Language-Based Bidirectional Human And Robot Interaction Learning For Mobile Service Robots

Vittorio Perera
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
vdperera@cs.cmu.edu

Thesis Proposal

Abstract

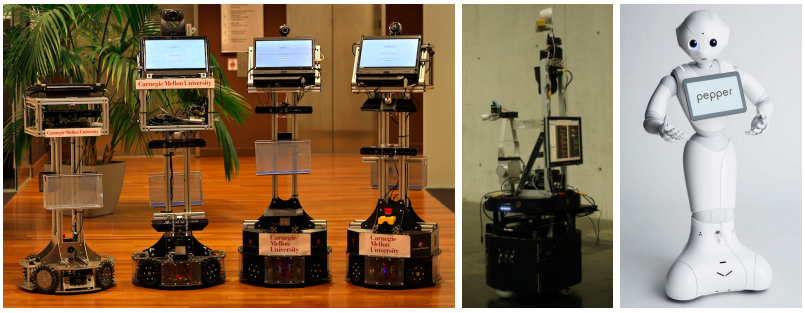
We believe that it is essential for robots that coexist with humans to be able to interact with their users in a seamless way. This thesis advocates the use of language as a rich and natural interface for human-robot interaction. Previous work on language-based human-robot interaction has extensively focused on enabling robots to understand spatial language in the context of users giving step-by-step directions to robots. We assume a mobile service robot, like our CoBot, is equipped with a map of its environment and is able to autonomously navigate to a desired goal position. This thesis will address the problem of user-to-robot interaction, but is going to assume users provide a high-level specification of the task (e.g., "Take the package to the small-size lab and then bring me tea") rather than step-by-step navigational instructions. Furthermore the thesis will focus on a novel robot-to-user interaction where the robot is able to adapt to different users, to answer user queries about its state (present, past or future), and to proactively take information-providing actions (i.e., reporting on the outcome of a task after finishing its execution). Summing up, this thesis will contribute a novel language-based bidirectional interaction approach for mobile service robot: from users to the robot and vice-versa. We will evaluate the work in real and extensive real-data constructed simulated environments.

Contents

1 Introduction

We believe that it is essential for robots that coexist with humans to be able to interact with their users in a seamless way. In this thesis we advocate the use of language as a rich and natural interface for human-robot interaction. Natural language allows users to interact with robots without requiring the need to being robot developers and understand the robot in a special way while, at the same time, offering a high expressive power.

Mobile service robots come in many different shapes and forms. Figure 1 shows different examples of service robots, ranging from a mobile base to full anthropomorphic. Although service robots can be implemented with a wide range of hardware they all share the same goal: offering services, that require to accurately navigate in the environment, to their users.



(a) (b) (c)

Figure 1: Different types of service robots: (a) CoBot, (b) Kejia, and (c) Pepper

In this proposal we make two assumptions about the nature of a service robot. First, the robot is equipped with a map of its environment. This map can be based on the architectural floorplans of the building, or it can be autonomously learned by the robot. We assume that the robot is able to autonomously navigate to specific (x, y) coordinates using its map. Second, we assume the robot is deployed in a environment with humans (e.g., an office building, a college or an hospital) where it interacts with a range of users persistently and repeatedly throughout time.

Our work focuses on two main directions: 1) to enable users to request the robot to perform tasks, and 2) to enable the robot to report on the tasks executed. We introduce these two directions, respectively, in Section 1.1 and Section 1.2.

1.1 User asking for tasks

Our CoBot robots, shown in Figure 1(a), currently offer to their user four different tasks: going to a specific place in the building, delivering a messages, transporting objects between locations, and escorting people to their destination. These tasks are requested, scheduled, and executed by the integration of multiple modules and algorithms, including scheduling, localization, navigation, and symbiotic autonomy [1]. The implementation uses ROS services[1], which are

[1] http://wiki.ros.org/Services

akin to remote procedure calls. For each CoBot service, there is a ROS-based definition of a parameterized procedure.

From a linguistic point of view, we model the tasks the robot executes as semantic frames [2, 3]. Each robot task corresponds to a single frame and to each argument of the ROS service corresponds a frame element. Given our representation, the problem of a robot receiving a spoken command, becomes the problem of finding the frame corresponding to the task required and instantiating all the frame elements for execution. The problem is twofold, first we need to extract from a natural language sentence the structure of the frame (i.e., parsing), and second we need to instantiate the frame elements to locations on the robot map (i.e., grounding). The solution we have developed, and propose to continue to investigate, is based on: 1) a semantic parser, in the form of a Conditional Random Field learned from a corpus, which enables the extraction of the frame structure from a sentence, and 2) a knowledge base that stores the grounding for natural language expression learned via dialog.

Our solution further allows our robots to understand *simple commands*, such as "Can you please go to the kitchen on Hillman side?" or "Can you please take this box to Christina?," and learn corresponding groundings. We have further investigated much more complex queries, which can be commonly used. As an example consider the following: "I need you to first bring me a cup of tea, or a bottle of water, or a soda, and then go to Christina's office and ask her to order more bottled water." The user is asking the robot to execute a sequence of two tasks, delivering an object and delivering a message. Moreover, the first task includes several options for which object needs to be provided. We define this type of sentences as *complex commands*, for which we identify four different sources of complexity:

1. a user asks for a *set* of tasks rather than a single one
2. *alternative* options are available to instantiate at least one of the frame elements evoked by the command
3. a user asks for a *sequence* of tasks
4. the task has to be executed only if a *condition* is met.

In order to understand complex commands, the solution we adopted breaks a convoluted sentence into simpler ones, each referring to a single frame. The procedures to break a complex sentence are defined by *templates*. Templates are structures in the syntactic parse of a sentence together with rules to modify the sentence and create simpler ones. Once a complex command has been reduced to simpler sentences, each referring to a single frame, we can use the approach proposed initially to execute the command [4].

Finally, in the approach we have developed and propose to keep, the robot learns the groundings of referring expressions in a knowledge base (KB). Since we assume the robot is persistently deployed in a shared environment with multiple users, the robot maintains personalized data in its KB. In order to have the robot more effectively learn the groundings of referring expressions and reduce the number of questions asked to its users over time, we introduce a method to infer user-specific groundings from the ensemble of the KB's gatherings. Our approach considers referring expressions as items in an item-item collaborative filter. We enable the robot to measure the similarity of two expressions with an

ad-hoc metric and use this measure to infer possible user-specific groundings, which are offered to the users.

