Enabling Robots to Find and Fetch Objects by Querying the Web (Extended Abstract)

Thomas Kollar, Mehdi Samadi, Manuela Veloso
School of Computer Science, Carnegie Mellon University
tkollar@cmu.edu, {msamadi, mmv}@cs.cmu.edu

ABSTRACT

This paper describes an algorithm that enables a mobile robot to find an arbitrary object and take it to a destination location. Previous approaches have been able to search for a fixed set of objects. In contrast, our approach is able to dynamically construct a cost function to find any object by querying the web. The performance of our approach has been evaluated in a realistic simulator, and has been demonstrated on a companion robot, which can successfully execute plans such as finding a "coffee" and taking it to a destination location like, "Gates-Hillman Center, Room 7002."

Categories and Subject Descriptors

I.2.9 [**Computing Methodologies**]: Robotics

General Terms

Algorithms

Keywords

Robotics, Object Search, Web, Cyber-physical systems

1. INTRODUCTION

Our aim is to make robots that can interact with people in a natural and intuitive way. Toward this end, we look at a problem domain where people ask a robot to find an object and then have the robot deliver the object to a specified destination. This is a challenging problem since people will specify object names using open-ended natural language, leading to many distinct queries. Humans address the variability in the query by using common-sense knowledge. For example, if a person is asked to find and deliver the object "coffee," a person easily knows that "coffee" is likely to be found in the "kitchen." Robots, however, generally have limited access to such knowledge.

To address this limitation, we enable robots to access the web when they are missing task-related knowledge. In the object-finding domain this knowledge relates objects to locations; for example, assuming the robot has no prior experience with an object, such as "coffee," the robot will search the web. Using the results of the web search, the robot is able to predict that a "coffee" is generally found in a "kitchen," instead of in a "printer room." Our approach dynamically incorporates these predictions into a cost function, which minimizes the distance the robot travels and the number of interactions it has with people, while at the same time maximizing the the probability that an object will be found in each of the visited locations. The inferred multi-step plan consists of a sequence of locations to visit and questions to ask people.

We evaluate the robot's performance by executing plans to find 80 objects in a realistic simulator and have demonstrated our approach on a companion robot. This work builds off of OpenEval [4], which is able to evaluate the probability of arbitrary facts (predicates) by querying the web.

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

2. APPROACH

To command a robot to find and fetch objects, people specify a query object (e.g., "coffee") and a place to which the object should be taken (e.g., "Gates-Hillman, room 7002"); the robot infers a plan to find the object and take it to the destination. Our robot overcomes its limitations in object detection and manipulation by asking humans for help [3].

We formulate the problem of finding and fetching objects as maximizing a utility function. If O is an object name (e.g., coffee), then the problem can be formulated as finding the plan that maximizes the utility function U:

$$\arg\max_{\text{plan}} U(\text{plan}|O) \tag{1}$$

The utility of a plan is broken down into the utility of each element: $U(\text{plan}_i)$. Each element of the plan ($plan_i$) visits a location and asks for an object from a person.

$$U(\text{plan}|O) = \sum_{i=1}^{N} U_i(\text{plan}_i|O) \tag{2}$$

The utility U_i consists of three components:

- The probability of the plan. We approximate this term as the probability of a location in the environment (e.g., "kitchen") given a query object (e.g., "coffee"). This probability is high when the location is likely to contain the object.
- A reward for traveling as little as possible. This term is computed by subtracting the distance traveled from the maximum distance the robot could travel for the current component of the plan.
- A reward for the number of interactions that the robot has with a person. This term is computed by subtracting the number of interactions required to search a location from the maximum number of interactions the

robot can have for the current component of the plan.

The first component (the probability of the plan) requires the system to compute the probability of finding an object in a location. This term connects a query object (e.g., "coffee") to a location type in the environment (e.g., "kitchen"). Connecting a query word for an object to a place is challenging because there are thousands of different object names people can use.

