EPiC Series in Computing
Volume 55, 2018, Pages 176–187
GCAI-2018. 4th Global Conference on Artificial Intelligence

Interpretability of a Service Robot: Enabling User Questions and Checkable Answers

Vittorio Perera and Manuela Veloso

School of Computer Science,
Carnegie Mellon University,
Pittsburgh, Pennsylvania
vdperera@cs.cmu.edu,
mmv@cs.cmu.edu

Abstract

Service robots are becoming more and more capable but at the same time they are opaque to their users. Once a robot starts executing a task it is hard to tell what it is doing or why. To make robots more transparent to their users we propose to expand the capabilities of robots to not only execute tasks but also answer questions about their experience.

During execution, our CoBot robots record log files. We propose to use these files as a recording of the robot experience. Log files record the experience of the robot in term of its internals. To process information from the logs we define Log Primitives Operations (LPOs) that the robot can autonomously perform. Each LPO is defined in terms of an operation and a set of filters. We frame the problem of understanding questions about robot past experiences, as grounding input sentences to LPOs. To do so, we introduce a probabilistic model to ground sentences to these primitives. We evaluate our approach on a corpus of 133 sentences showing that our method is able to learn the meaning of users' questions.

Finally we introduce the concept of checkable answers to have the robot provide answers that better explain the computation performed to achieve the result reported.

1 Introduction

Mobile service robots have become increasingly capable at performing autonomously many different tasks. Our CoBot robots have traveled autonomously for more than 1000km [4], the Keija robot was deployed in a mall as robotic guide [8], and the STRANDS project has developed robots aimed for long-term deployment in everyday environments [12].

Although the capabilities of robots have been steadily increasing, it is still unclear how robots will fit in our everyday life, and how the interaction between users and robots is going to shape up. One crucial aspect is related to the issue of trust between users and robots. In our deployment of the CoBot robots we observed how the internal state of the robot is often hidden to the users [3]. Ideally, users should be able to ask the robot what it did or why a particular choice or action was taken. In this work we take steps in this direction by enabling users to ask questions about the past autonomous experience of the robot. Since our interest is on mobile service robots, we focus on questions about the time and distance traveled while executing tasks, but the approach we propose is general.

Our first contribution is a novel use of *log files*. Typically, when available, these files are used by developers for debugging purposes; in this work we use the logs recorded by our robots as the source of information. The CoBot robots can search the logs to autonomously answer questions asked from their users using natural language. In order for the robot to automatically retrieve information from the logs files, we define Log Primitives Operations (LPOs). Using LPOs we extend the ability of our robots to not only execute tasks, but also to perform operation on the log files they record. Our second contribution is to frame the problem of question understanding as grounding input sentences into LPOs. We define a joint probabilistic model over LPOs, the parse of a sentence, and a learned Knowledge Base. Our Knowledge Base is designed to store and reuse mappings from natural language expressions to queries the robot can perform on the logs. To evaluate our approach for question understanding we crowd-sourced a corpus of 133 sentences. Our results show that, using the proposed approach, the robot is able to learn the meaning of the questions asked. Finally, our third contribution, is to introduce the concept of *checkable answers*. In order to provide an explanation to the answer provided to the user, we propose to use checkable answers that reveal the computation performed by the robot to come up with the result offered as answer.

The rest of the paper is organized as follows. First we review the relevant related work. Next we present the structure of the log files recorded by our robots. We then present the structure of the Log Primitives Operations our robot can autonomously perform on its logs. We then introduce our model for query understanding and present our experimental results. Finally we introduce the concept of *checkable answers* with comprehensive examples.

2 Related Work

Our goal is to enable robots to answer questions about their past experience in order to make them more transparent and acceptable to their users. In the literature, we identify two main categories of relevant work to our approach. First we review works pertaining to the intelligibility or explanation of intelligent agents, and second we look at works enabling autonomous robots to understand natural language.