1.2 Robots reporting on tasks

We consider a mobile service robot after it has finished executing navigation tasks. For a specific just-finished task, we investigate when the user asks the robot to recount the path it has traveled. For one specific path traveled, the robot could provide many different explanations based, for instance, on the level of detail provided or the language used. The Verbalization Space (VS) [5] captures the variations in possible explanations for a specific robot experience (e.g., navigation), with three main dimensions:

- **Abstraction:** to represent the vocabulary used in the explanation.
- **Locality:** to describe the segments of the route plan that the user is interested in.
- **Specificity:** to indicate the concepts to introduce in the explanation.

In our work, we contribute the use of a large corpus to learn a mapping from the language used to ask the robot to recount its navigation experience to specific points in the Verbalization Space. The point in the verbalization space can then be used to generate an explanation that matches the user expectations.

A user might be interested in querying not only about the path the robot took but also about its general state. We propose to enable a mobile service robot to answer queries about its present, past, and future states. In the six years of their deployment, our CoBot robots recorded over 660GB of data logs. These logs record all the messages exchanged by different components of the robot software during each deployment. They contain information at execution level (e.g., (x, y, θ) position, battery voltage), task level (e.g., the task being executed, the time since the task start), and human-robot interaction level (e.g., events recorded by the GUI, results of speech recognition).

The approach we propose is based on three steps. Given the large amount of data recorded by the CoBot robots, the first two steps are aimed at reducing the search space. In the first step, we use the temporal references in the queries to narrow down the search space to (ideally) a single log file. In the second step, we use the semantic of the query to select a handful of messages to inspect. In the third and last step, we select the correct routine to execute on the messages selected in the earlier steps.

Finally, we propose to enable our robots to provide historically-contextualized information. A robot that has just executed a task can report many different data on the execution. We believe that a raw number (e.g., the time taken, the distance traveled) is not very relevant or informative. Instead we propose to put this data in perspective using the history of past executions. Our proposed approach has four components. The first component, *feature identification*, defines a set of features relevant to describe the tasks executed by the robot (e.g., day, time, user of task, number of people encountered). The second component, *statistical execution history*, derives a statistical description of the tasks in the past from the logs collected (e.g., normal statistical distributions of navigation time per map segment). The third component, *relevant event detection*, identifies interesting events in a specific execution with respect to the statistical execution

history (e.g., outlier in execution time). The last component, *proactive language generation*, includes the learning to map appropriate language to events (e.g., "...more people than usual...") as well as the autonomous generation of the complete utterances to describe the relevant events and proactively convey them to the human users (e.g., "today there were more people than usual in the kitchen").

In the rest of this proposal document, we first review the relevant literature (Section 2). Then we present the work that has already been completed (Section 3), and our proposed remaining work (Section 4). We conclude with our expected thesis contributions, as well as a timeline.

2 Related Work

Since SHRDLU [6], there has always been a great interest in enabling artificial agents and their users to interact using natural language. In this section, we review the relevant literature: Section 2.1 focuses on works related to enabling agents, and in particular robots, to understand human inputs. Section 2.2 focus on the literature regarding agents generating text for users to read or listen.

2.1 Human-to-robot

Frame-like structures are a common approach to represent the action the robot has to carry out given an input sentence. BIRON [7], intended as a robot companion for home-tour scenarios, tries to parse the input using an LR(1)-parser and, when information is missing, it uses a slot-filling policy to drive the dialog and recover the missing information. MARCO [8] is a system to follow free-form natural language route instruction in a simulated environment. Using a probabilistic context free grammar (PCFG), the language is converted into a *compound action specification* as an imperative model of what the robot has to do under which circumstances. In [9] the focus is enabling an incremental discourse engine: to do so the authors use *discourse context*, representing the shared discourse history between user and robot, and *interaction templates*, boilerplates for discourse subject. In RoboFrameNet [10], the input is first parsed using the Stanford dependency parser and then using *lexical units* (i.e., example sentence, with hand annotated frame element) is mapped into semantic frames.

Another widely used approach is to reduce the input sentence to some logical form that can then be evaluated to answer queries or execute tasks. Godot [11, 12] translates sentences into first order logic statements to enable the robot to answer questions about its state and about the environment. Similarly, [13] uses description logic to enable a robot to act as a helper in a moving scenario. The most common logical representation used is λ-calculus. In [14] the goal is to create a joint model for context and semantics to enable robots to follow navigational instruction, by using λ-calculus to model actions as Neo-Davidsonian events. In [15] the goal is the same, namely to follow navigational instructions, but λ-calculus is used to turn the input sentence into Robot Control Language (RCL), a LISP-like language that formalizes the actions the robot needs to take. The robot KeJia, first designed to compete in Robocup@Home [16, 17] and then deployed as a mall guide [18], uses λ-calculus to transform a natural language query into a logical that is fed directly into the robot's planner module. In [19]

λ-calculus is combined with semantic annotations using temporal and dynamic logic to generate logic expressions representing robot goals directly from natural language directives.

Probabilistic models are also used to map language to robot actions. In [20], to remove the kinesthetic requirements of Programming by Demonstration (PbD), the problem of finding the right action to execute is framed as probabilistic inference. The goal is to find the command $c^* \in C$ to maximize the joint probability distribution $p(C, S, L)$ over commands C, robot states S and language L. In [21] the authors introduce Spatial Description Clauses (SDC) to model the structure of language in directions. SDCs are composed by a figure, a verb and a spatial relationship, and a sentence is modeled as a sequence of SDCs. Once again the goal is to find the path $p^* \in P$ to maximize the conditional joint probability $p(P, S|O)$ over the paths P, the sequence of SDCs S, and the observation of the robot O. While initially designed to follow instructions in a 2-dimensional environment, SDC can also be used to control air vehicles in three dimensions [22]. The Generalized Grounding Graph (GGG) framework [23], builds upon SDC and further generalizes by removing the assumption of a flat structure. Instead, GGG dynamically instantiates a probabilistical graphical model based on the command's hierarchical structure.

The work reviewed shares a particular focus on enabling a robot to understand spatial language. Understanding such language is critical when a robot needs to follow directions, or the user needs to provide step-by-step instructions. In our work, we assume the robot is able to autonomously perform its tasks. Therefore, we focus our attention on enabling a robot to understand a high level specification of its task where spatial language is unlikely to appear.

