Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence

Using the Web to Interactively Learn to Find Objects

Mehdi Samadi
Carnegie Mellon University
Pittsburgh, United States
msamadi@cs.cmu.edu

Thomas Kollar
Carnegie Mellon University
Pittsburgh, United States
tkollar@andrew.cmu.edu

Manuela Veloso
Carnegie Mellon University
Pittsburgh, United States
veloso@cs.cmu.edu

Abstract

In order for robots to intelligently perform tasks with humans, they must be able to access a broad set of background knowledge about the environments in which they operate. Unlike other approaches, which tend to manually define the knowledge of the robot, our approach enables robots to actively query the World Wide Web (WWW) to learn background knowledge about the physical environment. We show that our approach is able to search the Web to infer the probability that an object, such as a "coffee," can be found in a location, such as a "kitchen." Our approach, called ObjectEval, is able to dynamically instantiate a utility function using this probability, enabling robots to find arbitrary objects in indoor environments. Our experimental results show that the interactive version of ObjectEval visits 28% fewer locations than the version trained offline and 71% fewer locations than a baseline approach which uses no background knowledge.

Introduction

Our aim is to make mobile robots that are able to intelligently perform tasks. In this paper, we investigate a *find and deliver* task, where a person specifies an object in open-ended natural language (e.g., "coffee") and the robot will find and deliver the object to a specified destination (e.g., "GHC-8125"). This is a challenging problem because robots have limited perceptual abilities and people use highly variable language when specifying a task. For example, since our robot only has access to the type of a room (e.g., "office" or "kitchen"), it will need learn the relationship between a query object (e.g., "coffee") and these room types. In addition, since a person may ask the robot to find any of thousands of objects, including a "coffee," "pen," or "papers," the robot will need to learn these relationships over all possible query objects. The find and deliver task is easy for humans, who have learned a large set of background knowledge from experience with the environment; for robots, which have limited access to such knowledge, it is much more challenging.

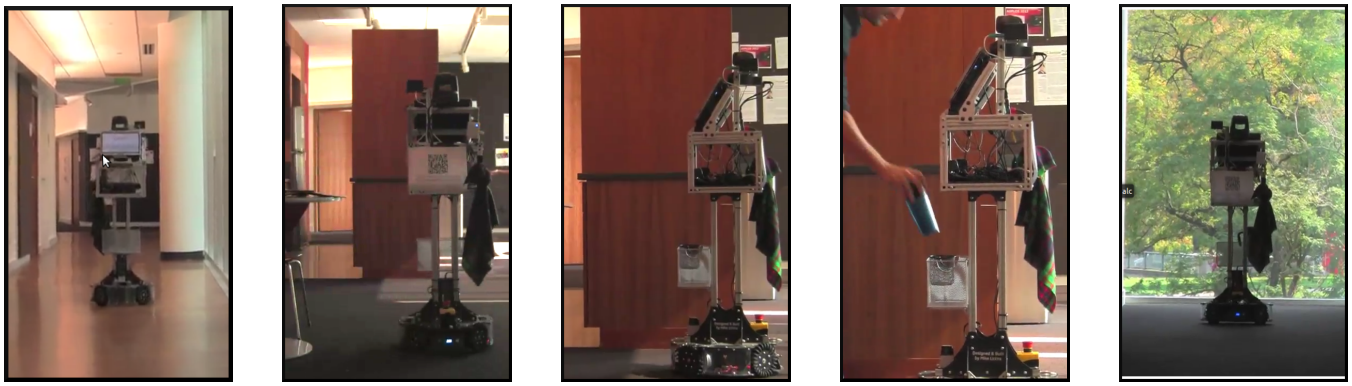
In this paper we introduce an approach, called ObjectEval, which addresses the challenges of the find and deliver task by querying the Web. By downloading and categorizing a set of web-pages on-demand, ObjectEval learns the probability that a location (e.g., "kitchen") will contain an object (e.g., "coffee"). This probability is then dynamically incorporated into a utility function, which takes into account the travel distance to a location, the number of human interactions required to get to a location, and the observation of the object during previous executions at that location. ObjectEval then infers the maximum-utility plan corresponding to a sequence of locations it should visit, asks a human to provide it with the object, and then takes the object to a destination.

