Robotics **2015**, *4*, 223-252; doi:10.3390/robotics4020223

OPEN ACCESS
robotics
ISSN 2218-6581
www.mdpi.com/journal/robotics

Article

Learning Task Knowledge from Dialog and Web Access

Vittorio Perera [1,*], Robin Soetens [2], Thomas Kollar [1], Mehdi Samadi [1], Yichao Sun [3], Daniele Nardi [4], René van de Molengraft [2] and Manuela Veloso [1]

[1] School of Computer Science, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA; E-Mails: tkollar@cmu.edu (T.K.); msamadi@cs.cmu.edu (M.S.); mmv@cs.cmu.edu (M.V.)
[2] Department of Mechanical Engineering, Eindhoven University of Technology, Den Dolech 2, Eindhoven; E-Mails: robinsoetens@gmail.com (R.S.); m.j.g.v.d.molengraft@tue.nl (R.M.)
[3] State Key Laboratory of Industrial Control Technology, Zhejiang University, 38 Zheda Road, Hangzhou 456555, China; E-Mail: inchaos.sun@gmail.com
[4] Department of Computer, Control, and Management Engineering "Antonio Ruberti", "Sapienza" University of Rome Via Ariosto 25, Rome 00185, Italy; E-Mail: nardi@dis.uniroma1.it

* Author to whom correspondence should be addressed; E-Mail: vdperera@cs.cmu.edu.

Academic Editors: Nicola Bellotto, Nick Hawes and Mohan Sridharan

Received: 20 March 2015 / Accepted: 5 June 2015 / Published: 17 June 2015

Abstract: We present **KnoWDiaL**, an approach for **L**earning and using task-relevant **Know**ledge from human-robot **Dia**log and access to the **W**eb. KnoWDiaL assumes that there is an autonomous agent that performs tasks, as requested by humans through speech. The agent needs to "understand" the request, (*i.e.*, to fully ground the task until it can proceed to plan for and execute it). KnoWDiaL contributes such understanding by using and updating a Knowledge Base, by dialoguing with the user, and by accessing the web. We believe that KnoWDiaL, as we present it, can be applied to general autonomous agents. However, we focus on our work with our autonomous collaborative robot, CoBot, which executes service tasks in a building, moving around and transporting objects between locations. Hence, the knowledge acquired and accessed consists of groundings of language to robot actions, and building locations, persons, and objects. KnoWDiaL handles the interpretation of voice commands, is robust regarding speech recognition errors, and is able to learn commands involving referring expressions in an open domain, (*i.e.*, without requiring a lexicon). We present in detail the multiple components of KnoWDiaL, namely a frame-semantic parser, a probabilistic grounding model, a web-based predicate evaluator, a dialog manager, and the weighted predicate-based Knowledge Base. We illustrate the knowledge access and

updates from the dialog and Web access, through detailed and complete examples. We further evaluate the correctness of the predicate instances learned into the Knowledge Base, and show the increase in dialog efficiency as a function of the number of interactions. We have extensively and successfully used KnoWDiaL in CoBot dialoguing and accessing the Web, and extract a few corresponding example sequences from captured videos.

Keywords: knowledge acquisition; knowledge based systems; knowledge transfer; robots; intelligent robots; service robots; mobile robots; human robot interaction; speech; speech recognition

1. Introduction

Speech-based interaction holds the promise of enabling robots to become both flexible and intuitive to use. When the robot is a mobile robot servicing people, speech-based interaction will have to deal with tasks involving locations and objects in the environment. For example, a human might command a robot like CoBot to "go to Dana's office" or to "get me a coffee". The mobile robot must then infer the type of action it should take, the corresponding location parameters and the mentioned object.

If we place no restrictions on speech, interpreting and executing a command becomes a challenging problem for several reasons. First, the robot may not have the knowledge necessary to execute the command in this particular environment. In the above examples, the robot must know where "Dana" or "a coffee" is located in the building, and it should understand the type of action a user asks for when using phrases like "get me" or "go to". Second, performing robust speech recognition can be challenging, resulting in multiple interpretation strings of which some might contain a partially correct translation while others can be less intelligible. Finally, speech-based interaction from untrained users requires understanding a wide variety of different ways to refer to the same location, object or action.

To bridge the semantic gap between the robot and human representations, we introduce **KnoWDiaL**, an approach for robot **L**earning of task-relevant environmental **Kno**wledge from human-robot **Dia**log and access to the **W**eb, as illustrated in Figure 1.

KnoWDiaL contains five primary components: A frame-semantic parser, a probabilistic grounding model, a Knowledge Base, a Web-based predicate evaluator, and a dialog manager. Once a user provides a spoken command, our frame-semantic parser maps the entire list of speech to text candidates to pre-defined frames containing slots for phrases referring to action types and slots for phrases referring to action parameters. Next, using the Knowledge Base, the probabilistic grounding model maps this set of frames to referents. In our system, referents are either known action types or room numbers, which we assume are known to the robot, (e.g., as physical coordinates on the robot's map for navigation). In case required information is missing, the dialog manager component attempts to fill missing fields via dialog or via Web searches. In the event that it attempts a Web search it generates a query to OpenEval, which is a Web-based predicate evaluator able to evaluate the validity of predicates by extracting information

from unstructured Web pages [1,2]. When the action type and required fields are set, the dialog manager asks for confirmation, executes the task and updates the Knowledge Base.

(a)

Dialog Example
User: Get me a coffee.
CoBot: According to OpenEval this object is most likely to be found in location "kitchen". Is that correct?
User: Yes.

(b)

Learned Facts
objectGroundsTo(coffee, 7602)
actionGroundsTo(get, BringObject)

(c)

Figure 1. Example of an interaction between a user and mobile service robot CoBot, with KnoWDiaL: (**a**) Speech-based verbal interaction; (**b**) Action and object inferred from spoken command, and access to the web with OpenEval for object location inference; (**c**) Learned knowledge base with "7602" being the room number of location "kitchen".

KnoWDiaL is implemented on and continues to be part of CoBot, a service robot able to autonomously navigate in office buildings [3]. With CoBot, we perform several empirical evaluations of KnoWDiaL. The first performance criterion we evaluate is the overall correctness of the knowledge that has been accumulated. Asking untrained users to provide commands, we evaluate how well the system is able to infer the correct action type, and the correct parameters required to execute this action type. In an environment constrained to eight locations and 40 objects, KnoWDiaL directly understands 90% of the requested action types, and 70% of the required arguments. These percentages were achieved in an experiment with 91 user-interactions from untrained users. Using the Web-based predicate evaluator improved the performance of parameter grounding by around 15%. A second performance criterion evaluates how well KnoWDiaL uses the knowledge it accumulates to increase dialog efficiency. Compared to a non-learning baseline system that directly asks for the action type and its corresponding parameters, our approach required significantly fewer user-questions (*i.e.*, 214 compared to 318).

With KnoWDiaL we contribute a dialog system that is efficient in terms of required user interaction and robust in terms of speech recognition and parsing errors. We handle knowledge in a probabilistic manner, representing uncertainty with probability distributions. Since many aspects of knowledge required to understand task-based dialog are intrinsically uncertain, we consider this an important contribution. Knowledge has been handled probabilistically by others (e.g., [4]), but our system handles Knowledge Base updates incrementally, and uses knowledge within a concrete, real-world, mobile, service robot.

Furthermore, KnoWDiaL does not rely on a single highly probable speech-to-text candidate being available. We are doing inference over the entire chain of probabilities involved in voice commands, and ground commands based on each of the speech-to-text candidates, while also keeping track of multiple parses and multiple plans the robot can execute. Such an approach is important especially

when dealing with mobile service robots, that operate in noisy environments with a large number of different users. These types of environments usually come with a flat probability distribution over speech-to-text candidates. Given that we are looking for specific task-related language, a speech-to-text candidate that gets a low probability score from the speech recognizer might actually be very important in understanding the command. As suggested by [5], it might turn out to be impossible to make automatic speech recognition as good as human speech recognition without taking context and interpretation into account. The KnoWDiaL way of grounding commands is a step in this direction.

Learning, via the Knowledge Base, ensures that KnoWDiaL does not make the same mistakes over and over again. Key to this is a confirmation question at the end of the dialog. Learned facts are added to the Knowledge Base only when confirmed by the user. Since we are storing small phrases in our predicates, as opposed to the entire speech-to-text sentence, our dialog system can generalize learned facts from one specific command to a comparable but slightly different new command. Therefore with KnoWDiaL we contribute a system that learns to understand more and more commands over time, while reducing the required user interaction.

Apart from learning, extracting information from the Web is another reason KnoWDiaL is efficient in terms of required user interaction. By querying the World Wide Web for common sense information (e.g., coffee is probably in a kitchen), we avoid asking the user. We have created a dialog system able to retrieve information from the Web in real time (*i.e.*, while engaged in dialog), which we also consider to be a contribution. In order to efficiently search the Web, KnoWDiaL processes its Knowledge Base also when not in dialog, to find valid referring expressions for future Web searches.

This article is organized as follows. In Section 2, we briefly discuss related work. Section 3 presents the complete KnoWDiaL system with its five main components. Section 4 presents our empirical evaluation, namely the controlled experiments, and an illustration of the different types of dialog interaction with KnoWDiaL implemented on CoBot. Section 5 concludes the article and discusses future work.

2. Related Work

KnoWDiaL is tested and implemented on CoBot, a mobile service robot [3,6,7]. It builds on our work on learning location groundings from spoken commands [8]. However, it moves beyond this initial work by enabling the robot to also deal with tasks related to the transportation of objects, and by incorporating Web searches as an additional source of information during dialog. To achieve the latter, we build on our earlier work in creating a Web-based predicate evaluator [1,2].