2.2 Robot-to-human

Robot-To-Human interaction beyoond dialog is less studied. We therefore start by reviewing the related field of Computational Storytelling. This field is surveyed in [24], where the author focuses on the production of stories by agents and is mostly interested in the creativity and unexpectedness of the generated stories. The "stories" we aim to generate are based on sensory data collected by a robot and, therefore, we are more interested in the actual story telling process. In [25] the authors are interested in Interactive Storytelling (IS) and are concerned with scalability issues and real-time performance of a system generating larger stories. They frame the problem of story telling as knowledge-based planning and show how using PDDL 3.0 [26] is possible to encode narrative control knowledge as state trajectory constraints and decompose the planning problem using landmark planning. This approach is further investigated in [27] where the same approach is used to generate stories with multiple storylines advancing at the same time through interleaved narrative. StatsMonkey [28] is a system for writing newspaper-style accounts of sporting events using internet-gathered statistical data from the games. The statistical data is mostly in the form of play-by-play or box-scores, and is combined using three different types of narrative, namley extreme or rare events, crossing thresholds, and comparison with expectation. These narratives are combined using *angles*, hand-designed rules ordering the narratives by importance, which helps provide a big picture of the game.

While the systems reviewed so far deal with agents autonomously creating

and telling stories, in [29] we have the first example of actual robot commentating a live game of robot soccer. The communication happens between a team of two humanoid commentators and the audience and uses both speech and gestures. The game is summarized as it evolves by a *Game History* that only retains the most salient events. In order to create an entertaining but consistent commentary, the output of the commentators is generated by combining two functions: a deterministic function that maps events in the game history to a set of actions the robots can take, and a non-deterministic function to select one of the options from the set generated. In [30], a NAO robot is used to give direction to visitors in a building. The robot generates a plan to the visitor destination running A^* on an annotated map. The path generated, a sequence of segments describing the distance and direction of travel, is then translated into natural language and gestures. To avoid the generation of verbose directions, the system adopts a set of simplification rules (e.g., skipping insignificant paths, merging coincident paths, or short-circuiting longer explanations).

3 Completed Work

In this section we go over the work that has already been completed. This section is divided in two parts, in Section 3.1 and 3.2 we review our work on understanding simple and complex commands; in Section 3.3 we report our work on adapting the robot behavior to user inputs. Our work is implemented on the CoBot robots [31] (see Figure 1(a)). CoBot is a service robot meant to autonomously navigate in our buildings and to complete tasks for its users. The tasks CoBot is currently able to perform are: going to a specific place in the building, delivering a messages, transporting objects between locations, and escorting people to their destination. Each of these tasks requires a number of arguments in order to be executed correctly. For instance, to go to a specific location the argument needed is the robot destination. On the other hand, to transport an object the robot needs three arguments: the object to be carried, the location where it can be found, and the location where the object needs to be delivered.

3.1 Understanding atomic commands

Given the structure of CoBot tasks, we decided to represent each of them with a semantic frame [32], where frame elements correspond to arguments needed to execute a task. The problem of understanding a natural language command is framed as extracting all the relevant frames from a sentence, and ground their elements to location, object or person known to the robot. We first look into how to understand an *atomic command*. We define as an atomic command a sentence referring to a single frame, where each frame element is uniquely instantiated.

Our approach to understand and execute an atomic command is based on three building blocks: 1) a probabilistic model that is able to interpret input sentences for service tasks, 2) a knowledge base that represents the mapping from referring expressions to locations and actions, and 3) a dialog system that is able to add facts to the Knowledge Base after interacting with users. The goal of the Knowledge Base (KB) is to maintain and to re-use knowledge that the

robot acquires as a part of the dialogs it has with humans. Our KB consists of categories, single argument predicates, and relations. The categories in our KB correspond to the labels extracted by our parser (i.e., action, location, person, and object). The relations, one for each of the categories, are used to save mapping from a referring expression to a grounding. The KB also keeps track of how many times the specific arguments of a relation have been correctly grounded; we refer to this number as the *count* of a relation. An example of a populated KB, as gathered by the robot can be seen in Figure 2.

`actionGroundsTo('go to', GoTo)`	`5.68`
`actionGroundsTo('goto', GoTo)`	`2.36375`
`actionGroundsTo('go 2', GoTo)`	`0.0425`
`actionGroundsTo('goto', BringObject)`	`0.3`
`actionGroundsTo('get', BringObject)`	`2.15`
`locationGroundsTo('the small size lab', 7412)`	`7.96375`
`locationGroundsTo('small sized lab', 7412)`	`0.0425`
`locationGroundsTo('the small size lav', 7412)`	`0.68`

Figure 2: An example KB as gathered by the robot. This section of the KB shows two relations `actionGroundsTo` and `locationsGroundsTo`. On the left the *count* of each relation is shown.

The second building block of our approach is a probabilistic model. In particular, we consider a joint probabilistic model over the groundings Γ, a parse P, and a speech S given access to a knowledge base KB. Formally we want to compute:

$$\arg\max_{\Gamma,P,S} p(\Gamma, P, S|\text{KB})$$

Our joint probability distribution can be rewritten as:

$$p(\Gamma, P, S|\text{KB}) = p(\Gamma|P, \text{KB}) \times p(P|S) \times p(S)$$

which defines the three main components of our model. First we have $p(S)$, a speech to text model that, given as input an audio file (i.e., the command spoken by the user) returns multiple possible transcriptions and confidence for each of them[2].

The parsing model, $p(P|S)$, takes as input one of the possible transcriptions from the speech to text model and parses it into a semantic structure (i.e., a semantic frame). The structure extracted is composed by an action, evoking the frame corresponding to the task to be executed, and a variable number of arguments. To extract such structure we first label each word in the sentence using a Conditional Random Field (CRF). The CRF model is learned from a small corpus using binary features that include part of speech tags for the current, next and previous word as well as the current, next and previous word in the sentence. The labels used to in the CRF correspond to the categories used in the knowledge base: action, location, person, and object. Once each word has been labeled we greedily groups together words with the same label. An example of the initial input transcription and the corresponding frame extracted is shown in Figure 3.

[2]We used the Google speech recognizer for our experiments.

`Go to the small size lav`	`0.85`
`go 2 small sized lab`	`0.425`
`goto the small size lab`	`0.2125`
`get the small sized love`	`0.2125`

(a) Speech recognition results

$[\texttt{Go to}]_{action}$ $[\texttt{the small size lav}]_{location}$	`0.8`
$[\texttt{go 2}]_{action}$ $[\texttt{small sized lab}]_{location}$	`0.1`
$[\texttt{goto}]_{action}$ $[\texttt{the small size lab}]_{location}$	`0.3`
$[\texttt{get}]_{action}$ $[\texttt{the small sized love}]_{object}$	`0.7`

(b) Parses

Figure 3: (a) Multiple transcriptions and their confideence for the same spoken commands, and (b) the frame extracted using the CRF.