2.1 Querying the Web

To evaluate the probability of a plan, we have developed a general predicate evaluator called OpenEval, which returns a probability distribution over instances of predicates [4]. Unlike other approaches which read the web [1], OpenEval evaluates the validity of existing predicates and returns results immediately. For this paper, OpenEval has been trained on a single predicate, *locationHasObject(L, O)*, which is true only when location L contains object O:

$$p(L = \text{kitchen} | O = \text{coffee}) \triangleq p(\text{locationHasObject}(\text{kitchen}, \text{coffee})) \quad (3)$$

At training time a small number of predicate instances, such as *locationHasObject(kitchen, refrigerator)*, are provided. A web search for {"kitchen", "refrigerator"} returns a set of documents (web-pages), from which text snippets are extracted. These text snippets are treated as ground truth examples of the predicate instance, and a SVM classifier is trained to discriminate between different location types. Because Equation 3 is multinomial over locations, OpenEval only needs positive training examples of location / object pairs. At test time, OpenEval will evaluate the probability of a new predicate instance, such as *locationHasObject(kitchen, coffee)* by converting the input relation instance to a search query, such as {"coffee" "kitchen"}, and downloading the highest ranked web pages. This set of documents (web-pages) is classified into one of the location types; a probability is computed according to the proportion of web-pages that is classified as being in a "kitchen" given the object "coffee."

2.2 Optimization

The system takes as input the name of an object (e.g., "coffee"), a destination (e.g., "Gates-Hillman, room 7002"), the current location of robot, a set of example predicates that are used to train OpenEval, and a map that includes the type of each room (e.g., "office," "kitchen," "bathroom," or "other"). A cost function is then instantiated given the query object (Equation 1), and our approach uses beam search to find a sequence of candidate locations that maximizes the utility. The robot then executes the corresponding plan, recomputing the plan after visiting each location.

3. EVALUATION

We have evaluated our approach by showing that OpenEval is able to correctly categorize the location of novel objects and locations. Table 1 shows the probability of different test objects for each of the four location types. The only erroneous prediction is that bathrooms likely contain cups.

To show that our approach is able to search for objects, we have simulated a semantic map of 305 spaces over three floors of an office building. We have evaluated the effectiveness of our approach against two baseline approaches on 80 commands for 40 different object types that were not a part of the training set for OpenEval. The first baseline (frontier) searches nearby locations, using the rewards from Section 2, but not the probability returned by searching the web. The second baseline (ESP) provides a baseline similar to previous work [2]. It uses all the terms in Section 2, but instead of searching the web for relevant documents, it searches data collected from ESP [5].

Object	**Location Types**			
	Bathroom	Printer Room	Kitchen	Office
laptop	0.07	0.23	0.13	**0.56**
papers	0.1	**0.57**	0.12	0.22
cup	**0.36**	0.16	0.29	0.18
coffee	0.22	0.21	**0.3**	0.28

Table 1: The probability that OpenEval assigns to different locations given each novel object type. The location type which has the maximum probability is shown as bold.

The frontier approach finds objects in an average of 45.4 visited locations (standard error of 6.8), ESP finds objects in an average of 31.56 visited locations (standard error of 5.64) and our approach finds objects in an average of 19.21 visited locations (standard error of 4.62). This indicates a clear downward trend in the number of steps required to find query objects when using the web to retrieve knowledge about the physical environment. The number of visited locations is relatively high because a few objects require searching many (or all) locations in the environment.

Sometimes our approach will retrieve a reasonable location for the object, but the test environment does not contain the object in that location. For example, our approach searches bathrooms for "cleaning liquid" first because it can often be found in a home bathrooms, even though office bathrooms in our environment do not contain this object. In addition, sometimes our approach will infer the correct location, but it may still take several steps before the object is found. For example, since not every "office" has a "laptop," "jacket," or "hat" (e.g., often people have desktops, or don't wear hats), our approach will visit a few locations before finding the object. Finally, we have demonstrated our approach on our companion robot (CoBot), showing that it is successfully able to find a "coffee" in the "kitchen" and take it to "Gates-Hillman, room 7002."

4. REFERENCES

[1] A. Carlson, J. Betteridge, R. C. Wang, E. R. H. Jr., and T. M. Mitchell. Coupled semi-supervised learning for information extraction. In *Proceedings of WSDM*, 2010.

[2] T. Kollar and N. Roy. Utilizing object-object and object-scene context when planning to find things. In *Proceedings of ICRA*, pages 4116–4121, 2009.

[3] S. Rosenthal, J. Biswas, and M. Veloso. An effective personal mobile robot agent through symbiotic human-robot interaction. In *Proc. of AAMAS*, pages 915–922, 2010.

[4] M. Samadi, M. Veloso, and M. Blum. Evaluating correctness of propositions using the web. In *Workshop on Learning by Reading and its Applications in Intelligent Question-Answering*, IJCAI, 2011.

[5] L. von Ahn and L. Dabbish. Labeling images with a computer game. In *Proc. of SIGCHI*, pages 319–326, 2004.