We evaluate ObjectEval in three ways. First, we show that ObjectEval is able to predict the location of novel objects against a baseline that is similar to Kollar and Roy (Kollar and Roy 2009). Second, we show that ObjectEval is able to efficiently and automatically find novel objects in a realistic simulated environment consisting of 290 spaces of an office building given only a topological map and the space types (e.g., "office," "bathroom" etc.). We further show that ObjectEval can improve its performance by learning from feedback it gets about the location of objects. Finally, we show that a mobile office robot and can successfully find and deliver "coffee" to a room in our office building. This paper builds on the preliminary work of Samadi et al. (Samadi, Veloso, and Blum 2011) and Kollar et al. (Kollar, Samadi, and Veloso 2012). The main contributions of this paper are:

- An approach for learning background knowledge about the environment by querying the Web.
- An approach for finding and delivering objects that dynamically instantiates a utility function using the results of a Web query, and which interactively learns about the physical environment by getting feedback from humans.

Related Work

Exploring indoor environments has been studied as a part of work on Simultaneous Localization and Mapping. Yamauchi (Yamauchi 1997) described a search process that would incrementally explore new regions of the map. Our baseline search which does not use background knowledge is similar to this approach. Later work focused on balancing the trade-off between localization quality and the need to explore new areas of the environment (Makarenko et al. 2002) (Stachniss, Grisetti, and Burgard 2005).



(a) Receive command (b) Go to location (c) Ask for "coffee" (d) Get "coffee" (e) Deliver object

Figure 1: Our robot searching for an object. In (a) the system gets a query to find a "coffee" and take it to room 7001. In (b) it goes to the nearest kitchen. In (c), it asks a person to place a coffee on it. In (d), it gets the coffee and the person says that the robot has the object. In (e), the robot delivers the object to its destination.

Searching for objects has received considerable interest in the robotics community. Much research has focused on visual object search that does not leverage the Web (Sjöö et al. 2009; Aydemir et al. 2011; Velez et al. 2011; Joho, Senk, and Burgard 2011). Sjöö et al. (Sjöö et al. 2009) present a method for search and localization of objects by using an attention mechanism as a primary step in the recognition process. Using a combination of view planning and visual search, the authors use existing computer vision algorithms to efficiently detect and localize different objects. Aydemir et al. (Aydemir et al. 2011) builds on this by using spatial relations to perform large-scale visual search for objects. Meger et al. Joho et al. (Joho, Senk, and Burgard 2011) focuses problem of finding an object with a mobile robot in an initially unknown, structured environment. While the primary focus is not on vision, they present two methods for object search. The first is a reactive search technique based on objects in the robot's immediate vicinity. The second is a global, inference-based approach that uses the object arrangements of example environments. Finally, Velez et al. (Velez et al. 2011) considers the task of autonomously navigating through the environment while mapping the location of objects. The authors describe an online any-time framework where vantage points provide the most informative view of an object given a noisy object detector. Unlike these approaches, our work uses help from humans to detect and manipulate objects.

Researchers have begun to think about how robots might be integrated with the Web. Meger et al. (Meger et al. 2008) describe an integrated robotic platform that uses web-based training data to train a visual object detector and then perform exploration, mapping, and active attention. Most similar to this work is Kollar et al. (Kollar and Roy 2009), who used the co-occurrences in the labels from the Flickr photo-sharing website as a prior over where objects are located in the physical environment. Posner et al. (Posner, Corke, and Newman 2010) demonstrate a system that queries the Web to help read the visible text in the scene. Tenorth et al. (Tenorth et al. 2011) describe how information on the World Wide Web and intended for human use might be useful for robots.