The structure of our parsing step represents a computational instantiation of previously introduced parsing formalisms [9–11], including ternary expressions to capture relations between words in a sentence [12]. Our parser is trained as a structured Conditional Random Field (CRF) [13], implemented by using the CRF++ toolkit [14]. For tagging, we use the Python NLTK library [15].

From very early on (e.g., with SHRDLU [16]), systems have exploited the compositional structure of language to statically generate a plan corresponding to a natural language command [17,18,20,21]. Our work moves beyond these frameworks by defining a probabilistic model that deals with the uncertainty in speech, parsing, and plans corresponding to a certain command.

Understanding route instructions is a popular application of task-based dialog. Generative and discriminative models have been introduced [22,23], but there was no explicit representation of the uncertainty in speech recognition, parsing and grounding of the natural language command. A motion grammar has also been investigated (e.g., [24]), providing guarantees of completeness and correctness for robotic systems when performing task decomposition, but not including teaching on how to execute the commands. A flat, fixed action space was used to train a CRF that followed route instructions [25].

In general, dialog systems have tended to focus on the language of specific tasks, such as finding objects, following directions, or manipulating objects. A variety of approaches aim at teaching a robot about objects and locations, some using a dialog system but with focus on using human-robot dialog to support visual object recognition and with reasoning about objects deterministically, based on a single highly-probable speech-to-text candidate [26]; some using large amounts of written text to extract common-sense knowledge about object locations and common tasks [27]. Groundings of spatial language (route directions and mobile manipulation) have been demonstrated using Generalized Grounding Graphs [28,29]. Complete action plans have been instructed from spoken language by humans within robot home assistive tasks (e.g., [30]). Furthermore, within the competition scenarios of RoboCup@Home (e.g., [31]) spoken language is captured with the aim of extracting action plans.

Research in task-constrained dialog for a mobile humanoid robot has been conducted, but with the requirement of lexicon, and without directly extracting environmental knowledge from dialog [32]. Others have extracted knowledge from dialog, but with a deterministic Knowledge Base, as opposed to our probabilistic approach. Rather than keeping track of a probability distribution in case conflicting information is gathered, existing knowledge is deterministically overwritten, or the user is queried to provide more information [33].

Using the World Wide Web as an additional source of information within a dialog system has previously been done, with the particular purpose of classifying the topic of a conversation [34]. Their source was Wikipedia, a subset of the information available on the Web. OpenEval handles free-form language (*i.e.*, any written text returned by a search engine). Several other efforts have targeted extracting action plans from the Web. In, e.g., [35] Websites like ehow.com, containing instructions initially intended for human users are used as input. Such work faces challenges similar to the challenges we face in dialog, (*i.e.*, grounding human readable text to robot or agent action frames). Others use the Web as a way to conduct dialog with large numbers of people [36], thereby accumulating common sense knowledge for a mobile service robot. These and additional efforts [37] take written natural language dialog as input, as opposed to spoken natural language, and are therefore not dealing with uncertainties in the speech-to-text engines. Work where truth of semantic relations is evaluated by extracting HTML tables from the Web exists as well [38]. But only using formatted tables, as opposed to free-form language, again limits the form and variety of input information. Other approaches, (e.g., KnowItAll [39], NELL [40], ReVerb [41], OLLIE [42] and CycL [43]) have been able to extract knowledge from unstructured, free-form, Web pages but have not been applied within strongly time-constrained human-robot dialog systems. Of particular closeness to our work, is the processing of single-user spoken commands for execution by a mobile service robot, but from access to structured sources of information on the Web, and represented as deterministic, instantiated robot

action plans [44,45]. Finally in [46] the Web is used to build a common-sense object locality (CSOL) Knowledge Base to guide a robotic visual search task.

3. KnoWDiaL

KnoWDiaL consists of five components, interacting with the user via third-party speech recognition and text-to-speech engines (Figure 2).

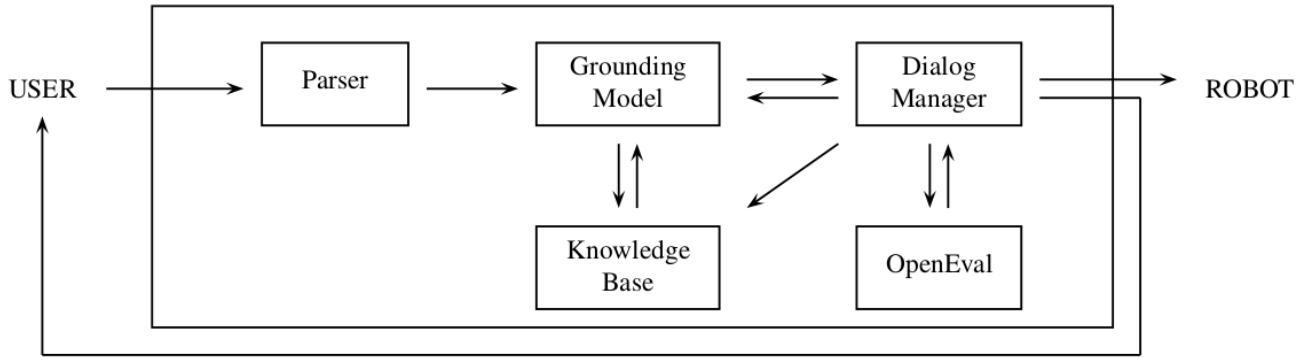


Figure 2. Schematic overview of KnoWDiaL with its five components.

The speech recognizer returns a set of possible interpretations; these interpretations are the input for the first component of KnoWDiaL, a frame-semantic parser. The parser labels the list of speech-to-text candidates and stores them in pre-defined frame elements, like action references, locations, objects or people.

The second component of KnoWDiaL is a Knowledge Base storing groundings of commands encountered in previous dialog. A grounding is simply a probabilistic mapping of a specific frame element obtained from the frame-semantic parser to locations in the building or tasks the robot can perform.

The Grounding Model, the third component of KnoWDiaL, uses the information stored in the Knowledge Base to infer the correct action to take when a command is received. Sometimes, not all of the parameters required to ground a spoken command are available in the Knowledge Base. When this happens, the Grounding Model resorts to OpenEval, the fourth component of KnoWDiaL. OpenEval is able to extract information from the World Wide Web, to fill missing parameters of the Grounding Model.

In case a Web search does not provide enough information, the fifth component of KnoWDiaL, the Dialog Manager, engages in dialog with the user, and explicitly asks for the missing parameters. The Dialog Manager also decides when to ask a follow-up question and when to ask for confirmation. When a command is successfully grounded, the Dialog Manager schedules the task in the CoBot planning system and updates the KnoWDiaL Knowledge Base.

Before describing each of the five components in more detail, we first formally introduce our high-level model.

3.1. High-Level Joint-Probabilistic Model

We formalize the problem of understanding natural language commands as inference in a joint probabilistic model over the groundings Γ, a parse P and speech S, given access to a Knowledge Base K and OpenEval O. Our goal is to find the grounding that maximizes the joint probability, as expressed by the following equation:

$$\arg\max_{\Gamma} \quad p(\Gamma, P, S|K, O) \tag{1}$$

This joint model factors into three main probability distributions: A model of speech, a parsing model and a grounding model. Formally:

$$p(\Gamma, P, S|K, O) = p(\Gamma|P, K, O) \times p(P|S) \times p(S) \tag{2}$$

In our system, the probability of the speech model $p(S)$ is given by a third-party speech-to-text engine. The factors for parsing and grounding will be derived in the upcoming sections.

3.2. Frame-Semantic Parser

The speech recognizer returns a set $\mathcal{S} = [S_1, ..., S_n]$ of speech-to-text candidates and a confidence for each of them $\mathcal{C}_{S_i}$; the first step of KnoWDiaL is to parse them. To train our parser we collected a corpus of approximately 150 commands from people within our group, read these out loud for the speech recognizer and annotated the resulting speech to text candidates by hand; Figure 3 shows a small sample of the annotations used. Labels we use are *action*, *toLocation*, *fromLocation*, *toPerson*, *fromPerson*, *robot*, *objectHere* and *objectElse*. Words we label as *action* are the words used to refer to the tasks the robot can execute (e.g., "go", "bring" or "please deliver"). Locations, both *toLocation* and *fromLocation*, include expressions like "classroom" or "printer room". People, labeled as *toPerson* or *fromPerson*, include expressions like "Tom" or "receptionist". Examples of object references are "cookies" or "tablet pc" and are labeled as *objectHere* or *objectElse*. With *robot* we label parts of the command that refer to the robot itself. Words that are supposed to be ignored are labeled with an additional label *none*.

Commands in Corpus and Their Annotation

- [Go]$_{action}$ to the [bridge]$_{toLocation}$ [CoBot]$_{robot}$
- Could [you]$_{robot}$ [take]$_{action}$ a [screwdriver]$_{objectElse}$ to the [lab]$_{toLocation}$
- [Go]$_{action}$ to [Dana's]$_{toPerson}$ [office]$_{toLocation}$
- Please [bring]$_{action}$ a [pencil]$_{objectElse}$ from the [lab]$_{fromLocation}$ to the [meeting room]$_{toLocation}$
- [Return]$_{action}$ these [documents]$_{objectHere}$ to [Diane]$_{toPerson}$
- Please [get]$_{action}$ [me]$_{toPerson}$ some [coffee]$_{objectElse}$ from the [kitchen]$_{fromLocation}$

Figure 3. A corpus of approximately 150 go to location and transport object commands is annotated by hand. Separate labels are used to distinguish whether an object can be found at the current location or elsewhere. After learning, our method is able to properly recognize the action type of these commands and extract the required parameters.