In Human-Computer Interaction a large body of works focus on intelligibility of intelligent systems (e.g., for context-aware systems [9]). In [16], the authors present a study performed over 200 participants. The results shows that automatically providing explanation for a system's decision can improve users trust, satisfaction and acceptance level. In [15], the authors use a music recommender system to show how the mental model of a user affect the ability of the user to more effectively operate the system itself. In particular, they show that completeness in the explanation provided by the system helps the user build more useful mental models. Last, [5] uses a Clinical Decision Support System (CDSS) to show how automatically generated explanations can increase the users trust. Similarly we aim at improving the understanding and trust users have toward our robots. As a first step, in this work we enable our robots to understand and answer questions on their past experience.

We expect users to ask questions to our robots using natural language. Enabling an agent to understand natural language has been a long standing goal for AI researchers [23]. In the robotics literature we identify two main approaches to enable robots to understand natural language. The first is to map the input sentence to a logical form that the robot can evaluate and act upon (e.g., λ-calculus [8, 7, 1, 17]).

Action:	GoTo
Arguments:	Destination

Action:	FindAndFetch
Arguments:	Object
	Source
	Destination

Action:	Escort
Arguments:	Person
	Destination

Figure 1: Semantic frames and their arguments for the tasks our service robots execute.

The second approach, that we also adopt, is to consider a probabilistic model to map language into robot actions as in [11, 14, 21, 13]. In many of the previous works, robots learned how to understand commands or instructions requiring them to execute physical actions. We move beyond these works by enabling robots to understand questions about the time and distance traveled while executing tasks. Finally, to represents the tasks executed by the robots, we use semantic frames [10], a formalism extensively applied both to linguistics [2] and to robotics [18, 22].

3 Robot Logs

Many systems come with a logging capabilities. These capabilities are designed to allow developers to find and fix unwanted behaviors (i.e., bugs). For a mobile service robot a log file might include sensory data (e.g., readings from cameras or depth sensors), navigation information (e.g., odometry readings or estimated positions), scheduling information (e.g., the action being performed or the time to a deadline) or all of the above. Although initially developed for debugging purposes we introduce a novel use for the log files recorded by a robot. We propose to use the logs recorded by a robot as its memory, and to use the information stored in the logs to have robots answer questions about their past experience.

The CoBot robots are developed using ROS [20], their code is designed in a modular fashion where each module can publish messages on a topic or subscribe to it to receive messages. Our logging system, native to ROS[1], records every message exchanged by the modules running and save them in a log file. When the robot is running messages are exchanged over more then 50 topics at 60Hz. The information exchanged on these topics ranges from low-level micro-controller information to scheduling data, from encoder readings to GUI (Graphical User Interface) events information.

A detailed description of the messages and information recorded in our log file is presented in [4]. Here we provide a short overview of the information we record and we categorize the messages 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 (Θ), 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.

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

"What was the farthest you had to travel while delivering a book?"
MAX(`quantity=distance, taskID=delivery,object=book`)

(a)

"What was the fastest task you have ever executed?"
MIN(`quantity=time`)

(b)

"How long does it usually take you to escort visitors to the lab?"
AVG(`quantity=time, taskID=escort, destination=F3201`)

(c)

"What was the total time you spent delivering something in the last three days?"
SUM(`quantity=time, taskID=delivery, startTime=10/28/2017, endTime=10/31/2017`)

(d)

Figure 2: Examples of input sentences and corresponding queries to be extracted. Each sentences imply a different set of filters to be used.

- **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 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.

Since our goal is to enable a robot to answer questions about the time and distance traveled while executing tasks we are going to focus on messages in the execution and task levels. Specifically, we focus on two topics, the first one is `/Localization` and belongs to the execution level. This topic records the position (x, y, z, Θ) of the robot (z indicates on which floor of the building the robot is currently located). Using the messages published on this topic we reconstruct the amount time and the distance traveled while executing a specific task together with the path the robot took. The second topic we consider is `/TaskPlannerStatus`. This topic records information about the task being performed which include: the *semantic frame* representing the task, the amount of time the task has been executed for, and the expected remaining time in the current task.

To represent the three different tasks our robot can perform we use semantic frames. Each frame is composed by the action the robot should take (i.e., the task name) and a variable number of arguments. Fig 1 shows the three semantic frames corresponding to the tasks our robots can perform.