Finally, there has been much work that is focused on using the Web to extract information. Many approaches use pointwise mutual information (PMI), which is able to measure the semantic similarity of different words (Turney 2001; Turney and Littman 2003; Soderl et al. July 2004; Magnini et al. 2002). The performance of all of these techniques depends on the accuracy of search engine hit counts. To address the fact that search engine hit counts change daily and are otherwise unreliable, Downey et al. (Downey, Etzioni, and Soderl 2005) developed a combinatorial "balls-and-urns" model (Urns model) that uses the redundancy of the same extraction from different sources to compute the probability of the correctness. Finally, the Never-Ending Language Learner (NELL) addresses actively reads the Web, learning structured information from the unstructured webpages (Carlson et al. 2010).

ObjectEval

Our approach, called ObjectEval, enables a robot with limited sensing to search for an object. By using symbiotic autonomy the robot is able to ask people to help it perform tasks, including manipulation and object detection (Rosenthal, Biswas, and Veloso 2010; Biswas and Veloso 2012). To find an object, the robot must therefore (1) receive a command to find an object (e.g., "coffee") and take it to a destination (e.g., room 7001), (2) compute a sequence of locations to visit by maximizing long-term utility, (3) visit a location, (4) ask a person to retrieve the object and finally (5) if there, deliver the object to the destination or if not, go to the next location to look for the object. An example of our robot finding a "coffee" can be seen in Figure 1.

Model

ObjectEval takes as input an object name (e.g., papers) and a destination room (e.g., room 8120), and returns a plan consisting of locations that robot should visit. Finding objects

requires trading off different objectives including: the number of interactions with people, the distance traveled, the existence of objects at previously visited locations, and probability of finding an object in a location. ObjectEval combines these objectives into a utility function that, when maximized, generates a plan that the robot can execute to find an object effectively. If O is an object name (e.g., "papers"), and U is the utility function, then the problem can be formulated as finding the plan that maximizes the utility:

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

The plan is broken down into a sequence of steps (plan_i), each of which visit a location and ask for an object from a person. The robot receives a reward (R) when it executes ith step of the plan. The current step in the plan is successful with probability $p(\text{plan}_i|O)$.

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

In order to capture the objective of finding objects quickly, the reward at each step is broken down into three components:

$$R(\text{plan}_i, O) = D(\text{plan}_i) \times I(\text{plan}_i) \times F(\text{plan}_i, O) \tag{3}$$

Since the robot should consider plans that travel as little as possible, we include the reward D, which is dependent on the distance the robot travels. D is computed by subtracting the distance traveled from the maximum distance the robot could travel. Since people are used as a part of the search process to find and manipulate objects, we include the reward I, which is dependent on the number of interactions that the robot has with a person. I is computed by subtracting the number of interactions required to search a location for an object from the maximum number of interactions the robot will need to search any location. Finally, in order to take advantage of feedback from people, we include the reward F, which uses previous searches to help search for objects. The value of F is 1 if a query object has been seen at the search location, 0.5 if the location has not been explored, and 0 if it is known not to exist there. Although F is fixed in this paper, learning a dynamic model for how objects move would enable ObjectEval to handle cases where the query object moves between different locations in the environment.

The second component of Equation 2 requires us to compute the probability of a part of the plan. As a proxy for the probability of the plan, we use the probability that the location at the ith step of the plan will contain an object given that the object was not seen at the previously visited locations in the plan. If l_j is multinomial over location types (e.g., "office," "printer room," "bathroom") and O is the query object, then the we can compute this probability as:

$$p(\text{plan}_i|O) \approx \left[\prod_{j=1}^{i-1} (1 - p(l_j|O))\right] \times p(l_i|O) \tag{4}$$

In order to find the plan with a maximum utility, the robot must be able to compute $p(l_i|O)$. This term connects a query object O (e.g., "papers") to a location type in the environment (e.g., "printer room"). Connecting a query word for an object to a place where the robot can find the object is challenging because there are thousands of different object names people might use. We calculate the probability $p(l_i|O)$ by querying the Web for the validity of the predicate *locationHasObject(l,O)* over all location types l. For example:

$$\begin{aligned} p(l_j = &\text{kitchen}|O = \text{coffee}) \\ &\triangleq p(\text{locationHasObject(kitchen, coffee)}) \end{aligned} \tag{5}$$

In the next section we describe how ObjectEval obtains the probability of instances of the predicate *locationHasObject*.