Labeling tasks often require a much bigger training set [25], but 150 commands proved to be enough to train our parser. There are three main reasons for this: First of all, for each command we get multiple, slightly different, speech-to-text candidates (typically 5 to 10), resulting in an increase in the effective size of the corpus. Second, our set of labels is relatively small. Third, the language used to give commands to our robot is limited by the tasks the robot is able to execute, transporting objects and going to a location.

If $l_i \in \{$*action*, *fromLocation*, *toLocation*, *fromPerson*, ...$\}$ is the label of the i-th word in a speech-to-text candidate S, and this candidate contains N words s_i, then the parsing model is represented as a function of pre-learned weights w and observed features:

$$p(P|S) \triangleq p(l_1 \dots l_N | s_1 \dots s_K) \tag{3}$$

$$= \frac{1}{Z(S)} \exp\left(\sum_i^N w \cdot \phi(l_i, s_{i-1}, s_i, s_{i+1})\right) \tag{4}$$

where $Z(S)$ is a normalization factor and 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. The weights w for each combination of a feature with a label were learned from the corpus mentioned before. We learned them as a Conditional Random Field (CRF) and used gradient descent (LBFGS) as our method to optimize.

After labeling all of the words in each of the speech interpretations in $\mathcal{S}$, we want to extract a frame from them. In order to do so for each $S \in \mathcal{S}$, we greedily group together words with the same label. The output of the semantic-frame parser is therefore a set of parses $\mathcal{P} = [P_1, ..., P_n]$, one for each of the speech interpretations in $\mathcal{S}$; each of the parses P_i consists of labeled chunks together with an overall confidence score C_{P_i}.

3.3. Knowledge Base

In the Knowledge Base of KnoWDiaL, facts are stored by using five different predicates. Four of them are used to store previously user-confirmed groundings of labeled chunks obtained from the semantic-frame parser, while the fifth is used when querying OpenEval. The rest of this section describes each of the predicates in detail.

The predicate *actionGroundsTo* stores mappings between references to actions and the corresponding tasks for the robot. Our robot can execute two tasks, *GoTo* and *BringObject*, and all of the actions are grounded to one of them. Examples of this type of predicate are *actionGroundsTo('take', BringObject)* and *actionGroundsTo('get to', GoTo)*.

The two predicates *personGroundsTo* and *locationGroundsTo* have very similar functions, as they both map expressions referring to people or locations to places the robot can navigate to. The map CoBot uses to navigate in the building has each room labeled with a four digit number; these labels are used as groundings for both of the predicates, as in *locationsGroundsTo('small-size lab', 7412)* or *personGroundsTo('Alex', 7004)*. While the functions of the two predicates are similar, they convey slightly different information: *LocationGroundsTo* saves the way people refer to specific rooms while *personGroundsTo* is intended to store information about where the robot is likely to find a specific person.

The fourth predicate storing information about grounding is *objectGroundsTo*. Similarly to the *personGroundsTo*, this predicate stores information about where the robot is likely to find a specific object. As for the two previous predicates, objects are grounded to room numbers. Examples are *objectGroundsTo("screwdriver", 7412)* or *objectGroundsTo("marker", 7002)*.

To each of the four grounding predicates in the Knowledge Base, a number is attached to keep track of how many times an expression e has been mapped to a grounding γ. From now on, we will refer to this number by using a dotted notation, such as *locationGroundsTo(e, γ).count* or simply as *count*; the updates of this value are explained through detailed examples in Section 4.

Finally, the last predicate used in KnoWDiaL is *locationWebFitness*. The Knowledge Base contains one instance of this predicate for each *locationGroundsTo* element. The goal of this predicate is to store how useful each expression referring to a location is, when querying the Web using OpenEval (more details in Section 3.5); to do so, we associate a score between 0 and 1 to each expression. Examples of this predicate are *locationWebFitness("the elevator", 0.890825)* and *locationWebFitness("elevated", 0.375724)*.

3.4. Grounding Model

In KnoWDiaL, all of the tasks the robot can execute are represented by semantic frames. A semantic frame is composed by an action a, invoking the frame, and a set of arguments e; therefore, *grounding* a spoken command corresponds to identifying the correct frame and retrieving all of its argument; Figure 4 shows two examples of semantic frames and their arguments.

Frame:	*GoTo*
- Parameters:	destination

Frame:	*BringObject*
- Parameters:	object,
	source,
	destination

Figure 4. Semantic frames of the two tasks our robot, CoBot, is able to execute.

We make the assumption that the action a and the arguments e of a frame can be grounded separately. The chunks returned by the frame-semantic parser correspond either to an action a or to one of the arguments e. Therefore, in order to compute $p(\Gamma|P, K, O)$ in Equation (2), we need to find the most likely grounding γ first for the action and then for each of its arguments. The general formula used to compute the likelihood of a grounding is the following:

$$p(\gamma_*|F; \mathrm{K}) = \frac{\sum_i C_{S_i} \cdot C_{P_i} \cdot \mathit{groundsTo}(\mathit{chunk}_i, \gamma_*).\mathit{count}}{\sum_{i,j} C_{S_i} \cdot C_{P_i} \cdot groundsTo(\mathit{chunk}_i, \gamma_j).\mathit{count}} \tag{5}$$

where γ_* is a specific grounding, C_{S_i} and C_{P_i} are respectively the confidence of the speech recognizer and of the parser, i ranges over the set of parses $\mathcal{P}$, j ranges over all of the different groundings for the frame element being considered (*i.e.*, the action a or one of the parameters e), and *groundsTo* is one of the predicates in the Knowledge Base, namely *actionGroundsTo*, *locationGroundsTo*, *personGroundsTo* or *objectGroundsTo*. The *chunks* used to compute Equation (5) are the ones matching the frame element

currently being grounded. For instance, if we are trying to ground the action, only the chunks labeled as *action* are considered.

Section 4 explains, with detailed examples, how Equation (5) is used to infer the correct grounding for a command.

3.5. Querying the Web with OpenEval

One of the core features of KnoWDiaL is the ability to autonomously access the Web *to ground* the location to which the robot needs to navigate, based on a request that does not explicitly mention a known location. So far, we have provided this ability for the robot to determine the location of objects. For example, if a user requests "Please, bring me coffee," the robot may not know the location of the object "coffee." KnoWDiaL accesses the Web by using OpenEval [1,2] to determine the possible location(s) of objects corresponding to its map of the building. The detailed description of OpenEval technology is beyond the scope of this work, but we review the general approach of OpenEval by focusing on its interaction with KnoWDiaL.

OpenEval [47] is an information processing approach capable of evaluating the confidence on the truth of any proposition by using the information on the open World Wide Web. Propositions are stated as multi-argument predicate instances. KnoWDiaL uses two predicates when invoking OpenEval to determine the possible and appropriateness of object locations, namely *locationHasObject* and *locationWebFitness*. KnoWDiaL fully autonomously forms the queries, as propositions (instantiated predicates) to OpenEval based on its parsing of a user request. An example of a proposition generated by KnoWDiaL to query OpenEval is *locationHasObject(kitchen, coffee)*, to which OpenEval could return a confidence of 80%, meaning that it computed a confidence of 80% on the truth of the proposition, namely that *kitchen* is a *location* that has the *object coffee*.

OpenEval has a training phase for each predicate, in which it is provided a small number of instances of the predicate, (*i.e.*, propositions), as seed positive examples. The instances that serve as positive examples of one predicate are negative examples of the others [48]. In terms of KnoWDiaL, it is assumed that OpenEval has been trained on the two predicates *locationHasObject* and *locationWebFitness*, which capture significantly different facts; namely, these are locations of objects and the validity of a particular expression referring to a location in terms of its fitness of information on the Web. For example, training instances of these two predicates would be positive examples *locationHasObject(office, desk)*, *locationHasObject(kitchen, refrigerator)*, and positive examples, such as *locationWebFitness(kitchen)*, and *locationWebFitness(office)*.

OpenEval uses the training examples to generate complex feature vectors that are used to train a classifier. Such feature vectors, termed Context-Based Instances (CBIs), are constructed based on the context of the Web pages returned by searching for the proposition in the open Web. In a nutshell, OpenEval composes CBIs from selective word counting (features) of some predefined amount of "neighboring" text to the training examples in the webpages returned by the search. When a new proposition is given to the classifier, OpenEval evaluates its correctness following a similar process to the training one. From the proposition, it generates a search on the web, and it constructs the corresponding CBIs, which are then given to the trained classifier. OpenEval returns a confidence value on the truth of

the new proposition based on several factors, including the number of test instances that are classified as positive, and the trust score of each source of information.

One of the core insights of OpenEval [47], consists of showing that training on CBIs for the seeded examples enables the testing of examples of the same predicate for instances not previously seen, as examples of the same predicate share the "context" in which they are referred to in the web. Basically, we expect that objects physically present in a location will frequently be found together on the web. For example, one of the top search results for the object "paper" and the location "printer room" is, "There is no more *paper* in the *printer room*, where can I find some more?" For objects unrelated to the location, such as "papers" and "elevator" there are fewer pages which often describe less sensical events such as, "Call for *Papers*, The International Space *Elevator* Consortium (ISEC) invites you to join us in Washington State." Therefore, we expect that the word patterns for related terms will be predictive, while un-related terms will be less predictive.

In terms of the two predicates of KnoWDiaL, the hope is that OpenEval returns high confidence on valid pairs of objects and locations, as well as on appropriate locations themselves. As an example, OpenEval returns high and low confidence, respectively, on queries about locations "kitchen" and "kitten," where this latter could have been incorrectly generated by a speech interpreter. KnoWDiaL uses the confidence values returned by OpenEval in two core ways: To decide whether to ask for further information from the user, and to update its Knowledge Base on grounding information. We provide details on the Knowledge Base updates in our Examples Section below.