Finally, although it is not crucial to our contribution, it is worth noticing that log files are sequential by nature. In order to quickly search through the logs and answer the users questions, we use an intermediate representation where each task and its relevant information (task type, task arguments, time and distance traveled) is indexed by its starting time.

4 Log Primitives Operations

In order to be able to answer questions our robots need to retrieve the relevant information from the logs. To do so, we designed Log Primitive Operations (LPOs) that the robot can autonomously perform on log files. An LPO is composed by an *operation* and a set of *filters*. The *operation* defines the computation that the robot performs on the records selected from the logs using the *filters*. Each record in the log files contains different fields (e.g., the position of the robot or the task being executed). Here, we define four quantitative operations that the robot can perform on the logs. A quantitative operation operates on one or more numerical field of the records being considered. The operations we define are:

MAX: returns the largest value for the field considered in the record specified by the filters.

MIN: return the smallest value for the field considered in the record specified by the filters.

AVG: returns the average value for the field considered in the record specified by the filters.

SUM: returns the total value for the field considered in the record specified by the filters.

In our previous work [19] we defined three additional non-quantitative operations. These operation are performed on the record(s) matching the LPO's filter and do not require to operate on any numerical field. We quickly review these additional operations here:

CHECK: returns true if the logs have at least a record matching all the filters specified by the LPO, otherwise it returns false.

COUNT: returns the number of records matching the filters specified in the LPO.

SELECT: returns all the records matching the filters specified in the LPO.

Filters are used to select the record(s) relevant to the query, we identify three types of filters. The first type of filters we define are *task-based* filters. A user might ask about the time spent executing a specific type of task or the distance traveled while going to a specific location. We allow for five task-related filters, namely: *taskID*, *destination*, *source*, *object*, *person*. These five filters match the argument of the structure of the semantic frames we use to represent the tasks the robot can execute.

LPOs performed on the logs refer to the past experiences of the robot, therefore we define a second type of filter to select the window of time relevant to the LPO. We define this type of filter as *time-based* filters. A time-based filter is typically characterized by a starting and ending time.

Finally, the third type of filter, *quantity-based* filter, is used to select which field should be considered when applying our quantitative operations. In this work we focus on understanding questions about the time and distance service robots travel during their deployment; hence the quantity-based filter is used to specify whether a question refers to time or distance.

In the next section we show how we map users questions to LPOs that the robot can autonomously perform on the logs. Fig. 2 shows, for each of the quantitative operation we defined, an example of input sentence and the corresponding IPO we aim to extract.

5 Question Understanding

In the previous section we introduced the Log Primitive Operations the CoBot robots can autonomously perform on their log files. In order to enable a mobile robot to answer questions about the distance and time it has traveled, we frame the problem of understanding an input sentence as finding the best matching LPO to perform on the log files. Formally we define a joint probabilistic model over the parse of a sentence (Ψ), the possible LPOs ($\mathcal{L}$), and a Knowledge Base ($\mathcal{KB}$). We aim at finding the LPO $\mathcal{L}^*$ that maximizes the joint probability, that is:

$$\arg\max_{\mathcal{L}} \; p(\mathcal{L}, \Psi | \mathcal{KB})$$

Assuming the parser is conditionally independent from the Knowledge Base we rewrite our joint model as:

$$p(\mathcal{L}, \Psi | \mathcal{KB}) = p(\mathcal{L} | \mathcal{KB}) p(\Psi)$$

We refer to the two factors of this model as the *Grounding Model* and the *Parsing Model.*

5.1 Parsing Model

In order to parse questions from users we adopt a shallow semantic parser. Each word is first labeled using one of the following labels: *Operation*, *Quantity*, *TaskID*, *Destination*, *Source*, *Object*, *Person*, and *Time*. We denote this set of label as L. These eight labels match the structure of an IPO the robot can perform on the logs and allow us to retrieve the part of the sentence that refer to the operation or one of the filters. A special label *None* is used to label words that can be disregarded. Once each word in a sentence has been labeled, we chunk together contiguous words with the same label. Fig 3 shows an example of a parsed sentence.

"What was the [shortest]$_{Operation}$ [time]$_{Quantity}$ it took you to complete an [escort]$_{TaskID}$ task in the [last three days]$_{Time}$?"