Querying the Web

The World Wide Web (WWW) contains an enormous amount of semantic information that might be useful for robots. In this paper, we investigate the use of the semantic information on the Web to predict the location of objects in real-world environments. We expect that objects physically present in a location will be found frequently 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. Figure 2 shows example of text snippets that are found on the Web for object "papers" and locations "printer room" and "bathroom."

ObjectEval will compute the probability from Equation 5 by converting predicate instances in first-order logic, such as *locationHasObject(papers, printer room)*, into a search query such as { "papers", "printer room"}. These search queries can return hundreds or thousands of the most relevant web-pages that relate these terms. The search query includes both the name of the location type and the name of the query object in order to retrieve highly relevant web-pages. In contrast, a search query such as *"papers"* will return both relevant and irrelevant web-pages for determining if "papers" can be found in a "printer room."

The text on the web-pages that is most relevant to a predicate instance will be near the search terms. We therefore extract text snippets from each of the web-pages that include up to 10 words before, after, and in between the query object and location words. If there are multiple text snippets extracted from the same web-page, we merge them into a single text snippet. Each of the text snippets is then transformed into a *feature vector*, where each element corresponds to the frequency of a distinct word in the text snippet. The dimension of the vector is equal to the total number of distinct words that exist in the training data. All the stop words have been deleted, since we expect these features to only add

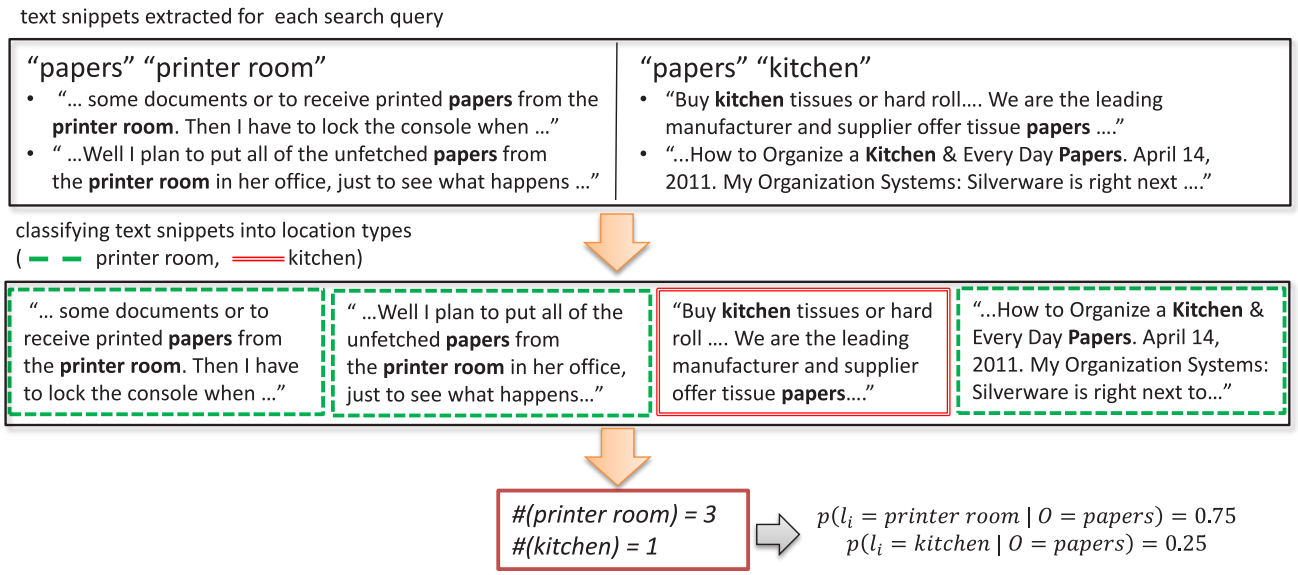


Figure 2: Querying the Web to find the maximum probability location of "papers." In this example, two location types exist: printer room and kitchen. A Web search for "papers," "printer room" and "papers," "kitchen" returns a set of web-pages from which text snippets are extracted. The features of these text snippets are then categorized as one of the location types. The red border indicates that the text snippet was categorized as a "kitchen" and the green border indicates it was categorized as a "printer room." The frequency of the resulting categorization is then used to compute the maximum likelihood probability of each location given a query object.