To compute the grounding model $p(\Gamma|P, \text{KB})$ we first rewrite the conditional probability using Bayes rule:

$$p(\Gamma|P; \text{KB}) = \frac{p(P|\Gamma; \text{KB}) \times p(\Gamma; \text{KB})}{\sum_P p(P|\Gamma; \text{KB}) \times p(\Gamma; \text{KB})}$$

The prior over groundings $p(\Gamma; \text{KB})$ is computed by looking at the counts of each element of Γ in the knowledge base (the category predicates). The other term $p(P|\Gamma; \text{KB})$ is computed by grounding referring expressions for actions, locations, people and objects (the relation predicates). In particular, if a parse P is composed by an action a evoking a frame and its elements e we can compute this probability as:

$$p(P|\Gamma; \text{KB}) = p(a|\Gamma, \text{KB}) \times p(e|\Gamma; \text{KB})$$

where the two terms $p(a|\Gamma, \text{KB})$ and $p(e|\Gamma; \text{KB})$ can be computed directly from the count of the predicates in the knowledge base.

The last building block of our system is the dialog used by the robot to fill unknown components. Given a natural language command, which is parsed into frames, we use a slot-filling technique to drive the dialog. If any part of the frame is missing the robot will ask a question. If there is no action, then it will ask the person to say the action. If one of the frame elements is missing, the robot will ask for it according to frame structure. In the case where the frame template is filled, but there is no grounding for either the action or the arguments, then the robot asks explicitly for the grounding of these components. Finally, at the end of the dialog, the robot always asks for confirmation before executing an action.

To evaluate this approach we compared it with two baselines. In the *Task Baseline*, the robot executes the task without learning any semantic information about the environment while using the *Learning Baseline*, it executes the assigned task while learning semantic knowledge about the environment but without using a knowledge base. Our results show how the approach proposed is able to dramatically reduce the number of question the robot asks before executing its task, and therefore achieve a more natural interaction with the user

when compare with the two baselines. All the details of the approach introduced here are available in [2, 3]

3.2 Understanding complex commands

The previous section introduced our approach to understand atomic commands. Here we present our method to understand *complex commands*. We define as a *complex command* any sentence referring to robot tasks that is not an atomic command. In particular, we identify four different types of complexity that can arise in a command:

Set of tasks: The user asks the robot to perform a set of tasks, therefore the command refers to multiple frames. (e.g., "Go to the lab and bring these papers to my office.")

Disjunctive task elements: The command refers to a single frame but some of the frame elements are not univocally instantiated. (e.g., "Bring me some coffee or tea.")

Explicit sequence of tasks: The user asks the robot to perform an ordered sequence of tasks. (e.g., "Go to the lab and then to my office.")

Conditional sequence of tasks: The user asks the robot to perform an ordered sequence of tasks, while using conditionals. (e.g., "Bring me some coffee if it's freshly brewed.")

Given that we are already able to understand and ground atomic commands, our goal is to reduce a complex command to a structured sequence of atomic commands. To represent this structure we defined 4 operators, one for each type of complexity, and we use them to connect atomic commands and preserve the original meaning of the command. The operators defined are:

- **AND** used for commands referring to a set of tasks (e.g., "Go to the lab and bring these papers to my office." → [Go to the lab] **AND** [Bring these papers to my office])
- **OR** used when an element of the frame can be instantiated in multiple ways. (e.g., "Bring me some coffee or tea." → [Bring me some coffee] **OR** [Bring me some tea])
- **THEN** orders tasks in a sequence (e.g., "Go to the lab and then to my office." → [Go to the lab] **THEN** [Go to my office])
- **IF** used for the sequence of tasks involving conditionals. (e.g., "Bring me some coffee if it's freshly brewed." → [Coffee is freshly brewed] **IF** [Bring me some coffee])

Our approach is based on the use of *templates*. A template is a hand-designed structure, composed by nodes (both inner and leaf) of a syntactic parse tree together with specific rules to decompose a complex commands into simpler sentences (possibly atomic commands). Figure 4 shows one example of template, with the template structure highlighted in boldface, for each of the four operator defined. Our *template-based algorithm* proceeds as follows. We

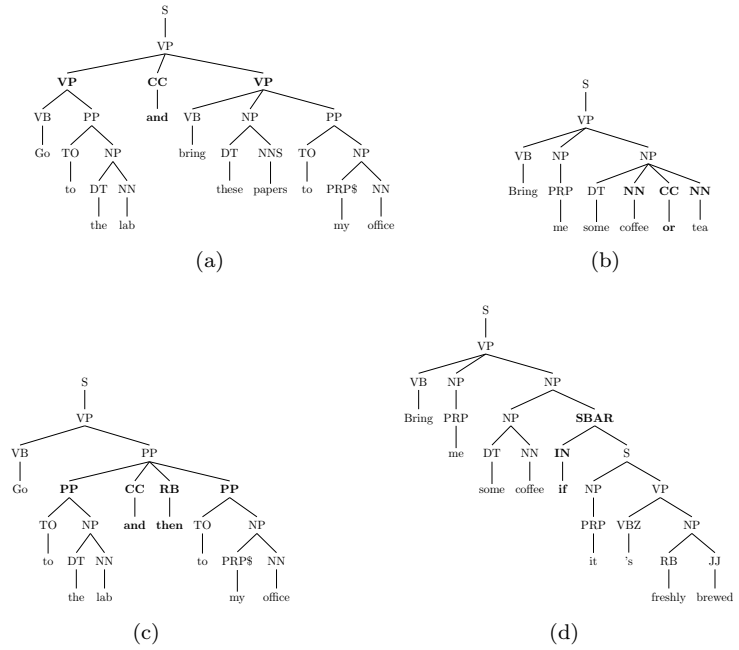


Figure 4: Four example of templates. The structures extracted are: (a) [Go to the lab] **AND** [Bring these papers to my office], (b) [Bring me some cofee] **OR** [Bring me some tea], (c) [Go to the lab] **THEN** [Go to my office], (d) [Coffee is freshly brewed] **IF** [Bring me some coffee]

first obtain a syntactic parse tree[3] of the input sentence. We then search the parse tree breadth-first, left to right. Whenever we find a structure matching one of the templates defined we apply the corresponding rule to break the sentence in simpler components and apply, recursively, the algorithm to the simpler phrases extracted.

In order to test our approach we gathered a corpus of 100 complex commands. The commands had levels of complexity from one to eight (the level of complexity is measured as the number of atomic commands composing he complex command). Our algorithm is able to correctly decompose a complex command in atomic commands in 72% of the cases. In order to recover the remaining 28% of the sentences we propose two dialogue strategies the robot can use to recover the structure of a complex command.