Figure 3: An example of a parsed sentence. Each word not between square bracket was labeled as *None* and therefore ignored.

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

$$\begin{aligned} p(\Psi) &= p(l_1, ..., l_N | s_1, ..., s_N) \\ &= \frac{1}{Z} \exp\left(\sum_{1}^{N} w \cdot \phi(l_i, s_{i-1}, s_i, s_{i+1})\right) \end{aligned}$$

where Z is a normalization constant that ensures the distribution $p(\Psi)$ sums to 1.

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

5.2 Grounding Model

Using the parsing model we are able to extract from a sentence the structure of the LPO the robot needs to perform on the logs. The semantic parser identifies, for each chunk of the sentence, whether it refers to the operation to be performed, to one of the filters, or if we can disregard it. Users can refer to the same operation in multiple different ways. As an example, consider a user asking the robot to perform an AVG operation, they might ask "what is the usual time" or "what is the typical time". Therefore, in order to fully understand a sentence, we need to map words to symbols the robot can process, that is we need to ground the sentence.

The possible groundings for the *Operation* label are the four operations we defined on the logs. For our robot, the *Quantity* label can be grounded to either time or space. The *Destination* and *Source* labels are grounded to office number in the building. The *Object* and *Person* labels do not require to be grounded as we can directly search the logs for matching strings. Finally, we need to ground the chunks labeled as *Time* to an explicit start and end date.

In order to save and reuse mapping from natural language expression to groundings we designed a Knowledge Base (KB). Our Knowledge Base is designed as a collection of binary predicates where the first argument is the natural language expression, and the second is its grounding. We use four different predicates that closely match the label used by our semantic parser, namely: *operationGrounding*, *quantityGrounding*, *taskGrounding locationGrounding*. Fig. 4 shows an example of the knowledge base.

OperationGrounding("farthest", `MAX`)
OperationGrounding("longest", `MAX`)
TaskGrounding("delivering", `DELIVER`)
QuantityGroundsing("far", `SPACE`)
LocationGrounding("Manuela's office", `F8002`)

Figure 4: An example of the Knowledge Base and the predicates it stores.

To each predicate in the Knowledge Base we also attach a confidence score measuring how often a natural language expression e has been grounded to a specific grounding γ; we use $C_{e,\gamma}$ to refer to the confidence score of a specific predicate.

We use the confidence score attached to each predicate in the Knowledge Base to compute $p(\mathcal{L}|\mathcal{KB})$. As we have seen an LPO $\mathcal{L}$ is composed by an operation O and a set fo filters f, therefore we approximate the probability of a query as:

$$p(\mathcal{L}|\mathcal{KB}) = p(O|\mathcal{KB}) \prod^{i} p(f_i|\mathcal{KB})$$

Each of the term in this product can be computed directly from the confidence scores stored in the Knowledge Base. To compute the probability of a specific grounding γ^*, whether this it for the operation or for one of the filters, we use the following formula:

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

When our robot receives a question it first parses it to extract the structure of the LPO. Next, for each of the chunks extracted, it searches the Knowledge Base for matching predicates and computes the most likely grounding. When a natural language expression e cannot be grounded using the Knowledge Base, the robot enters a dialog asking the user to explicitly provide the grounding and it then updates its Knowledge Base.

Finally, to ground the expressions referring to time-related filters we use SUTime [6] an external library to recognize and normalize time expression. Our Knowledge Base is able to learn static mapping from natural language expressions to groundings but time expression often requires to be functionally grounded, that is we also need to take into account the current time. As an example consider the expression "in the last three days"; we cannot ground this expression to fixed start and end dates as they continuously change.