noise to the learning. Figure 2 shows an example of the text snippet that is found in one of the returned web-pages for query {*"Papers" "Printer Room"*}. Some of the extracted features include: {*documents*, *printed, office, unfetched*}.

To find which location type should be assigned to a web-page, we train an SVM classifier that takes the feature vector for the web-page as an input and learns a multi-class classifier that outputs one of the location types. The classifier is trained using the web-pages resulting from a corpus of predicate instances. These instances can come from human annotation or from a robot interacting with the environment. ObjectEval only needs positive examples of the *locationHasObject* predicate since the SVM is a multi-class classifier over all location types.

Thus, given a query object O, ObjectEval will create a set of predicate instances over all locations l in the environment (Figure 2). If *papers* is given as an object query and we have only two location types in the environment, *printer room* and *kitchen*, then ObjectEval formulates two search queries: {"papers" "printer room"} and {"papers" "kitchen"}. Each search query then returns a set of the highest ranked web-pages from a Web search engine[1]. Text snippets are then extracted for each of the web-pages as shown in the top box of Figure 2. The result of classification on all of the text snippets is shown by two colors: green means that the ObjectEval has classified the text snippet to location "kitchen" and red means that ObjectEval has classified it as "printer room". In total, three of the text snippets are classified as "printer room" and one is classified as "kitchen". This leads to a probability of 0.75 (3/4) for object "papers" to be found in "printer room" and a probability of 0.25 (1/4) for "papers" to be found in the "kitchen."

[1]such as Bing, www.bing.com

Inferring a Plan

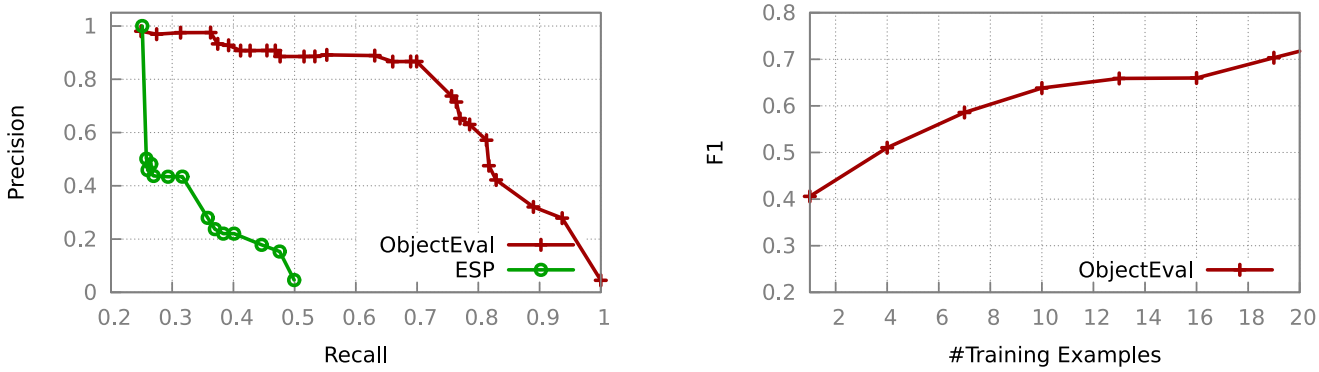
ObjectEval searches over candidate plans to maximize the utility function in Equation 1. We use a beam search with a beam width of 10 and search up to depth $N = 10$ in the search tree. At each step of the search, a new plan step $plan_i$ is added to the overall plan. Each plan step $plan_i$ can visit any of the locations in the test environment to find an object. The beam search disallows loops by preventing plans from revisiting previously visited locations.