The first strategy, called *rephrasing dialog* simply consist in asking the user to rephrase the initial sentence while giving short examples of known structure it can understand. Using this strategy the number of sentences correctly decomposed increase to 88%. To recover the remaining 12% we introduce the *structure-based dialog*. When using this strategy the robot first asks the user if the command received is simple or complex. If it is a simple command, the robot asks for confirmation for it and then executes it. If it is a complex command, the robot needs to recover all of its components: the operator, and the two simpler commands. Since both the simpler commands can be themselves

[3]For our experiment we used the Stanford Parser.

complex the dialog will recurs on each of them. This dialog strategy always ask $2n - 1$ questions for a sentence of complexity n. Even if the structure-based dialog can lead a large number of question, it guarantees that the structure of a complex command is recovered correctly. The details of the approach described here, together with an algorithm to use the structure extracted from a complex command to improve the execution time of them, are available in [4].

3.3 Adapting to user inputs

In the work of Rosenthal et al. [5] the authors introduce the idea of *verbalization* as "the process by which an autonomous robot converts its own experience into language". Consequently, the *verbalization space* (VS) represents the variations in possible explanations for the same robot experience.

The VS is the space of possible verbalizations, consisting of a set of axes or parameters along which the variability in the explanations are created. For the purpose of describing the path of our CoBot robots, the VS contains three orthogonal parameters: abstraction, locality, and specificity.

Abstraction represents the vocabulary or corpus used in the text generation. At its lowest level the explanation are generated in terms of the robot's world representation, directly using points (x, y, θ) in the robot path. At the highest level of abstraction the explanation generated provides information in terms of landmarks, corridors, and bridges from our annotated map.

Locality describes the segment(s) of the route plan that the user is interested in. In the most general case, the user is interested in the plan for the entire path. In other cases they may only be interested in a particular Region defined as a subset of points in our map (e.g., the 8th floor or in Building 2), or only interested in the details around a Location (e.g., 8th floor kitchen or office 4002).

Specificity indicates the number of concepts or details to discuss in the text. At the lowest level of specificity the robot only provides a general picture (i.e., a short description specifying the start and end points or landmarks, the total distance covered and the time taken). At the highest level it contains a complete description of the route plan, including a sentence between every pair of turning points.

In [5] the authors introduce a *Variable Verbalization Algorithm* that takes as input specific levels for the three verbalization dimensions, a path traveled by the robot and returns a recount of the robot experience. The problem we address in our work is how to find a point in the verbalization space matching the user request for a verbalization task (e.g., when a user ask for a "detailed recount of the path take" we want to return a point with very high specificity). Moreover we are also interested in how to move in the verbalization space if a user iteratively refine his request for verbalization.

To understand how users might ask a robot for verbalization tasks we crowdsourced[4] a corpus of 2400 sentences. The survey used to gather the corpus was designed in a way such that, each sentences entered by the users corresponded to

[4]for this task we used Amazon Mechanical Turk, www.mturk.com

either a high or low value on one of the three dimensions of the VS. This allowed us to automatically label the corpus using six different labels. Using different combination of features (unigrams and bigrams both in lemmatized and surface form) and models (Naive Bayes, and Logistic Regression) we show how it is possible to learn a model that with good accuracy (73.37%) classifies correctly to which dimension in the VS and what intensity each sentence refers to. To further validate our approach we run a user study. In this study we found that, by using the model trained to infer the values of the three dimensions in the VS, the robot is able to generate answers that either match or partially match the expectations of users in 82.6% of the cases. The details of the approach just described, are available in [33].

3.4 Adapting to specific users

In Section 3.1 we showed how, by using a Knowledge Base, we can enable a robot to understand and ground natural language commands. In our experiments the grounding learned for each user are saved in a single Knowledge Base. In this scenario, if two users refer to different locations using the same expression, the robot selects the most likely grounding. Unfortunately, the probability of each grounding is computed using the *count* of each predicate and, since it represents a measure of how often an expression has been mapped to a specific location, the robot is going to select the grounding meant by the user who interacted most with the robot (i.e., the user who provided more instances of the mapping from the given expression to a specific location).

A simple solution to this problem would be to maintain user-specific KB's. Unfortunately this would dramatically increase the number of interaction required by the robot to learn he grounding for any referring expression. Our solution is to maintain user-specific KB's but to leverage techniques from recommender system, in particular item-item collaborative filtering [34], to infer possible grounding for expression that are not present in a specific user KB.

Our first step is to find a measure of similarity between natural language expressions. Here we focus on the grounding of location, therefore we consider the predicates $locationGroundsTo(E, L)$ where E is an expression such as "the lab" or "the kitchen" and L is a location on the map of he robot. First we find, for each expression, the group of user sharing the same grounding for it. This can easily be done by iterating over all the user-specific KB the robot currently has. For a specific referring expression E, we indicate the result of this first step as, $\mathcal{S}_E = \{S_{L=1}, ..., S_{L=n}\}$ where $S_{L=1}$ is the set of users whose KB share the predicate $locationGroundsTo(E, 1)$.

Next, to compare two referring expressions E and E', we compute the similarity between the two sets of sets $\mathcal{S}_E$ and $\mathcal{S}_{E'}$. To do so we use the best match algorithm [35] using the Jaccard Distance $J(A, B)$:

$$J(A, B) = \frac{|A \bigcup B| - |A \bigcap B|}{|A \bigcup B|}$$

as similarity measure between two sets A and B. The best match algorithm finds, for each set $S_i \in \mathcal{S}_E$, its best match in $\mathcal{S}_{E'}$ by looking for the set $S' \in \mathcal{S}_{E'}$ such that $J(S_i, S') \leq J(S_i, S'_j)$ for all $S'_j \in \mathcal{S}_{E'}$. A first measure of how well $\mathcal{S}_{E'}$ represents $\mathcal{S}_E$ is given by the sum of distances of each set $S \in \mathcal{S}_E$ to its best

representative in $S' \in \mathcal{S}_{E'}$, that is:

$$\sum_{i=1}^{n} \min_{j=1,\dots,m} J(S_i, S'_j)$$

In order to make this measure of similarity symmetric we also add the distances from every member of $\mathcal{S}_{E'}$ to its best representative in $\mathcal{S}_E$, and average by the total number of sets. The final formula of the Best Match Distance (BMD) is therefore:

$$\frac{\sum_{i=1}^{n} \min_{j=1,\dots,m} J(S_i, S'_j) + \sum_{j=1}^{m} \min_{j=1,\dots,n} J(S'_j, S_)}{|\mathcal{S}_E| + |\mathcal{S}_{E'}|}$$

When a user gives a command to the robot and the Knowledge Base for the specific user (KB*) does not have a match for the referring expression used, E^*, we can leverage the BMD between E^* and all the other referring expressions already present in the KB to infer possible groundings.