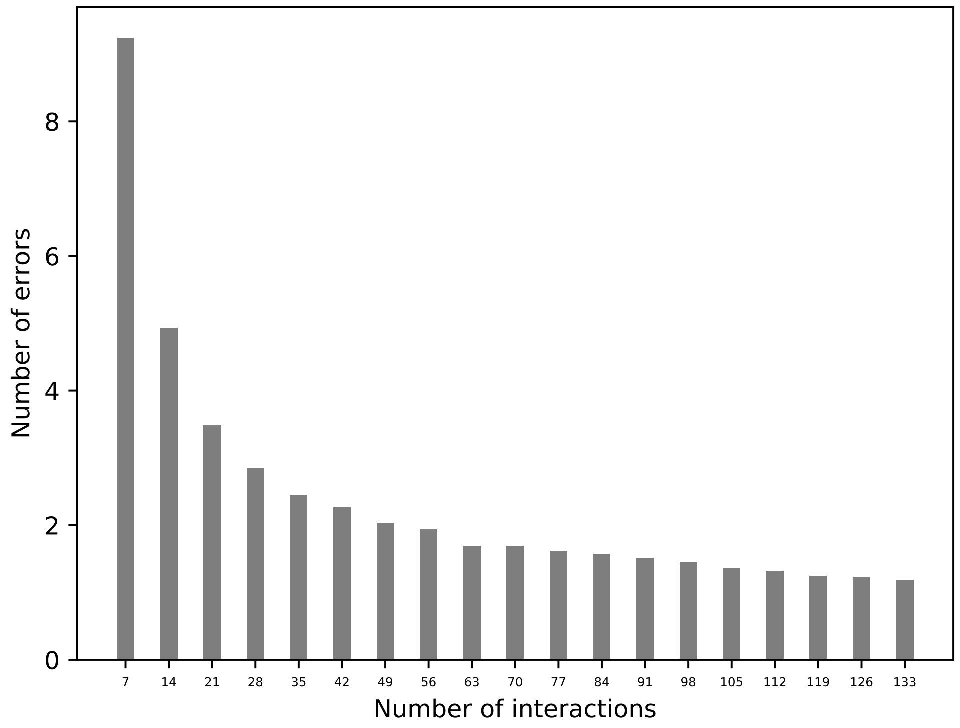

Figure 5: The number of error made by the robot in grounding questions.

6 Experimental Evaluation

In order to evaluate our approach, we crowd-sourced a corpus of 140 sentences asking users to provide questions asking a robot about the time and distance it spent executing tasks. These 140 sentences were provided by 20 different users but we had to discard 7 of them because either non-grammatical or because they could not be matched by any of the operation we defined. Therefore, in our experiments, we use a total of 133 sentences.

We hand labeled each sentence twice, first we label each word in the sentence with labels for the parsing model. Second, we label each sentence with the corresponding LPO to be performed on the logs. After training our CRF, we evaluate the accuracy of the semantic parser using leave-one-out cross validation. We iterate on our corpus by leaving out each of the twenty users that took part in the crowd-sourcing survey; the parser achieves an average F_1 scores of 0.84.

To evaluate the grounding model we first look at the error our robot makes in grounding input sentences. We consider an error both when: 1) the robot cannot infer the grounding using the knowledge base and has to enter a dialog, and 2) the grounding inferred does not match our annotation. In Fig 5 each bin represents seven interactions, that is seven sentences the robot received and grounded. We start with an empty knowledge base, therefore the robot is initially asking for the grounding of each chunk identified by the parser. As the Knowledge Base grows, the number of errors made quickly decreases. By the end of our experiment we can observe how the robot makes less than two errors, that is it is able to understand five out of seven sentences without needing to ask or making any mistake. It is worth noticing that, for

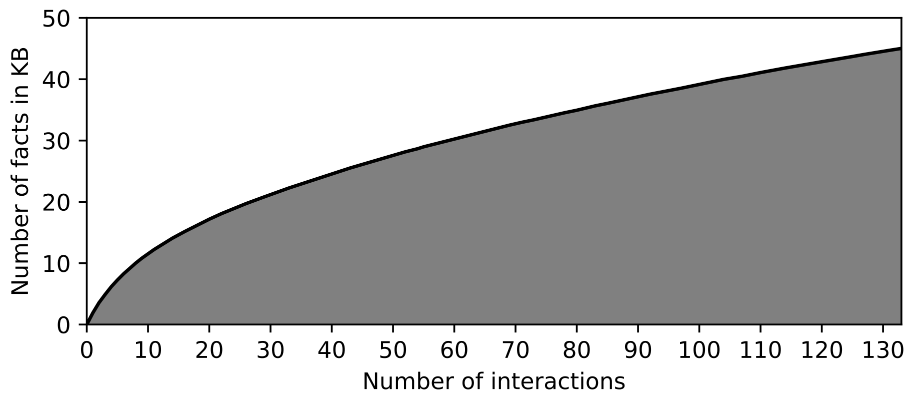

Figure 6: The number of facts stored in the Knowledge Base after each interaction.