Evaluation

ObjectEval is evaluated in three primary ways. First, we query the Web to predict the probability distribution $p(l_i|O)$ described in our model over a set of test objects. Secondly, a set of simulated commands were executed for three floors of an office building. Finally, we have demonstrated ObjectEval on our robotic platform.

Predicting the Location of Objects

We have collected a corpus of 134 unique instances for the predicate *locationHasObject* for the "kitchen," "bathroom," "office," and "printer room" locations. These instances were acquired by asking subjects on Amazon's mechanical Turk to look at pictures of each location type and describe objects



(a) (b)

Figure 3: In (a) is the precision/recall curve for the 45 test predicate instances. In (b) is the F1-score for the 45 test predicate instances when training on a subset of the training dataset.

that tend to reside there. The data is split by randomly choosing 68% of data for training and 32% for testing. ObjectEval is trained and tested by using the first 20 web-pages that are returned by the search engine. Table 1 shows the result for a subset of the test objects. ObjectEval is able to correctly determine the most likely location for most objects. It incorrectly classifies "whiteout" to be found in "bathroom." ObjectEval also chooses "bathroom" as the most likely location for "cup". Although this is correct in some environments (e.g. hotels), we generally expect robot to find "cup" in either a "kitchen" or an "office". The results show that by requesting more specific query such as "coffee cup," ObjectEval will change its classification to the "kitchen."

ObjectEval was then evaluated using precision, recall, and F1 (which is a combination of precision and recall) over this dataset. The ESP baseline replaces web search with a search over tag documents that contain the search terms (von Ahn and Dabbish 2004) in order to provide comparison to (Kollar and Roy 2009). Figure 3(a) shows that the model trained on ESP performs worse than ObjectEval, which likely happens because few locations are tagged in the ESP dataset.

Finally, the speed at which ObjectEval learns was evaluated. Figure 3(b) shows the F1-score of ObjectEval when increasing the number of predicate instances used for the training. The results are obtained by training on a subset of the training instances and evaluating on all of the test instances. The result, somewhat surprisingly, shows that ObjectEval achieves a high F1 value even when it uses a few training examples. For example, it achieves a F1 score of about 60% when it uses only 6 training examples for the training. ObjectEval learns quickly because a single training instance could return thousands or millions of web-pages. For example, the number of documents referencing "papers" and "printer room" is 61,200 according to Google. This result indicates that ObjectEval might be trained even with only a few predicate instances.

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
papers	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**

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

Simulated Experiments

We have created a large simulated environment to evaluate how ObjectEval will search for objects. Since the simulator uses exactly the same procedures as the physical robot, the number of interactions (I) will be exactly the same as on the real robot. In general, when the robot asks for an object, a person must answer two questions and when it is moving between floors (using the elevator) a person must answer five questions.

To simulate the objects present in the building, we have created a semantic map of 290 spaces over three floors of an office building that contain names for objects and locations present in each space. This was done by asking subjects on Amazon's Mechanical Turk to label images of 46 rooms with the location and objects present. These labels were transferred to spaces for which we were not able to acquire images by sampling from the data collected from Mechanical Turk. To test the ability of ObjectEval to search for objects, we have selected 80 object types that were not a part of the training set. ObjectEval was given only the loca-

Approach	Visited locations		Distance		Interactions	
	Mean	Standard Error	Mean	Standard Error	Mean	Standard Error
Baseline	35.8	6.1	69.6	7.2	71.5	12.3
ObjectEval (offline)	14.3	4.3	33.9	4.6	28.7	8.6
ObjectEval (interactive)	10.2	3.8	32.5	4.4	20.5	7.7

Table 2: Average and standard error for the number of visited locations, distance and number of interactions for different approaches. The baseline uses only the terms for interaction I and distance D from Equation 2. ObjectEval (offline) uses batch training and ObjectEval (interactive) is given no training data, but instead uses the presence of objects in locations to update the probability of a location given the object as it performs a search (as from Equation 2).

tion types (e.g., "kitchen" or "printer room") and a map of the environment. For each query object, a random location is chosen as the object delivery destination.