Starting with an empty set C, the first step is for the robot to iterate over all the referring expressions E already present in the user KB. For each of those expressions the robot computes the BMD with the E^*. If the Best Match Distance is below a threshold τ the robot adds E to C. The result of this operation is $C = \{E \in KB^* : BMD(E, E^*) < \tau\}$ which, intuitively, represents the set of referring expression similar enough to E^*.

Next, the robot iterates over all the KB of the remaining users. For each of them, if they share the predicate $locationGroundsTo(E, L)$ with KB* for any of the E in C, the location $\mathcal{L}$ such that $locationGroundsTo(E^*, \mathcal{L}) \in KB$ is added as a potential grounding with weight equal to $BMD(E^*, E)$. Finally the candidate with the highest weight is returned as the correct grounding.

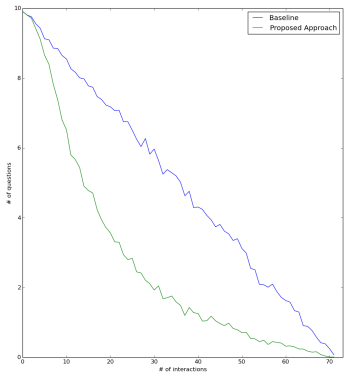


Figure 5: The results obtained in our simulation. On the X axis the progressive iterations. On the Y axis the average number, over 100 simulation, of the questions asked for every 10 interactions.

In order to test our approach we created a simulated environment based on the physical maps of a single floor in our department building. We populated our simulation with a total of 60 students and we considered a total of six

different referring expression. The six expressions used are: kitchen, elevator, bathroom, stair, advisor, and lab. Each user is assigned to a specific office in the building and, for four of referring expression (kitchen, elevator, bathroom, and stair), their preference is set to the closest location on the map matching the expression. The remaining two referring expressions (advisor and lab) are set arbitrarily but are directly connected.

In each simulation we run a total of 720 interactions, where each of the 60 users ask the robot to reach one of the 6 referring expressions twice. The order of the interactions is randomized every time a new simulation is generated. We compare our approach with a naive baseline that, whenever lacks the grounding for a referring expression ask the user for it and updates the user KB. Our approach instead tries to infer the correct grounding and only asks if the result inferred is not correct. Figure 5 shows the average number of question asked over 100 simulations where both our approach start with an empty KB. This work is currently in submission as [36].

4 Proposed Work

In our initial work, we focus on Human-to-Robot communication, where the human asks the robot to perform a task. The goal is to transform the human requests in language, and in any subsequent dialog, into information that the robot can use *to execute* the task.

The other part of our research enables the robot to report on its experience, after execution. We have researched on learning to map human requests about the robot's experience to a verbalization space. We propose to further research on Human-to-Robot requests that inquire for additional information about the robot execution (Section 4.1).

Furthermore, we propose to focus on a new Robot-to-Human communication, in which the robot initiates offering information about its experience (without the need for a request from the human) by determining deviations from its normal execution and then translating such experience to appropriately qualified language, e.g., "Today there were a lot of people in the kitchen area" (Section 4.2).

4.1 Understanding status queries

We propose to enable service robots to answer queries about their current, past, and future states. The robot can retrieve the answers to such queries by, respectively, looking into its current execution status, the logs of its previous deployment, and its schedule.

For our CoBot robots, we believe it is particularly interesting to have them answer questions about their past experience. In the 5 years of their deployment, from 2011 to 2016, our CoBot robots recorded over 660GB of data logs. The CoBot robots are designed using the modular ROS architecture[5], where each module, or node, runs independently and is able to exchange message with other nodes. Nodes are able to exchange messages by subscribing or publishing to a topic, and the topic defines the format and information of each message. Fig. 6 shows all the nodes running on the robot and the topics each subscribes and

[5]http://www.ros.org/

publishes to. Depending on the task the robot is being deployed for, the logs record all or part of the messages exchanged during the execution. The size of each log depends on the length of the deployment, but the data is produced at rates varying from 15 to 33 KB/sec.

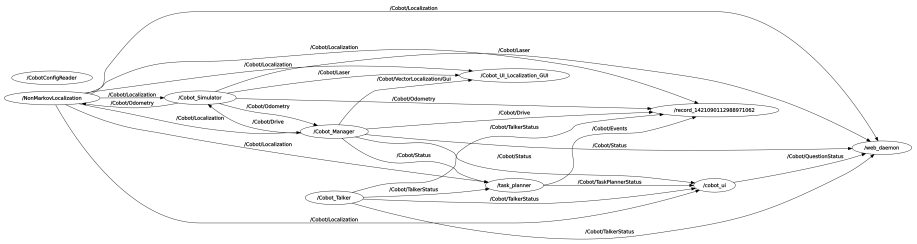


Figure 6: The nodes running on CoBot connected by the topics they publish and subscribe to.

The logs are recordings of the messages exchanged by the modules of the robot software. These messages contain all sort of information, which we can categorize in three different levels:

- **Execution Level:** this is the lowest level and contains information such as: the (x, y) position of the robot on the map together with its orientation, the current voltage of the batteries, the linear and angular velocity, all the readings from the sensors and information about the navigation, such as the current path being followed and whether it is blocked or not.
- **Task Level:** at this level we find all the information regarding the tasks the robot can execute, which includes: the current task being executed, the current subtask (if any is present), the time since the task has started, the estimated time until completion and the list of the tasks already scheduled.
- **Human-Robot Interaction Level:** finally at this level, we find the information related to the interactions with humans, such as: event recorded by the GUI (i.e. pressing a button), results of speech recognition, number of humans detected, open doors detected, and the questions asked by the robot.

Table 1 shows a few examples of the messages recorded in the logs, their content, and the level they belong to.

The logs recorded by our CoBot robots contain very large amounts of data. Therefore, to have robots extract specific bits of the logs and answer the queries received, we need to restrict the search space. We propose a three-step process.

The first step is for the robot to narrow down the search space. Each log file saves data for a single run, from when the robot is turned on until is turned off. Most of the questions querying a robot state will inevitably have a temporal reference. The robot can use such temporal reference to inspect only a limited number of log files, ideally narrowing down to one and therefore limiting the search space.