3.6. Dialog Manager

Our Dialog Manager uses each of the modules described in the previous sections to interpret commands and to come up with a single complete grounding. The Dialog Manager takes as input the speech-to-text candidates $\mathcal{S}$ and tries to ground the command received to one of the tasks the robot can execute and all of its parameters by using the Knowledge Base. If some of the groundings cannot be retrieved from the Knowledge Base, the Dialog Manager tries to fill the missing fields either by asking specific questions or by querying the Web via OpenEval. Once all of the groundings have been retrieved, the Dialog Manager asks for confirmation, updates the Knowledge Base and schedules the task. Algorithm 1 shows all of the steps described.

Algorithm 1 dialog_manager(S)

$F \leftarrow$parse_and_frame(S)
$\Gamma \leftarrow$ground(F)
$\Gamma^* \leftarrow$fill_missing_fields(Γ)
ask_confirmation(Γ^*)
update_knowledge_base(Γ^*, F)
schedule_task(Γ^*)

Figure 5 shows a typical dialog for a user who asks CoBot to deliver a pencil to the meeting room. The speech recognizer returns three text-to-speech candidates (Figure 5b). These are parsed (Figure 5c) and then grounded by using the Knowledge Base. Given the current Knowledge Base, not shown here,

KnoWDiaL is able to ground the action required by the command to *BringObject*, that the object to be delivered is *pencil* and that it should be delivered to room 7502. The information missing to completely ground the command is where the object can be found (Figure 5d); to retrieve it, KnoWDiaL queries the Web and, after receiving confirmation from the user (Figure 5e), executes the task.

USER: "Go deliver a pencil to the meeting room."

(a) Spoken Command

$s_1 =$ "go deliver a pencil to the meeting rooms"
$s_2 =$ "good liver pen still in the meeting room"
$s_3 =$ "go to live for pencil to the meaning room"

(b) Speech to Text Candidates

$$f_1 = [\text{go deliver}]_{\text{act}}\ [\text{pencil}]_{\text{objElse}}\ [\text{meeting rooms}]_{\text{toLoc}}$$
$$f_2 = [\text{good liver}]_{\text{act}}\ [\text{pen still}]_{\text{objHere}},\ [\text{meeting room}]_{\text{toLoc}}$$
$$f_3 = [\text{go}]_{\text{act}}\ [\text{live}]_{\text{toPer}}\ [\text{pencil meaning room}]_{\text{toLoc}}$$

(c) Parsing and Framing

$$\Gamma = [a = \text{BringObj}, e_{\text{source}} = \text{NULL}, e_{\text{destination}} = 7502, e_{\text{obj}} = \text{pencil}]$$

(d) Action / Parameter Grounding

CoBot: "You want me to deliver the object pencil, is that correct?"
USER: "Yes"
CoBot: "I am querying the web to determine where the object pencil is located."

Location-references with high count and web fitness are extracted from the Knowledge Base: "kitchen," "office," and "lab." OpenEval is evaluated for these expressions.

$$\Gamma^* = [a = \text{BringObj}, e_{\text{source}} = 7602, e_{\text{destination}} = 7502, e_{\text{obj}} = \text{pencil}]$$

CoBot: "I have found that object *pencil* is most likely to be found in location *office*. I am going to get object *pencil* from location *office* and will deliver it to *room 7502*, is that correct?"
USER: "Yes"

(e) Dialog

Figure 5. A dialog example. Given the command the robot is able to ground, using its Knowledge Base, the action, and two of the frame parameters ("*destination*" and "*object*"). The third parameter needed to completely ground the command ("*source*") is grounded using OpenEval.

Additionally, in order to make KnoWDiaL easier to use, we added to the Dialog Manager a small set of keywords. If the words "cancel" or "stop" are detected, the current dialog is canceled, and the user can give a new command to the robot. If the words "wrong action" are recognized, KnoWDiaL asks explicitly for the task it needs to perform and then resume its normal execution. The Dialog Manager also recognizes keywords, such as "me" and "this", to handle commands involving the current location as one of the action parameters (e.g., "Bring me a cookie" or "Bring this paper to the lab"). In this case, a temporary location *here* is created to ground the location and, during execution, is converted to the room nearest to the position where the robot received the command.

4. KnoWDiaL: Detailed Examples

We now illustrate the complete KnoWDiaL approach with two examples to show the details of the accesses and computations underlying the updates to the Knowledge Base. In the two examples, the new groundings come from the dialog with the human, and from the access to the Web, respectively.

4.1. Example 1: Accessing and Updating the Knowledge Base from Dialog

When receiving a spoken command, in this example the sentence *"Go to the small-size lab"*, the first step is to process the audio input and get a set of multiple transcriptions from the ASR. The speech recognizer used (for our experiments we used the Google ASR) returns an ordered set of interpretations, $\mathcal{S} = [S_1, ..., S_n]$, but only provides a confidence score, C_{S_1}, for the first one. In order to obtain a confidence score for each of the interpretations we used the following formula:

$$C_{S_i} = max(C_{S_1} - \alpha \cdot C_{S_1} \cdot i, \alpha \cdot C_{S_1}) \tag{6}$$

where i is the rank of each interpretation and α is a discount factor; Figure 6a shows all of the transcriptions obtained from the Automated Speech Recognition (ASR) together with the confidence scores computed.

Next, each transcription is parsed. The result of this step is shown in Figure 6b and consists of a set of parses $\mathcal{P} = [P_1, ..., P_n]$ where each parse P_i contains a set of labeled chunks together with a confidence score, C_{P_i} for the whole sentence.

Since the goal is to completely fill a semantic frame representing one of the tasks the robot can perform, we first need to identify the frame invoked by the command received that is to *ground* the action of the command. To do so, we query the Knowledge Base (KB) for all the *labelsGroundTo* predicates whose first argument matches any of the label sequence in $\mathcal{P}$ and for all the *actionGroundsTo* predicate whose first argument matches any of the chunks labeled as *action* in $[P_1, ...P_n]$. This query returns a set of j possible groundings γ, in this example [*GoTo, BringObject*]. To select the correct grounding, we use Equation (5) for the *actionGroundsTo* predicate and compute the probability for each of the j groundings returned. We select the grounding with the highest value as correct, which in this example, is the task *GoTo* with a probability of 0.994 .

Once the action has been grounded, the corresponding semantic frame shows which parameters are needed. For the *GoTo* frame, the only parameter needed is the destination. To ground the destination, we query the KB for all the *locationGroundsTo* predicates, whose first argument match the chunks labeled as *toLocation*. Similarly to what happened for the action, j possible groundings are returned, and for each of them, we compute its probability by using Equation (5) for the *locationGroundsTo* predicate. The one with the highest probability is then selected as the final grounding; in our example, room 7412 is selected with probability 1 as it is the only *locationGroundsTo* predicate available.

At this point, the semantic frame representing the command received has been completely filled. Before executing the corresponding task, KnoWDiaL engages in a short dialog with the user, and if everything is confirmed, lets the robot execute the task.

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

$[\text{Go to}]_{action}$ $[\text{the small size lav}]_{toLocation}$	0.8
$[\text{go 2}]_{action}$ $[\text{small sized lab}]_{toLocation}$	0.1
$[\text{goto}]_{action}$ $[\text{the small size lab}]_{toLocation}$	0.3
$[\text{get}]_{action}$ $[\text{the small sized love}]_{objectHere}$	0.7

(b) Parses

actionGroundsTo('go to', GoTo)	5.0
actionGroundsTo('goto', GoTo)	2.3
actionGroundsTo('goto', BringObject)	0.3
actionGroundsTo('get', BringObject)	2.15
locationGroundsTo('the small size lab', 7412)	7.9

(c) Initial Knowledge Base

Figure 6. Multiple transcriptions (**a**) and parses (**b**) for the command "Go to the small-size lab". (**c**) Shows the initial Knowledge Base with the count for each predicate.

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 7. Update KB with in blue the predicates that have been added or updated.

Finally, while the robot is executing the task KnoWDiaL updates its KB; for each of the chunks of all of the parses in $\mathcal{P}$, the count of the corresponding predicate is increased by $C_{S_i} \cdot C_{P_i}$; in particular, the chunks labeled as *action* increase the count of the *actionGroundsTo* predicate, while the chunks labeled as *toLocation* increase the count of the *locationGroundsTo* predicate. If, for any of the predicates, an

instance is not already present in the KB a new one is simply added. Figure 7 shows the updated KB after the command has been executed.

4.2. Example 2: Accessing and Updating the Knowledge Base from Accessing the Web

For this second example, we will consider the command *"Bring coffee to the lab"*. Similarly to what happened in Example 1, the audio input is processed by the ASR, and its output is parsed. The result, a set of speech interpretations $\mathcal{S}$ and a set of parses $\mathcal{P}$, are shown together with the initial KB in Figure 8.

`Briggs coffee to the lav`	`0.85`
`bring the cofee to the lab`	`0.425`
`Rings coffe the lab`	`0.2125`

(a) Speech recognition results

$[\texttt{Briggs}]_{action}$ $[\texttt{coffee}]_{objectElse}$ $[\texttt{to the lav}]_{toLocation}$	`0.23`
$[\texttt{bring}]_{action}$ $[\texttt{the cofee}]_{objectElse}$ $[\texttt{to the lab}]_{toLocation}$	`0.88`
$[\texttt{Rings}]_{action}$ $[\texttt{coffe}]_{objectElse}$ $[\texttt{the lab}]_{toLocation}$	`0.18`

(b) Parses

```
actionGroundsTo('bring', BringObj)                    2.1
actionGroundsTo('rings', BringObj)                    0.3

locationGroundsTo('to the lab', 7412)                 4.3
locationGroundsTo('kitchen', 7602)                    1.62
locationGroundsTo('kitcheen', 7602)                   0.34
locationGroundsTo('kitchenette', 7602)                1.35
locationGroundsTo('office', 7004)                     2.8

locationWebFitness('office', 0.98)
locationWebFitness('kitchen', 0.92)
locationWebFitness('kitcheen', 0.34)
locationWebFitness('kitchenette', 0.93)
locationWebFitness('to the lab', 0.88)
```

(c) Initial Knowledge Base

Figure 8. Multiple transcriptions (**a**) and parse (**b**) for the command "Bring coffee to the lab". (**c**) Shows the initial Knowledge Base with the count for each predicate.