We evaluate ObjectEval in two scenarios: *offline mode* and *interactive mode*. In the offline mode, ObjectEval learns the probability from Equation 4 by using a small dataset of predicate instances consisting of objects and a place where that object can generally be found. In interactive mode, the robot starts performing the find and deliver task in an unknown environment without this training data. By interacting with people, ObjectEval acquires examples of objects and the corresponding place where the object was found. This is then used to learn a model of $p(l_i|O)$ in Equation 4. When the robot finds an object in a location, it adds this to the current set of training instances. ObjectEval will then search the Web and use the resulting web-pages as additional training examples that relate the object to the observed location.

Table 2 shows the results of different approaches that have been used to find objects. The baseline only uses the distance and interaction terms of Equation 2 to greedily generate the next location to visit and uses no semantic information about the environment. ObjectEval maximizes the expected utility Equation 2 in both offline or interactive modes.

There is a clear downward trend in the number of visited locations and number of interactions for ObjectEval when compared with this baseline, indicating that the system is learning about the physical environment. Surprisingly, the interactive mode of ObjectEval achieves better results then the offline version of ObjectEval. Since the training data from Mechanical Turk can be different from the objects and the locations that are found by the robot, the interactive version ObjectEval may have an advantage since it learns the locations of objects directly in the test environment. The offline version starts with a biased set of data (obtained from Mechanical Turk) that may not accurately reflect the real-world. For example, people from Mechanical Turk have annotated "cup" or "glasses" as example of objects that can be found in bathroom. However, in our office environment these objects are expected to be found in offices. By training on these examples, the offline version of ObjectEval would be biased toward finding these objects in bathroom, whereas the interactive version does not have this problem because it only uses training data about objects in the environment.

Although the number of visited locations in Table 2 may seem high, the interactive version of ObjectEval finds 80% of the objects within five locations or less, whereas the baseline finds only 39% in the same five locations. One reason that this term is high is because of a high penalty for choosing the wrong location. For example, if the robot incorrectly classifies "soap" as being in an "office", it will have to search an order of magnitude more locations because the environment contains hundreds of offices, whereas it only contains a few bathrooms.

Figure 4: The number of locations visited by the robot before finding the query object for the interactive mode of ObjectEval (red line) and the baseline (green line). The data is sorted by the number of visited locations per simulation run.

Finally, we have profiled the number of locations visited before finding an object. Figure 4 shows the result when a search for 20 objects is repeated 5 times starting from different initial location to obtain 100 runs. The figure shows that ObjectEval, after having gathered only a few facts, has quickly learned to execute efficient plans to find objects when compared with the baseline approach.

Robot Experiments

We have demonstrated the ability of ObjectEval to find and deliver an object on our mobile office assistant robot. We have queried ObjectEval for a "coffee" and asked it to deliver the object to office "7001." The robot drove to the nearest kitchen and asked for a coffee. When a person came by, they placed a coffee on the robot and the robot returned to 7001 with the coffee. This search can be seen in Figure 1.

Conclusion

In this paper, we have presented an approach, called ObjectEval, which is able to find and deliver objects in real-world environments. We have shown that our system learns to query the Web to evaluate the probability of physical background knowledge that relates objects and locations. In addition, we present an approach for finding and delivering objects that integrates information about object locations from the Web, and interactively learns about the physical environment by getting feedback from humans. We have shown promising results over a baseline approach and have demonstrated our system on a mobile robot navigating in an indoor environment.

Acknowledgments

This research was partly supported by the National Science Foundation award number NSF IIS-1012733. The views and conclusions contained in this document are those of the authors only.

References

Aydemir, A.; Sjöö, K.; Folkesson, J.; Pronobis, A.; and Jensfelt, P. 2011. Search in the real world: Active visual object search based on spatial relations. In *Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA'11)*.