The second step is still aimed at reducing the search space. The amount of messages to be considered across all the topics can still be very large but,

Topic	Content	Level
CoBot/Drive	x-velocity, y-velocity, y-velocity, ...	Execution
CoBot/Localization	x, y, angle, angleUncertainty, locationUncertainty, map , ...	Execution
CoBot/TaskPlannerStatus	currentTask, currentSubTask, currentNavigationSubTask, timeBlocked, taskDuration, subTaskDuration, navigationSubTaskDuration, navigationTimeRemaining, timeToDeadline ...	Execution, Task
CoBot/QuestionStatus	question, multiple-choice, click-image, ...	Task, HRI
CoBot/DoorDetector	door-X, door-Y, door-status	HRI
CoBot/SpeechRecognition	utterances, confidences	HRI

Table 1: Table showing the topics, their content and the level they are assigned to.

using the semantic of the query received, the robot can do two things: reduce the number of topics to inspect, possibly down to one and, and, for each topic, only select some specific subfield.

The third and last step consists of selecting the correct data processing routine to execute. The routine will take as input the logs selected in the first step and will only operate on the topic and message fields selected in the second step. The choice of the routine will be guided by the semantic of the query. We identify two types of routines: *functional routines* that only need to return a single value, and *procedural routines* that use the data found in the logs to execute some computational steps. Similarly to what we have done for commands, we propose to model these routines using semantic frames. For each routine, we will define a set of frame elements that provide all the information to execute the routine and retrieve the information requested.

Let us now show an example that goes through the three steps just outlined. The query asked by the user is the following: "What were you doing yesterday at 3pm?" The first step is to use the temporal reference in the query to select the right log to inspect. The expressions "yesterday" and "3pm" specify a date and time that the robot can use to select the correct log file. Next the robot can narrow down its search to a specific topic. Since the query is asking for "What were you doing" the robot can restrict its search to the `CoBot/TaskPlannerStatus` topic and in particular only look at the `currentTask` field. Finally, the routine that needs to be executed is a functional routine that will look for a task that started before 3pm and ended after.

Let us now consider a second example where the user is asking the robot "Were you busy this morning?" Similarly to the first example, the first step is to use temporal reference to narrow down the search. Since the query is asking about "this morning" instead of looking for a log file the robot will use its current message list. The message list contains the same information as the logs and therefore the robot can use it seamlessly. In the second step, the robot identifies which topic and message field to use as input for the procedure that

will be executed next. Since the query asks about "being busy" the robot will again look at the `CoBot/TaskPlannerStatus` topic and in the `currentTask` field. The procedural routine executed will look for all the tasks executed in the morning (i.e., from 8am to 12pm).

We propose to enable robots to answer queries about their present and future state as well, but the logs can only be queried after they have been completely recorded. Fortunately the same information stored in the logs can be retrieved from the messages currently being exchanged to answer queries about the present, or from the schedule of the robot to answer queries about the future.

4.2 Reporting on task executions

When a robot is executing or has just executed a task, it can report on a variety of data, such as: the time required to execute the task, the distance traveled, or the number of users encountered while executing the task. We believe this information is not very relevant if considered without the appropriate context. We propose to enable robots to provide reports that take into account the history of the robot experience.

The approach we propose to enable robots to provide history-contextualized information is composed by three main steps. The first step is to define a set of features that are relevant to describe the tasks executed by the robot. For our mobile service robot we identify, but we are not limited to, the following:

- **Time Executing:** the time spent actually executing the task;
- **Time Blocked:** the time spent blocked while trying to executing the task (e.g., when stopped because of an obstacle);
- **Total time:** the total time from when the task was started until its end;
- **Distance:** the distance traveled to execute the task;
- **Users:** the number of users encountered while executing the task;
- **Navigational Information:** average velocity along the x and y axis as well as rotational velocity'

The second step makes use of the logs collected to derive a statistical description of the tasks executed in terms of the features derived in the first step. We propose to provide this statistical description in term of an appropriate parametrized statistical distribution, such as Normal with mean and variance.

In this third step the robot autonomously generate a description starting from the statistical distribution derived from the logs and the information of the current task. The current task could be an outlier when compared to the historic distribution or be aligned with it, the problem we address is how to automatically generate a language description for each possible case. As a concrete example consider the task of traveling between two rooms, room 7002 and room 7412. This task takes on average 132". The robot just finished executing the same task but it took 120", the robot will report to the user by saying "I just came from room 7002 and it took me *slightly less* than usual". On the other hand if the same task took 526" the robot will say "I just came from room 7002, it took me *forever* to get here".

We propose to learn appropriate descriptions for each of the features identified in the first step, as well as the mappings for the relation between current execution and historical data to language through crowd-sourcing of interaction with the robot.

4.3 Evaluation

In our experience, a limiting factor when developing novel approaches for human and robot interaction is the availability of data for training end evaluation. We aim to use a rich simulation environment to evaluate our proposed contributions. Our CoBot robots provide a simulation environment developed for testing their navigation algorithm, see Figure 7.

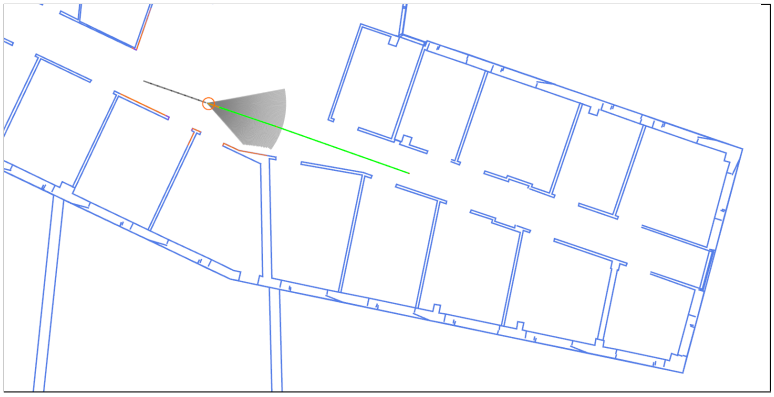


Figure 7: A screenshot from the navigation simulation [37]. The map is drawn in blue, the robot in orange, the LIDAR sensing is displayed in gray, the past trajectory in black and the future one in green.

This environment provides information about the current position of the robot, the lidar sensing and the trajectory of the robot. We propose to enhance this simulator to include, interaction with humans in terms of their queries. In order to keep our simulation realistic, we plan to gather the queries and humans behavior to be implemented in the simulator through crowd-sourcing. Moreover, we propose to extensively demonstrate the results of our proposed contribution on the real robots.

