[1] M. Veloso, J. Biswas, B. Coltin, and S. Rosenthal, "CoBots: Robust Symbiotic Autonomous Mobile Service Robots," in *Proceedings of IJCAI'15, the International Joint Conference on Artificial Intelligence*, July 2015.

[2] J. Biswas and M. Veloso, "The 1,000-km challenge: Insights and quantitative and qualitative results," *IEEE Intelligent Systems*, vol. 31, no. 3, pp. 86–96, 2016.

[3] S. Rosenthal, S. P. Selvaraj, and M. Veloso, "Verbalization: Narration of autonomous mobile robot experience," in *Proceedings of IJCAI*, vol. 16, 2016.

[4] V. Perera, S. P. Selveraj, S. Rosenthal, and M. Veloso, "Dynamic generation and refinement of robot verbalization."

[5] K. Chen, D. Lu, Y. Chen, K. Tang, N. Wang, and X. Chen, "The intelligent techniques in robot kejia–the champion of robocup@ home 2014," in *Robot Soccer World Cup*. Springer, 2014, pp. 130–141.

[6] Y. Chen, F. Wu, W. Shuai, N. Wang, R. Chen, and X. Chen, "Kejia robot–an attractive shopping mall guider," in *International Conference on Social Robotics*. Springer, 2015, pp. 145–154.

[7] P. Khandelwal, S. Zhang, J. Sinapov, M. Leonetti, J. Thomason, F. Yang, I. Gori, M. Svetlik, P. Khante, V. Lifschitz, J. K. Aggarwal, R. Mooney, and P. Stone, "Bwibots: A platform for bridging the gap between ai and human–robot interaction research," *The International Journal of Robotics Research*, 2017.

[8] N. Hawes *et al.*, "The STRANDS project: Long-term autonomy in everyday environments," *CoRR*, vol. abs/1604.04384, 2016.

[9] A. K. Dey, "Explanations in context-aware systems." in *ExaCt*, 2009, pp. 84–93.

[10] B. Y. Lim, A. K. Dey, and D. Avrahami, "Why and why not explanations improve the intelligibility of context-aware intelligent systems," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2009, pp. 2119–2128.

[11] T. Kulesza, S. Stumpf, M. Burnett, S. Yang, I. Kwan, and W.-K. Wong, "Too much, too little, or just right? ways explanations impact end users' mental models," in *Visual Languages and Human-Centric Computing (VL/HCC), 2013 IEEE Symposium on*. IEEE, 2013, pp. 3–10.

[12] A. Bussone, S. Stumpf, and D. O'Sullivan, "The role of explanations on trust and reliance in clinical decision support systems," in *Healthcare Informatics (ICHI), 2015*. IEEE, 2015, pp. 160–169.

[13] T. Winograd, "Procedures as a representation for data in a computer program for understanding natural language," DTIC Document, Tech. Rep., 1971.

[14] K. Jokinen and G. Wilcock, *Dialogues with Social Robots: Enablements, Analyses, and Evaluation*. Springer, 2016, vol. 999.

[15] Y. Artzi and L. Zettlemoyer, "Weakly supervised learning of semantic parsers for mapping instructions to actions," *Transactions of the Association for Computational Linguistics*, vol. 1, pp. 49–62, 2013.

[16] C. Matuszek, E. Herbst, L. Zettlemoyer, and D. Fox, "Learning to parse natural language commands to a robot control system," in *Experimental Robotics*. Springer, 2013, pp. 403–415.

[17] M. Forbes, R. P. Rao, L. Zettlemoyer, and M. Cakmak, "Robot programming by demonstration with situated spatial language understanding," in *Robotics and Automation (ICRA), 2015*. IEEE, 2015, pp. 2014–2020.

[18] T. Kollar, S. Tellex, D. Roy, and N. Roy, "Toward understanding natural language directions," in *Human-Robot Interaction (HRI), 2010*. IEEE, 2010, pp. 259–266.

[19] A. S. Huang *et al.*, "Natural language command of an autonomous micro-air vehicle," in *Intelligent Robots and Systems (IROS), 2010*. IEEE, 2010, pp. 2663–2669.

[20] S. A. Tellex, T. F. Kollar, S. R. Dickerson, M. R. Walter, A. Banerjee, S. Teller, and N. Roy, "Understanding natural language commands for robotic navigation and mobile manipulation," 2011.

[21] C. J. Fillmore, "Frames and the semantics of understanding," *Quaderni di semantica*, vol. 6, no. 2, pp. 222–254, 1985.

[22] C. F. Baker, C. J. Fillmore, and J. B. Lowe, "The berkeley framenet project," in *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*. Association for Computational Linguistics, 1998, pp. 86–90.

[23] V. Perera, R. Soetens, T. Kollar, M. Samadi, Y. Sun, D. Nardi, R. van de Molengraft, and M. Veloso, "Learning task knowledge from dialog and web access," *Robotics*, vol. 4, no. 2, pp. 223–252, 2015.

[24] B. J. Thomas and O. C. Jenkins, "Roboframenet: Verb-centric semantics for actions in robot middleware," in *Robotics and Automation (ICRA), 2012*. IEEE, 2012, pp. 4750–4755.

[25] A. X. Chang and C. D. Manning, "Sutime: A library for recognizing and normalizing time expressions." in *LREC*, vol. 2012, 2012, pp. 3735–3740.