Biswas, J., and Veloso, M. 2012. Depth camera based indoor mobile robot localization and navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, ICRA'12.

Carlson, A.; Betteridge, J.; Wang, R. C.; Jr., E. R. H.; and Mitchell, T. M. 2010. Coupled semi-supervised learning for information extraction. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining (WSDM 2010)*.

Downey, D.; Etzioni, O.; and Soderl, S. 2005. A probabilistic model of redundancy in information extraction. In *IJCAI*, 1034–1041.

Joho, D.; Senk, M.; and Burgard, W. 2011. Learning search heuristics for finding objects in structured environments. *Robotics and Autonomous Systems* 59(5):319–328.

Kollar, T., and Roy, N. 2009. Utilizing object-object and object-scene context when planning to find things. In *Proceedings of the IEEE international conference on Robotics and Automation*, ICRA'09, 4116–4121. Piscataway, NJ, USA: IEEE Press.

Kollar, T.; Samadi, M.; and Veloso, M. 2012. Enabling robots to find and fetch objects by querying the web. In *Proceedings of the Eleventh International Conference on Autonomous Agents and Multiagent Systems*, AAMAS.

Magnini, B.; Negri, M.; Prevete, R.; and Tanev, H. 2002. Is it the right answer? exploiting web redundancy for answer validation. In *In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 425–432.

Makarenko, A.; Williams, S.; Bourgault, F.; and Durrant-Whyte, H. 2002. An experiment in integrated exploration. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 1, 534–539. IEEE.

Meger, D.; Forssén, P.-E.; Lai, K.; Helmer, S.; McCann, S.; Southey, T.; Baumann, M.; Little, J. J.; and Lowe, D. G. 2008. Curious george: an attentive semantic robot. *Robotics and Autonomous Systems* 56:503–511.

Posner, I.; Corke, P.; and Newman, P. 2010. Using text-spotting to query the world. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*.

Rosenthal, S.; Biswas, J.; and Veloso, M. 2010. An effective personal mobile robot agent through symbiotic human-robot interaction. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '10.

Samadi, M.; Veloso, M.; and Blum, M. 2011. Evaluating correctness of propositions using the web. In *Proceedings of the Workshop on Learning by Reading and its Applications in Intelligent Question-Answering*, IJCAI'11.

Sjöö, K.; López, D. G.; Paul, C.; Jensfelt, P.; and Kragic, D. 2009. Object search and localization for an indoor mobile robot. *Journal of Computing and Information Technology* 17(1):67–80. doi:10.2498/cit.1001182.

Soderl, S.; Etzioni, O.; Shaked, T.; and Weld, D. S. July 2004. The use of web-based statistics to validate information extraction. In *AAAI Workshop on Adaptive Text Extraction and Mining*.

Stachniss, C.; Grisetti, G.; and Burgard, W. 2005. Information gain-based exploration using rao-blackwellized particle filters. In *Proc. of robotics: science and systems (RSS)*, 65–72.

Tenorth, M.; Klank, U.; Pangercic, D.; and Beetz, M. 2011. Web-enabled robots. *Robotics & Automation Magazine, IEEE* 18(2):58–68.

Turney, P. D., and Littman, M. L. 2003. Measuring praise and criticism: Inference of semantic orientation from association. *ACM Trans. Inf. Syst.* 21:315–346.

Turney, P. 2001. Mining the web for synonyms: Pmi-ir versus lsa on toefl.

Velez, J.; Hemann, G.; Huang, A.; Posner, I.; and Roy, N. 2011. Planning to perceive: Exploiting mobility for robust object detection. In *Proceedings of the International Conference on Automated Planning and Scheduling*.

von Ahn, L., and Dabbish, L. 2004. Labeling images with a computer game. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '04, 319–326. New York, NY, USA: ACM.

Yamauchi, B. 1997. A frontier-based approach for autonomous exploration. In *Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings., 1997 IEEE International Symposium on*, 146–151. IEEE.