4.4 Timeline

The timeline we foresee for completing the proposed work is the following:

- **January 2017 - April 2017:** Understand status queries, Section 4.1
- **May 2017 - August 2017:** Report task execution, Section 4.2
- **September 2017 - December 2017:** Evaluation and thesis writing
- **January 2018:** Thesis defense

5 Expected Contributions

We expect the thesis will provide three main contributions:

1. An approach to understand user-specific task-related queries. This include:
    - A Knowledge Base to record mappings from referring expressions to task and elements of the environment (person, objects and locations)
    - A probabilistic model to parse and ground the input sentence
    - A template-based algorithm to break complex commands into atomic commands
    - An algorithm to infer possible groundings for specific users based on previous interactions with different users
2. An approach to understand status-related queries. This includes:
    - An approach to effectively search the relevant data in the history of the robot
    - A model to ground user input sentence into operation to be performed on the logs
3. An approach to enable robots to provide historically-contextualized information on the task executed.

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*, Buenos Aires, Argentina, July 2015.

[2] T. Kollar, V. Perera, D. Nardi, and M. Veloso, "Learning environmental knowledge from task-based human-robot dialog," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 4304–4309.

[3] 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.

[4] V. Perera and M. Veloso, "Handling complex commands as service robot task requests," in *Proceedings of the 24th International Conference on Artificial Intelligence*. AAAI Press, 2015, pp. 1177–1183.

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

[6] T. Winograd, "Procedures as a representation for data in a program for understanding natural language," Ph.D. dissertation, Ph. D. dissertation, MIT, 1970.

[7] A. Haasch, S. Hohenner, S. Hüwel, M. Kleinehagenbrock, S. Lang, I. Toptsis, G. A. Fink, J. Fritsch, B. Wrede, and G. Sagerer, "Biron–the bielefeld robot companion," in *Proc. Int. Workshop on Advances in Service Robotics*. Citeseer, 2004, pp. 27–32.

[8] M. MacMahon, B. Stankiewicz, and B. Kuipers, "Walk the talk: Connecting language, knowledge, and action in route instructions," *Def*, vol. 2, no. 6, p. 4, 2006.

[9] T. Brick, P. Schermerhorn, and M. Scheutz, "Speech and action: Integration of action and language for mobile robots," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2007, pp. 1423–1428.

[10] B. Thomas and O. C. Jenkins, "Verb semantics for robot dialog," in *Robotics: Science and Systems Workshop on Grounding Human-Robot Dialog for Spatial Tasks, Los Angeles, CA, USA (June 2011)*, 2011.

[11] J. Bos, E. Klein, and T. Oka, "Meaningful conversation with a mobile robot," in *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 2*. Association for Computational Linguistics, 2003, pp. 71–74.

[12] J. Bos and T. Oka, "A spoken language interface with a mobile robot," *Artificial Life and Robotics*, vol. 11, no. 1, pp. 42–47, 2007.

[13] S. Lemaignan, R. Ros, E. A. Sisbot, R. Alami, and M. Beetz, "Grounding the interaction: Anchoring situated discourse in everyday human-robot interaction," *International Journal of Social Robotics*, vol. 4, no. 2, pp. 181–199, 2012.

[14] 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.

[15] 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.

[16] X. Chen, J. Ji, J. Jiang, G. Jin, F. Wang, and J. Xie, "Developing high-level cognitive functions for service robots," in *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*. International Foundation for Autonomous Agents and Multiagent Systems, 2010, pp. 989–996.

[17] 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.

[18] 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.

[19] J. Dzifcak, M. Scheutz, C. Baral, and P. Schermerhorn, "What to do and how to do it: Translating natural language directives into temporal and dynamic logic representation for goal management and action execution," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 2009, pp. 4163–4168.

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

[21] T. Kollar, S. Tellex, D. Roy, and N. Roy, "Toward understanding natural language directions," in *2010 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 2010, pp. 259–266.

[22] A. S. Huang, S. Tellex, A. Bachrach, T. Kollar, D. Roy, and N. Roy, "Natural language command of an autonomous micro-air vehicle," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 2010, pp. 2663–2669.

[23] 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.

[24] P. Gervás, "to storytelling and creativity," 2009.

[25] J. Porteous, M. Cavazza, and F. Charles, "Applying planning to interactive storytelling: Narrative control using state constraints," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 1, no. 2, p. 10, 2010.

[26] A. Gerevini and D. Long, "Plan constraints and preferences in pddl3," *The Language of the Fifth International Planning Competition. Tech. Rep. Technical Report, Department of Electronics for Automation, University of Brescia, Italy*, vol. 75, 2005.

[27] J. Porteous, F. Charles, and M. Cavazza, "Plan-based narrative generation with coordinated subplots," 2016.

[28] N. D. Allen, J. R. Templon, P. S. McNally, L. Birnbaum, and K. J. Hammond, "Statsmonkey: A data-driven sports narrative writer." in *AAAI Fall Symposium: Computational Models of Narrative*, 2010.

[29] M. Veloso, N. Armstrong-Crews, S. Chernova, E. Crawford, C. McMillen, M. Roth, D. Vail, and S. Zickler, "A team of humanoid game commentators," *International Journal of Humanoid Robotics*, vol. 5, no. 03, pp. 457–480, 2008.

[30] D. Bohus, C. W. Saw, and E. Horvitz, "Directions robot: In-the-wild experiences and lessons learned," in *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2014, pp. 637–644.

[31] J. Biswas and M. Veloso, "Localization and Navigation of the CoBots over Long-term Deployments," *International Journal of Robotics Research*, vol. 32, no. 14, pp. 1679–1694, December 2013.

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

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

[34] B. Sarwar, G. Karypis, and J. Konstan, Joseph a nd Riedl, "Item-based collaborative filtering recommendation algorit hms," in *Proceedings of the 10th international conference on W orld Wide Web*. ACM, 2001, pp. 285–295.

[35] M. K. Goldberg, M. Hayvanovych, and M. Magdon-Isma il, "Measuring similarity between sets of overlapping clusters," in *Social Computing (SocialCom), 2010 IEEE Second Intern ational Conference on*. IEEE, 2010, pp. 303–308.

[36] V. Perera and M. Veloso, "Planning robot action under language-derived hard and soft constraints," in *27th International Conference on Automated Planning and Scheduling (current submission)"*.

[37] Biswas, "Vector map-based, non-markov localization for long-term deployment of autonomous mobile robots," PhD Thesis, Carnegie Mellon University, 12 2014.