this kind of experiment, the order in which the sentences are processed can have a big impact. To smooth out the possible effect of specific sequences of sentences we computed the results presented in Fig. 5 as the average of 1000 randomized runs.

We also analyze the size of the Knowledge Base as the robot process more and more sentences. Fig 6 shows the number of facts (i.e., different predicates) stored in the Knowledge Base. Once again, we smooth out the possible effect of specific sequences of sentences by plotting the average number of facts stored in the Knowledge Base over 1000 randomized runs. Initially we expected this plot to flatten out after the first few interactions. Instead we observe that during the first interactions facts are quickly added to the Knowledge Base (i.e., the plot shows a high slope), as time progress the rate at which facts are added decreases but never goes to zero. By inspecting the Knowledge Base after the experiment we observed that few predicates have very high count and many of the remaining ones have only been used once. This behavior mirrors the long-tail distribution typically found in language models, few expressions are used very commonly and are quickly learned but, from time to time, we still encounter new ways to refer to operations or filters.

7 Checkable Answers

So far we have discussed how to enable a robot to understand questions. A user might ask "How much time did your shortest task take?", the robot parses and grounds the sentence, searches in its log files and comes up with the answer "42 seconds". Although simply reporting on time (or distance) answers the question, it hardly provides enough information for the user to verify if the robot is truthful or not. Therefore, we propose for the robot to use *Checkable Answers* to reveal the computation performed by the robot, and to better explain the result offered as answer.

Checkable Answer: we define Checkable Answer for quantitative operations as an answer that provides both the final value resulting from the operation and a procedure to verify the consistency of such answer.

For each of the four quantitative operations we design a template the robot can use to generate Checkable Answer. These templates use information the robot can extract from the log files using a different operations or the map it uses for navigation.The templates we designed

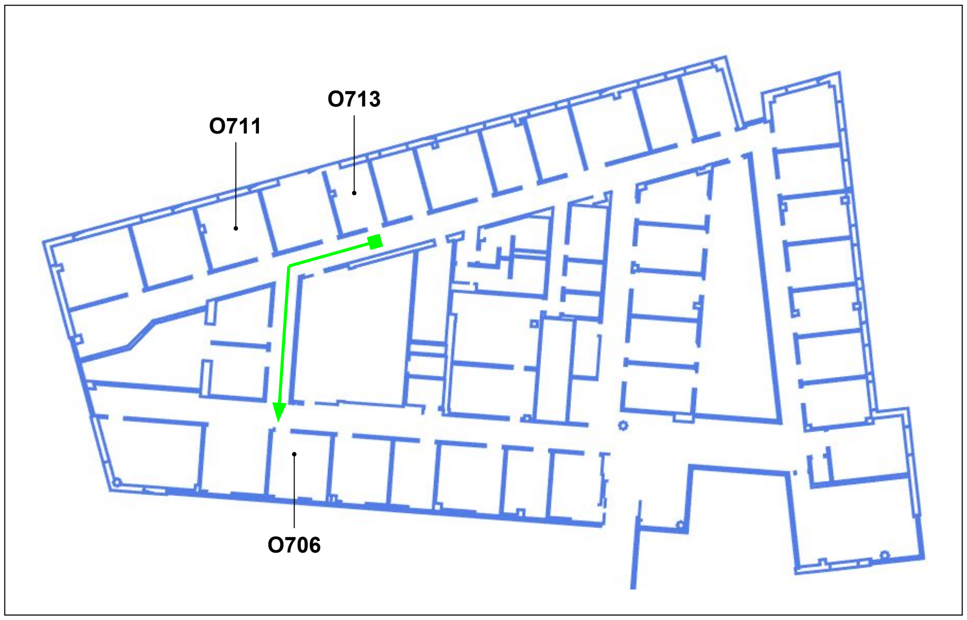

Figure 7: The path taken by the robot where we identify two segments

are the following:

MAX/MIN: the robot reports on each segment of the path taken while executing the task matching the filters in the query. We extract the path taken using the messages on the `/Localization` topic. We segment the path every time the robot turns, and refer to each turning point using the closest office to it. Fig. 7 shows in green the path taken by the robot for the following checkable answer:

Q: *What was the shortest task you ever executed?*
A: *I executed my shortest tasks on the 2nd of April 2013, it took me 42 seconds. I started from office O713 at 1:41:01pm, passed by office O711 at 1:41:16 and I arrived at office O706 at 1:41:43pm*

AVG: the robot reports on the total number of tasks matching the filters in the query, the longest and shortest task (computed using the MIN and MAX operators) and the total time/distance traveled (computed using the SUM operator).

Q: *On average how long do GoTo tasks take?*
A: *I performed a total of 5 tasks matching your request. The longest task took 218 seconds while the shortest took 4 seconds. The total time spent executing tasks matching your request is 525 seconds.*

SUM: the robot reports on the total number of tasks matching the filters in the query, the longest and shortest task (computed using the MIN and MAX operators) and the average time/distance traveled (computed using the AVG operator).

Q: *What is the total time you spent escorting someone?*
A: *I performed a total of 14 tasks matching your request. the total time was 2163 seconds.*

The longest task took 407 seconds, while the shortest took 4 seconds. The average time spent executing tasks matching your request is 145.50 seconds.

As shown in the examples above, all the templates we propose to provide checkable answers are designed in terms of the answer to the LPO matching the user requested and the answer to additional LPOs. These additional LPOs help provide more context and reveal the process the robot use to compute the answer provided.

8 Conclusions

With the goal of making service robots more transparent to their users, we enabled them to answer questions about the time and distance traveled while they execute tasks. To do so we first contribute a novel use of the log files our robot records. Rather then limiting the logs to debugging tool we propose to use them as the memory of the robot to answer questions. Our second contribution is a probabilistic approach to question understanding. We introduce Interrogation Primitives Operations, defined in term of operations and filters, that our robot can autonomously perform on the log files. Our approach defines a probabilistic model over the parse of a questions, possible LPOs and a Knowledge Base. We evaluated the proposed approach on a corpus of 133 sentences showing how the robot are able to quickly learn the meaning of questions asked. Finally, our last contribution is the use of *checkable answers*, where answers to users questions are provided using in terms of multiple log operation. This provides additional context and let the user quickly verify the answer received.

References

[1] Yoav Artzi and Luke Zettlemoyer. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1:49–62, 2013.

[2] Collin F Baker, Charles J Fillmore, and John B Lowe. The berkeley framenet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*, pages 86–90. Association for Computational Linguistics, 1998.

[3] Kim Baraka, Stephanie Rosenthal, and Manuela Veloso. Enhancing Human Understanding of a Mobile Robot's State and Actions using Expressive Lights. In *Proceedings of RO-MAN'16, the IEEE International Symposium on Robot and Human Interactive Communication*, Columbia University, NY, August 2016.

[4] Joydeep Biswas and Manuela Veloso. The 1,000-km challenge: Insights and quantitative and qualitative results. *IEEE Intelligent Systems*, 31(3):86–96, 2016.

[5] Adrian Bussone, Simone Stumpf, and Dympna O'Sullivan. The role of explanations on trust and reliance in clinical decision support systems. In *Healthcare Informatics (ICHI), 2015*, pages 160–169. IEEE, 2015.

[6] Angel X Chang and Christopher D Manning. Sutime: A library for recognizing and normalizing time expressions. In *LREC*, volume 2012, pages 3735–3740, 2012.

[7] Kai Chen, Dongcai Lu, Yingfeng Chen, Keke Tang, Ningyang Wang, and Xiaoping Chen. The intelligent techniques in robot kejia–the champion of robocup@ home 2014. In *Robot Soccer World Cup*, pages 130–141. Springer, 2014.