Again, the first step is to ground the action of the command, i.e., to identify the corresponding semantic frame. To do so, we query the KB for *actionGroundsTo* predicates and then use Equation (5)

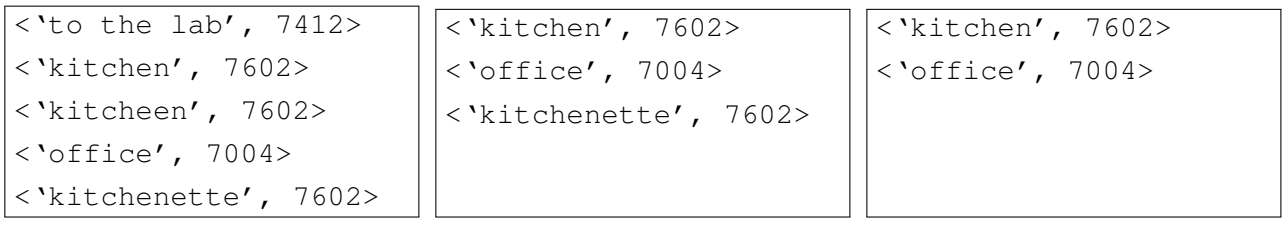
to compute the most likely action corresponding to the command received. Given the KB for this second example the only action matching is also the only action in the KB and is therefore selected with a probability of 1.

Having grounded the action, means the semantic frame corresponding to the command has been identified; the next step is therefore to ground all of the parameters of the frame. For the *BringObject* frame, we need three parameters: the object, the location where it can be found, and the location where it has to be delivered. First, we check for the object. That is, we see if, in any of the $\mathcal{P}_i$ parses, we can find a chunk labeled as *objectHere* or *objectElse*. KnoWDiaL simply selects as the object the first chunk whose combined speech and parse confidence is greater than a given threshold, $C_i \cdot C_{P_i} \geq \tau$ with $\tau = 0.15$. In our example the chunk selected to represent the object is *"coffee"*.

Next, we need to figure out where the object can be found. To do so, we first check if the command explicitly mentions it, and we see if, in any of the parses in $\mathcal{P}$, we can find a chunk labeled as *fromLocation*. If this is the case, for each *fromLocation* chunk i we query the KB for a matching *locationGroundsTo* predicate. This operation returns a set of j possible groundings γ and, again, to compute the more likely we use Equation (5).

If the location from where the object has to be retrieved is not explicitly mentioned, we query the KB for all of the *objectGroundsTo* predicates, whose first argument matches any of the *objectElse* chunks and compute the more likely grounding applying Equation (5) to the *objectGroundsTo* predicate.

Unfortunately, in our example there is no chunk labeled as *fromLocation* in $\mathcal{P}$ and no *objectGroundsTo* predicate in the KB. When this happens, to figure out where we can find the object, we resort to OpenEval. In order to query OpenEval, we need a specific object, which we have already identified as *"the coffee"* and a set of possible locations $\mathcal{L}$. To build the set $\mathcal{L}$, we first query the KB for all of the *locationGroundsTo* predicates and add their first argument to $\mathcal{L}$. Next, we make a second query to the KB and filter out all of the elements in $\mathcal{L}$ having a *locationWebFitness* score below a threshold of 0.9. Finally, we make sure that all of the elements left in $\mathcal{L}$ refer to different physical locations by checking the groundings of the *locationGroundsTo* and selecting, among the ones referring to the same room number, the reference with the highest count. This whole process, for the KB of our example, is shown in Figure 9.



(a) Initial $\mathcal{L}$ **(b)** $\mathcal{L}$ after checking for `webFitness` **(c)** Final $\mathcal{L}$

Figure 9. The set of possible locations used to query OpenEval.

Querying OpenEval returns a score, C_O, for finding *"the coffee"* in each of the locations in $\mathcal{L}$; KnoWDiaL then asks the user to select the correct location between those with a score above 0.8. In our example, out of the two locations in $\mathcal{L}$, only *kitchen* is above the threshold and the user is simply asked to confirm if it is the correct location.

Finally, we need to find out where the object needs to be delivered. To do so, we look in all of the parses in $\mathcal{P}$ if there are chunks labeled as *toLocation*. If we find at least one such chunk, we query the KB for all of the matching *locationGroundsTo* predicates and compute the more likely grounding by using Equation (5). In our example, we find multiple chunks labeled as *toLocation* but only one *locationGroundsTo* predicate matching them. Therefore, the grounding 7412 is selected with a score of 1. In general, if there is no *toLocation* chunk or the grounding cannot be retrieved from the KB, KnoWDiaL engages in a short dialog with the user asking for the location explicitly.

Now that the semantic frame has been completely filled, the robot can execute the corresponding task. While this happens, KnoWDiaL updates its KB; as in the previous example, for each chunk, the corresponding predicate is increased by the combined score of the speech recognizer and the parse. Moreover, a new predicate, *objectGroundsTo* is added to the KB with, as a first argument, the object used to query OpenEval, *"the coffee"*; as second argument, the grounding associated to the location with highest C_O score and as *count* the product of the three confidences $C_O \cdot C_{S_i} \cdot C_{P_i}$.

5. Results

The performance of our system is determined by the performance of each of the five subsystems. Therefore, in this section, we describe three experiments, gradually increasing the number of subsystems involved. First, we describe an experiment involving *GoTo* commands only. Performance is measured in terms of the amount of required interactions (*i.e.*, number of questions our participants need to answer), and we compare the ways people refer to different types of locations. Second, we show that, given a set of location references, OpenEval generally comes up with a reliable clue to look for a certain object. Hereafter, another dialog experiment is conducted, where the dialog system as a whole is tested, involving both *GoTo* and *BringObj* commands. Finally, we implement our dialog system on CoBot itself.

5.1. Learning Location References

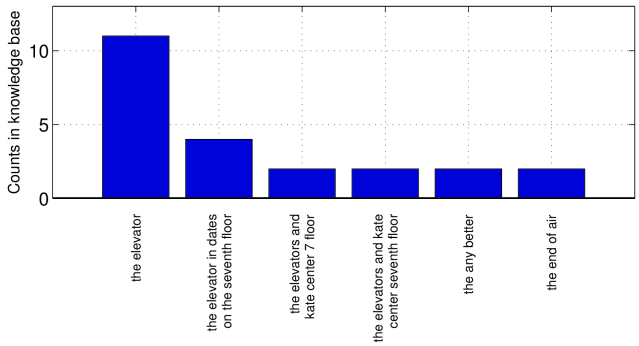
Nine different people were asked to command the robot for about ten minutes, sending it to different locations on its map (in simulation). The subjects were both native and non-native English speakers, which made speech-to-text more challenging. Although the task itself was fixed, people could use the language that was natural to them. In order to prevent priming our participants with location references, we used a printed map of the building CoBot operates in. Six locations were marked on this map and annotated with a room number. Our aim here was to test the ability of our algorithm to learn referring expressions for different locations through dialog alone. Therefore the initial Knowledge Base was empty.

A total number of 57 "go to" location commands were given by our subjects. These contributed to a Knowledge Base with 177 predicates, grounding either a location reference or phrases referring to a person. We compared our algorithm to a non-learning baseline that directly asks to spell the room number of the robot's destination. Measured over the entire experiment, our method requires only 79% of the questions that the baseline system would need. Since this is a percentage measured over the entire experiment, it does not guarantee that we will outperform the baseline system in each specific dialog. A particularly bad scenario would be a case in which we find a strong grounding, ask for confirmation,

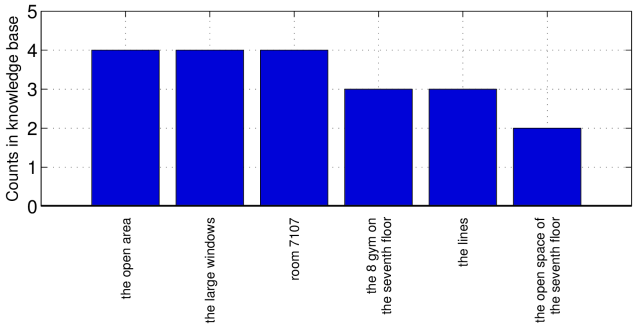
but turn out to have found the wrong grounding. This worst-case scenario involves one extra question, compared to the baseline. Given that on average, we clearly outperform the baseline system, we consider this to be acceptable.

Entropy provides a good measure to evaluate whether or not different people refer to a location by using very different expressions. When calculating statistical entropy over the set of referring expressions for a specific location, we find the lowest entropy (2.8) for the location "the elevator". The highest entropy (3.3) was found for "the meeting room" and "the atrium". For the latter location, people were using references like "the open area", or "the place with the large windows". On average, entropy was 3.0.

For the lowest and highest entropy location, the Knowledge Base count corresponding to each reference is plotted (Figure 10a,b). Since speech-to-text is not always perfect, "the atrium" was often translated into "the 8 gym". Our dialog system does not see a difference between the two translations and will learn to understand commands involving an inaccurate speech-to-text translation just as quick as ones involving the right translations. As long as speech-to-text is consistent, it does not have to be perfect to allow our dialog system to learn.