[8] Yingfeng Chen, Feng Wu, Wei Shuai, Ningyang Wang, Rongya Chen, and Xiaoping Chen. Kejia robot–an attractive shopping mall guider. In *International Conference on Social Robotics*, pages 145–154. Springer, 2015.

[9] Anind K Dey. Explanations in context-aware systems. In *ExaCt*, pages 84–93, 2009.

[10] Charles J Fillmore. Frames and the semantics of understanding. *Quaderni di semantica*, 6(2):222–254, 1985.

[11] Maxwell Forbes, Rajesh PN Rao, Luke Zettlemoyer, and Maya Cakmak. Robot programming by demonstration with situated spatial language understanding. In *Robotics and Automation (ICRA), 2015*, pages 2014–2020. IEEE, 2015.

[12] Nick Hawes, Chris Burbridge, Ferdian Jovan, Lars Kunze, Bruno Lacerda, Lenka Mudrová, Jay Young, Jeremy L. Wyatt, Denise Hebesberger, Tobias Körtner, Rares Ambrus, Nils Bore, John Folkesson, Patric Jensfelt, Lucas Beyer, Alexander Hermans, Bastian Leibe, Aitor Aldoma, Thomas Faulhammer, Michael Zillich, Markus Vincze, Muhannad Al-Omari, Eris Chinellato, Paul Duckworth, Yiannis Gatsoulis, David C. Hogg, Anthony G. Cohn, Christian Dondrup, Jaime Pulido Fentanes, Tomás Krajník, João Machado Santos, Tom Duckett, and Marc Hanheide. The STRANDS project: Long-term autonomy in everyday environments. *CoRR*, abs/1604.04384, 2016.

[13] Albert S Huang, Stefanie Tellex, Abraham Bachrach, Thomas Kollar, Deb Roy, and Nicholas Roy. Natural language command of an autonomous micro-air vehicle. In *Intelligent Robots and Systems (IROS), 2010*, pages 2663–2669. IEEE, 2010.

[14] Thomas Kollar, Stefanie Tellex, Deb Roy, and Nicholas Roy. Toward understanding natural language directions. In *Human-Robot Interaction (HRI), 2010*, pages 259–266. IEEE, 2010.

[15] Todd Kulesza, Simone Stumpf, Margaret Burnett, Sherry Yang, Irwin Kwan, and Weng-Keen Wong. Too much, too little, or just right? ways explanations impact end users' mental models. In *Visual Languages and Human-Centric Computing (VL/HCC), 2013 IEEE Symposium on*, pages 3–10. IEEE, 2013.

[16] Brian Y Lim, Anind K Dey, and Daniel Avrahami. Why and why not explanations improve the intelligibility of context-aware intelligent systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2119–2128. ACM, 2009.

[17] Cynthia Matuszek, Evan Herbst, Luke Zettlemoyer, and Dieter Fox. Learning to parse natural language commands to a robot control system. In *Experimental Robotics*, pages 403–415. Springer, 2013.

[18] Vittorio Perera, Robin Soetens, Thomas Kollar, Mehdi Samadi, Yichao Sun, Daniele Nardi, René van de Molengraft, and Manuela Veloso. Learning task knowledge from dialog and web access. *Robotics*, 4(2):223–252, 2015.

[19] Vittorio Perera and Manuela Veloso. Learning to Understand Questions on the Task History of a Service Robot. In *Proceedings of RO-MAN'17, the IEEE International Symposium on Robot and Human Interactive Communication*, Lisbon, Portugal, August 2017.

[20] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.

[21] Stefanie A Tellex, Thomas Fleming Kollar, Steven R Dickerson, Matthew R Walter, Ashis Banerjee, Seth Teller, and Nicholas Roy. Understanding natural language commands for robotic navigation and mobile manipulation. 2011.

[22] Brian J Thomas and Odest Chadwicke Jenkins. Roboframenet: Verb-centric semantics for actions in robot middleware. In *Robotics and Automation (ICRA), 2012*, pages 4750–4755. IEEE, 2012.

[23] Terry Winograd. Procedures as a representation for data in a computer program for understanding natural language. Technical report, DTIC Document, 1971.