(a) Low-entropy location "the elevator"



(b) High-entropy location "the atrium"

Figure 10. For two different locations, the six location references that were most frequently encountered are shown.

5.2. Predicting Object Locations with OpenEval

To measure the accuracy of OpenEval, first, we have to train it. We have collected a corpus of 1350 unique instances for predicate $locationHasObject$ and 250 unique instances for predicate $locationWebFitness$. Instances for the has-object predicate were acquired by asking subjects on Amazon's Mechanical Turk to name a set of objects for 40 different location types; using Amazon's mechanical Turk made it easy to collect a large number of such instances. On the other hand, instances of the $locationWebFitness$ predicate were acquired by manually extracting a set of expressions that were used to refer to locations in previous dialogs that people had with our robots, and the set available for testing was therefore more limited. The data is split by randomly choosing 80% of data for training and 20% for testing. OpenEval was trained by using the first 50 Web pages that are returned by a search engine.

Figure 11 shows the results by using precision and recall measures over the test data. As a baseline, we show the results of Pointwise Mutual Information (PMI), which is a predicate evaluation method solely based on the number of search engine hits. The PMI score of $locationHasObject(e, \gamma)$ equals the normalized number of hits when searching for e and γ together, divided by a multiplication of the normalized number of hits e and γ have individually.

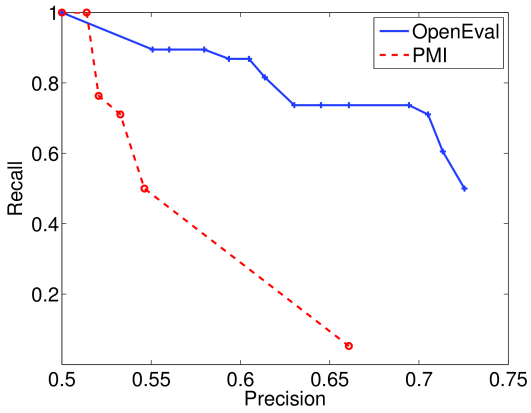


Figure 11. Precision/recall curve for the $locationHasObject$ predicate (270 test instances). OpenEval is compared to PMI, which is a web-based predicate evaluation method solely based on the number of search engine hits.

Table 1 shows the results for a subset of the test objects for four location types in our test environment. OpenEval is able to correctly determine the most likely location for most objects. OpenEval chooses "bathroom" as the most likely location for "cup". Although this is correct in some environments (e.g., hotels), we generally expect to find "cup" in either a "kitchen" or an "office". The results show that by requesting more specific queries such as "coffee cup", OpenEval will change its classification to location "kitchen".

Table 1. The probability that OpenEval assigns to different test objects for each location type. The location type with maximum probability is shown as bold.

Object	**Location Types**			
	Bathroom	Printer Room	Kitchen	Office
coffee	0.08	0.02	**0.72**	0.18
marker	0.33	**0.53**	0.08	0.06
pen	0.15	0.27	0.23	**0.35**
toner	0.05	**0.87**	0.02	0.06
scissors	0.26	0.01	**0.61**	0.12
whiteout	**0.66**	0.02	0.24	0.08
laptop	0.1	**0.48**	0.08	0.34
paper	0	0.17	0.13	**0.7**
cup	**0.42**	0.1	0.36	0.12
coffee cup	0	0.01	**0.73**	0.27
speakers	0.34	0.06	0.25	**0.35**

5.3. Learning Object Groundings

A final experiment involving ten people, six of them non-native English speakers, has been conducted. Our aim here is to test the dialog system as a whole, including the grounding of objects. Participants were provided with a map that had seven marked locations, annotated with room numbers. As opposed to our initial experiment, not all participants were familiar with the building this time. Therefore, we also provided a suggested expression for each of the marked locations on the map. Participants were free to use this suggested expression or any other synonym or different expression that they found to be natural. Also, they were shown a sheet with forty pictures of objects that our robot would be able to transport. We chose pictures as opposed to a list of words to prevent priming our participants with a way to refer to specific objects.

Each of the participants was asked to command the robot through its speech interface for about fifteen minutes. The participants were free to choose whether they would ask the robot to transfer one of the objects, or to simply send it to a specific location. A *BringObj* command could involve asking the robot to deliver an object provided by the user to any of the locations on the map or to ask it to deliver an object that first has to be collected at some other place. In the latter case, the *source* could either be explicitly provided in the command ("bring me a cookie from the kitchen") or not be specified by the user ("bring me a cookie"). In case the *fromLocation* is not explicitly provided, the robot had to come up with a reasonable place to look for the object. It could do so either by doing inference over the $objectGroundsTo$ predicates in its Knowledge Base (implicit grounding of the *fromLocation*) or by using the expression for the object to query OpenEval.

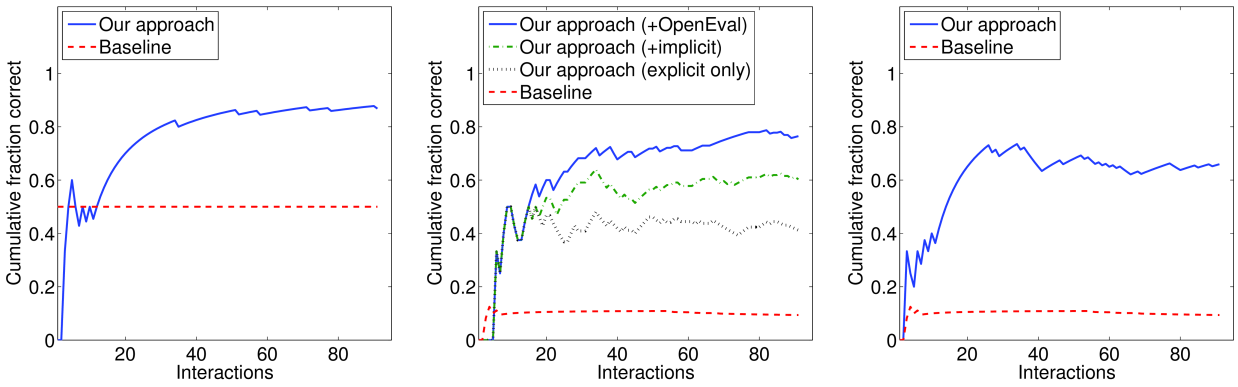
The baseline that we are comparing our system to, is a dialog manager that simply asks which action it should take, followed by a question for each of the parameters needed to execute this action. In case of a *BringObject* command, as shown in Figure 4, three parameters are necessary: (i) the source of the object (*i.e.*, the location where it can be found), (ii) its destination and, since the robot should ask somebody to put the object in its basket at the source, (iii) a reference to the object itself. Therefore, the baseline for a *BringObject* command is four questions, while the baseline for *GoTo* location commands is two.

We started the experiment with an entirely empty Knowledge Base. After a total of 91 speech-commands, our system had asked only 67% of the number of questions the baseline system would have asked. Only in 12% of the commands our system posed more questions than the baseline system would have done (worst-case was three additional questions).

In order to explain this result in more detail, we take a closer look at each of the elements that we are grounding. Most of the learning with respect to grounding action types takes place during the commands provided by the first three people in the experiment (Figure 12a). Apparently their way of invoking an action, generalizes easily. In the remainder of the experiment, starting at command 33, the wrong action type was recognized only six times.

Roughly two thirds of the commands in this experiment were *BringObject* commands; the others were *GoTo* location commands. In case of a *BringObject* command, our subjects chose not to provide the from-location 31 times; out of these, for 19 times, the expression for the object could not be grounded directly from the Knowledge Base. OpenEval was able to come up with a correct object location 11 times (Figure 12b). This was done either by returning a single correct location (2 times), by asking the user to

choose out of two high-probability locations (7 times), or by offering three high-probability locations (2 times). As can be seen in the graph, it takes some commands before OpenEval becomes useful, which is because first some location references with web fitness scores need to be in the Knowledge Base.



(a) Action recognition **(b)** FromLocation **(c)** toLocation

Figure 12. Grounding performance during experiment involving 91 go to location and transport object tasks. The baseline for these graphs consists of the percentage we would be able to guess correct by picking a random action type, or a random location from our Knowledge Base. A *fromLocation* can be found in three ways: By grounding a location reference provided by the user (explicit grounding) by grounding the object reference to a location (implicit grounding) or by using OpenEval.

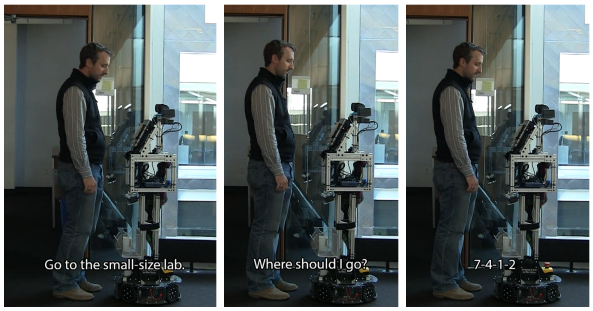
At the end of the experiment, 41% of the *fromLocations* are found by grounding a location reference provided by the user. Taking into account that the user did not always explicitly provide this parameter, 75% of the provided *fromLocation* references are grounded correctly, slightly better than what has been achieved when grounding the toLocation (Figure 12c).

Average entropy calculated over the referring expressions for each of the locations in the Knowledge Base was 3.0, which is equal to what we obtained in our first experiment. Therefore, we conclude the suggested location reference did not lead to a smaller spread in referring expressions that participants were using for locations.

Measured over the entire experiment, 72% of the object expressions in *BringObject* commands were extracted correctly. Since, in our system extracting the object reference from a speech to text candidate does not require a Knowledge Base, this percentage remained constant throughout the experiment.

5.4. Demonstration on CoBot

In this section, we demonstrate the different kinds of interaction CoBot can have, depending on the task assigned and the state of the Knowledge Base. First, we show one of the first interactions we had with CoBot (Figure 13). The Knowledge Base at this stage is still relatively empty and allows to correctly ground the action but not the location. Therefore, the robot asks a question to the user, updates the Knowledge Base, and then correctly executes the command.



User:	Go to the small-size lab.
Cobot:	Where should I go?
User:	The small-size lab.
Cobot:	I don't know that location, please give me a four-digit number
User:	7-4-1-2
Cobot:	Going to the lab, room 7412

Figure 13. One of the first interactions with CoBot, when the Knowledge Base is in an early stage.

As the Knowledge Base grows, less additional questions are asked by the dialog system. However, sometimes, the stored information does not match the intention of the user. Such a situation is shown in Figure 14. The user asks for the "conference room". KnoWDiaL grounds this expression to room 7101, but while room 7101 indeed is a conference room, the user is referring to a different one, room 7501. Through a short dialog, the user is able to correct CoBot, and after the new information is added to the Knowledge Base, the task is executed. Notice how after such an interaction, the Knowledge Base has now multiple groundings for the same expression with different weights; this does not represent a problem as KnoWDiaL always selects the more likely grounding.

User:	Go to the conference room.
Cobot:	I am going to room 7101, is it correct?
User:	No.
Cobot:	Ok, can you please spell the room number of my destination?
User:	7-5-0-1
Cobot:	I am going to room 7501, is it correct?
User:	Yes.

Figure 14. An example of interaction where the user corrects the information retrieved from the Knowledge Base.

Next we demonstrate an interaction in which CoBot queries the Web by using OpenEval (Figure 15). The user asks for "a coffee" but does not specify where this object can be found. As a first attempt, the robot tries to find a match in its Knowledge Base, but nothing is found. Therefore, CoBot queries the Web and comes up with two alternatives: The kitchen or the office. Before executing the task, the robot asks the user to choose one of these two options.

User:	Please bring a coffee to the lab.
CoBot:	The object is not in my Knowledge Base, I will search the web using OpenEval to see if the object is in location “kitchen”, “office”, “meeting room” or “printer room”.
CoBot:	OpenEval returns high probability for object “coffee” in location “office” and also for object “coffee” in location “kitchen”. Which location do you think is the best?
User:	Kitchen.
CoBot:	I will collect the object “coffee” and deliver it at the lab. I am going to the kitchen, room 7602.

Figure 15. CoBot queries the web using OpenEval in order to retrieve the information it needs to execute the required task.

Finally, we show an interaction in which the robot is able to correctly ground every part of the command (Figure 16). The user asks to bring lunch from the kitchen to the lab; CoBot first asks for the object that needs to be delivered, and then asks for confirmation for both of the locations involved in the task. The robot asks for confirmation to make sure it will execute the correct task but it is able to ground “lunch”,“kitchen”, and “lab” autonomously. Once the robot gets to the kitchen, it asks someone for help in order to put the lunch in its basket, and finally it goes to the lab and delivers the lunch.

All of the interactions presented in this section are also available in a video [19].

User1:	Bring lunch from the kitchen to the lab.
CoBot:	Which object should I deliver? Please give me the name of the object only.
User1:	Lunch.
CoBot:	You want me to deliver the object lunch. Is that correct?
User1:	Yes.
CoBot:	I will collect the object lunch at the kitchen, room 7602, and deliver it to the lab, room 7412. Is that correct?
User1:	Yes.
CoBot:	I am now planning my route to specific locations. I am going to the kitchen, room 7602, to pick up the object "lunch". Bye.
CoBot goes to the kitchen	
CoBot:	Hello, can you please put the object "lunch" in my basket and press "done" when I am ready to go?
User2:	**takes lunch from the fridge, puts it into the basket and presses the button**
CoBot:	Thank you, I am going to the lab, room 7412, to deliver the object "lunch"
CoBot goes to the lab	
CoBot:	Hello I have brought the object "lunch" from the kitchen, room 7602, to your room. Please press the button "done" when I can leave.
User3:	**takes the lunch and presses the button**

Figure 16. An interaction between CoBot and multiple users. The robot is able to correctly ground every part of the command.

6. Conclusions

We have presented KnoWDiaL, an approach for a robot to use and learn task-relevant knowledge from human-robot dialog and access to the World Wide Web. We have introduced the underlying joint probabilistic model consisting of a speech model, a parsing model, and a grounding model. We focus on tasks of a mobile service robot, CoBot, involving actions, locations, and objects. Our model is used in a dialog system to learn the correct interpretations of referring expressions the robot was not familiar with beforehand. Commands involving different actions, locations and people can be dealt with by adding new facts to the Knowledge Base and by searching the Web for general knowledge. KnoWDiaL is an integral

part of the continuously operating CoBot robot. We have presented experiments that show the number of questions asked by the robot in order to understand a command decreases, as it interacts with more people, and that our KnoWDiaL approach outperforms a non-learning baseline system. The KnoWDiaL approach is a general contribution to the challenge of speech interpretation and knowledge processing for autonomous robots, which we demonstrate within mobile service robots.

Now that we successfully reached this core KnoWDiaL approach, our ongoing and future works include addressing several upcoming challenges, including handling dynamic knowledge, either temporary, changing, or incorrect knowledge. For example, currently, if Dana changes her office after users confirmed Dana's office to be "room 7008" for some number of times, it would require an equivalent number of interactions about the new office to overrule this knowledge. Different techniques can be investigated, such as time-stamping the Knowledge Base updates, and asking or searching for clues on how confident users are. Confidence scores could be used for non-linear and Bayesian Knowledge Base updates.

We are also interested in searching for additional features that may be useful in the grounding process, including the robot's current position in the building, the time of the day, and features regarding the person who is interacting with the robot. For example, if the robot does not recognize the person, such person is likely to be a visitor, and is more likely to be asking for a *GoTo* command. The nature of our grounding model makes it straightforward to add such features in our model by including their *count* in Equation (5).

Finally, we are actively pursuing the sharing of the learned knowledge base among our multiple robots in the same building, as well as with other robots in other locations. Our approach could also be integrated with other dialog systems. We are committed to investigating this knowledge sharing among robots towards an understanding of common and general representations that can be shared among multiple learning autonomous robots.

Once a grounding is confirmed, all frame elements of each speech interpretation are added to the Knowledge Base (Figure 17), allowing the system to learn, and to generalize knowledge acquired from this dialog to other task assignments that are similar but not the same. For each of the framed actions and parameters, the corresponding $groundsTo$ relations are added to the Knowledge Base and initialized to a count equal to its weight (*i.e.*, speech to text confidence multiplied by parser confidence). If a certain grounding was already present, its count is incremented by the weight.

Waiting for user confirmation before the Knowledge Base is updated does not imply all of this knowledge makes sense for human readers. An example from our Knowledge Base is "bring me", which in some cases incorrectly got parsed as "to-person". In the resulting dialog, the destination of the robot got confirmed being the robots current location, so a predicate $personGroundsTo(bring\ me, here)$ was added to the Knowledge Base. Another example is $locationGroundsTo(kick\ in, 7602)$, where "kick in" is an incorrect translation of "kitchen". By storing the groundings of such frame elements, we are able to learn correct interpretations of mistakes in speech to text and parsing.

labelsGroundTo([act][obj-else][to-loc], BringObj)
labelsGroundTo([act][obj-here][to-loc], BringObj)
labelsGroundTo([act][to-per][to-loc], BringObj)
actionGroundsTo(go deliver,BringObj)
actionGroundsTo(good liver,BringObj)
actionGroundsTo(go,BringObj)
locationGroundsTo(meeting room,7502)
locationGroundsTo(meeting rooms,7502)
locationGroundsTo(pencil meaning room,7502)
personGroundsTo(live,7502)
objectGroundsTo(pencil,7502)
objectGroundsTo(pen still,7502)
locationWebFitness(meeting room,0.58)
locationWebFitness(meeting rooms,0.52)
locationWebFitness(meaning room,0.58)
locationHasObject(kitchen,pencil)
locationHasObject(office,pencil)
locationHasObject(lab,pencil)

Figure 17. Knowledge base updates resulting from the dialog example. Specific weights have been omitted.

Acknowledgments

This work was generously supported by multiple funding sources, including La Sapienza University, Eindhoven University of Technology, Zheijang University, and by the National Science Foundation under awards IIS-1012733 and IIS-1218932. Finally, the work would not have been possible without the remarkable performance of the CoBot robots, for which we thank Joydeep Biswas, Brian Coltin, and Stephanie Rosenthal, for their development of the autonomous service mobile robots, and Mike Licitra for the design and construction of the always-functional and reliable CoBot robot platform.

Author Contributions

The first two authors contributed equally to this work.

Conflicts of Interest

The authors declare no conflict of interest.

References

1. Samadi, M.; Veloso, M.; Blum, M. Evaluating correctness of propositions using the web. In Proceedings of the Workshop on Learning by Reading and its Applications in Intelligent Question-Answering, Barcelona, Spain, 18 July 2011.
2. Samadi, M.; Kollar, T.; Veloso, M.M. Using the web to interactively learn to find objects. In Proceedings of the 26th Conference on Artifical Intelligence (AAAI-12), Toronto, ON, Canada, 22–26 July 2012.
3. Biswas, J.; Veloso, M.M. Localization and navigation of the cobots over long-term deployments. *Int. J. Robot. Res.* **2013**, *32*, 1679–1694.
4. Hanheide, M.; Gretton, C.; Dearden, R.; Hawes, N.; Wyatt, J.; Pronobis, A.; Aydemir, A.; Göbelbecker, M.; Zender, H. Exploiting probabilistic knowledge under uncertain sensing for efficient robot behaviour. In Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence, Barcelona, Spain, 16–22 July 2011; pp. 2442–2449.
5. Porzel, R.; Gurevych, I. Contextual coherence in natural language processing. In *Modeling and Using Context*; Springer: Berlin, Germany, 2003; pp. 272–285.
6. Rosenthal, S.; Biswas, J.; Veloso, M. An effective personal mobile robot agent through symbiotic human-robot interaction. In Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems, Toronto, ON, Canada, 9–14 May 2010; pp. 915–922.).
7. Biswas, J.; Coltin, B.; Veloso, M. Corrective Gradient Refinement for Mobile Robot Localization. In Proceedings of the IROS'11, the IEEE/RSJ International Conference on Intelligent Robots and Systems, San Francisco, CA, USA, 25–30 September 2011.
8. Kollar, T.; Perera, V.; Nardi, D.; Veloso, M. Learning environmental knowledge from task-based human-robot dialog. In Proceedings of the International Conference on Robotics and Automation, Karlsruhe, Geramny, 6–10 May 2013.
9. Jackendoff, R.S. *Semantics and Cognition*; MIT Press: Cambridge, MA, USA, 1983; pp. 161–187.
10. Landau, B.; Jackendoff, R. Whence and whither in spatial language and spatial cognition? *Behav. Brain Sci.*, **1993**, *16*, 255–265.
11. Talmy, L. The fundamental system of spatial schemas in language. In *From Perception to Meaning: Image Schemas in Cognitive Linguistics*; Hamp, B., Ed.; Mouton de Gruyter: Berlin, Geramny, 2005; pp. 199–234.
12. Katz, B. Using English for indexing and retrieving. In *Artificial Intelligence at MIT Expanding Frontiers*; MIT Press: Cambridge, MA, USA, 1988.
13. Wallach, H. *Conditional Random Fields: An Introduction*; Rapport Technique MS-CIS-04-21; Department of Computer and Information Science, University of Pennsylvania: Philadelphia, PA, USA, 2004; Volume 50.
14. Kudo, T. CRF++: Yet Another CRF Toolkit. 2009. Available online: http://sourceforge.net/projects/crfpp/ accessed on 12 June 2015).
15. Bird, S.; Klein, E.; Loper, E. *Natural Language Processing with Python*; O'Reilly: Sebastopol, CA, USA, 2009.

16. Winograd, T. Procedures as a Representation for Data in a Computer Program for Understanding Natural Language. Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 1970.
17. Hsiao, K.; Tellex, S.; Vosoughi, S.; Kubat, R.; Roy, D. Object schemas for grounding language in a responsive robot. *Connect. Sci.* **2008**, *20*, 253–276.
18. MacMahon, M.; Stankiewicz, B.; Kuipers, B. Walk the talk: Connecting language, knowledge, and action in route instructions. In Proceedings of the 21st National Conference on Artificial Intelligence-AAAI, Boston, MA, USA, 16–20 July 2006; Volume 2, pp. 1475–1482.
19. Task-Based Dialog on CoBot. Available online: https://www.youtube.com/watch?v=2LojY9gFK5A (accessed on 16 June 2015).
20. Skubic, M.; Perzanowski, D.; Blisard, S.; Schultz, A.; Adams, W.; Bugajska, M.; Brock, D. Spatial language for human-robot dialogs. *IEEE Trans. Syst. Man Cybern. C* **2004**, *34*, 154–167.
21. Dzifcak, J.; Scheutz, M.; Baral, C.; Schermerhorn, P. 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 Proceedings of the 2009 IEEE International Conference on Robotics and Automation; Kobe, Japan, 12–17 May 2009.
22. Matuszek, C.; Fox, D.; Koscher, K. Following directions using statistical machine translation. In Proceedings of the 5th ACM/IEEE International Conference on Human-robot Interaction, New York, NY, USA, 2–5 March 2010; pp. 251–258.
23. Vogel, A.; Jurafsky, D. Learning to follow navigational directions. In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, Uppsala, Sweden, 11–16 July 2010; pp. 806–814.
24. Dantam, N.; Stilman, M. The motion grammar: Linguistic perception, planning, and control. In *Robotics: Science and Systems (RSS)*; MIT Press: Cambridge, MA, USA, 2011.
25. Shimizu, N.; Haas, A. Learning to follow navigational route instructions. In Proceedings of the 21st International Jont Conference on Artifical Intelligence, Pasadena, CA, USA, 11–17 July 2009; pp. 1488–1493.
26. Holzapfel, H.; Neubig, D.; Waibel, A. A dialogue approach to learning object descriptions and semantic categories. *Robot. Auton. Syst.* **2008**, *56*, 1004–1013.
27. Kaiser, P.; Lewis, M.; Petrick, R.; Asfour, T.; Steedman, M. Extracting common sense knowledge from text for robot planning. In Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 31 May–7 June 2014; pp. 3749–3756.
28. Kollar, T.; Tellex, S.; Roy, D.; Roy, N. Toward understanding natural language directions. In *Proceedings of HRI-2010*; EEE Press: Piscataway, NJ, USA, 2010.
29. Tellex, S.; Kollar, T.; Dickerson, S.; Walter, M.; Banerjee, A.; Teller, S.; Roy, N. Understanding natural language commands for robotic navigation and mobile manipulation. In Proceedings of the National Conference on Artificial Intelligence (AAAI), San Francisco, CA, USA, 7–11 August 2011.
30. Rybski, P.; Stolarz, J.; Yoon, K.; Veloso, M. Using dialog and human observations to dictate tasks to a learning robot assistant. *Intell. Serv. Robot.* **2008**, *1*, 159–167.

31. Chen, X.; Stone, P.; Sucar, L.-E.; van-der Zant, T. *RoboCup 2012: Robot Soccer World Cup XVI*; Springer Verlag: Mexico City, Mexico, 2012.
32. Mu, Y.; Yin, Y. Task-oriented spoken dialogue system for humanoid robot. In Proceedings of the IEEE 2010 International Conference on Multimedia Technology (ICMT), Ningbo, China, 29–31 October 2010; pp. 1–4.
33. Lemaignan, S.; Ros, R.; Sisbot, E. A.; Alami, R.; Beetz, M. Grounding the interaction: Anchoring situated discourse in everyday human-robot interaction. *Int. J. Soc. Robot.* **2012**, *4*, 181–199.
34. Breuing, A.; Waltinger, U.; Wachsmuth, I. Harvesting wikipedia knowledge to identify topics in ongoing natural language dialogs. In Proceedings of the 2011 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT), Lyon, France, 22–27 August 2011; Volume 1, pp. 445–450.
35. Tenorth, M.; Nyga, D.; Beetz, M. Understanding and executing instructions for everyday manipulation tasks from the world wide web. In Proceedings of the 2010 IEEE International Conference on Robotics and Automation (ICRA), Anchorage, AK, USA, 3–7 May 2010; pp. 1486–1491.
36. Gupta, R.; Kochenderfer, M.J. Common sense data acquisition for indoor mobile robots. In Proceedings of the National Conference on Artificial Intelligence, San Jose, CA, USA, 25–29 July 2004; pp. 605–610.
37. Daoutis, M.; Coradeshi, S.; Loutfi, A. Grounding commonsense knowledge in intelligent systems. *J. Ambient Intell. Smart Environ.* **2009**, *1*, 311–321.
38. Cafarella, M.J.; Halevy, A.; Wang, D.Z.; Wu, E.; Zhang, Y. Webtables: Exploring the power of tables on the web. *Proc. VLDB Endow.* **2008**, *1*, 538–549.
39. Schiff, S. Know it all. *The New Yorker*, 31 July 2006.
40. Carlson, A.; Betteridge, J.; Wang, R.C.; Hruschka, E.R., Jr.; Mitchell, T.M. Coupled semi-supervised learning for information extraction. In Proceedings of the Third ACM International Conference on Web Search and Data Mining, New York, NY, USA, 3–6 February 2010; pp. 101–110.
41. Fader, A.; Soderland, S.; Etzioni, O. Identifying relations for open information extraction. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, Edinburgh, UK, 27–31 July 2011; pp. 1535–1545.
42. Nie, Z.; Wu, F.; Wen, J.-R.; Ma, W.-Y. Extracting objects from the web. In Proceedings of the 22nd International Conference on Data Engineering, Atlanta, GA, USA, 3–8 April 2006; pp. 123–123.
43. Matuszek, C.; Witbrock, M.; Kahlert, R.C.; Cabral, J.; Schneider, D.; Shah, P.; Lenat, D. Searching for common sense: Populating cyc^{TM} from the web. In Proceedings of the National Conference on Artificial Intelligence, Pittsburgh, PA, USA, 9–13 July 2005; Volume 20, p. 1430.
44. Chen, X.; Xie, J.; Ji, J.; Sui, Z. Toward open knowledge enabling for human-robot interaction. *J. Hum. Robot Interact.* **2012**, *1*, 100–117.
45. Chen, X.; Ji, J.; Sui, Z.; Xie, J. Handling open knowledge for service robots. In Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, Beijing, China, 3–9 August 2013.

46. Zhou, K.; Zillich, M.; Zender, H.; Vincze, M. Web mining driven object locality knowledge acquisition for efficient robot behavior. In Proceedings of the International Conference on Intelligent Robots and Systems (IROS), Vilamoura, Portugal, 7–12 October 2012; pp. 3962–3969.
47. Samadi, M. Facts and Reasons: Anytime Web Information Querying to Support Agents and Human Decision Making. Ph.D. Thesis, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA, 2015.
48. Samadi, M.; Veloso, M.; Blum, M. OpenEval: Web information query evaluation. In Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence (AAAI-13), Bellevue, WA, USA, 14–18 July 2013.