Finding Action Dependencies Using the Crowd

Walter S. Lasecki, Leon Weingard, George Ferguson, Jeffrey P. Bigham
University of Rochester
160 Trustee Rd., Rochester, NY, USA
{wlasecki, weingard, ferguson, jbigham}@cs.rochester.edu

ABSTRACT

Training intelligent systems is a time-consuming and costly process that often limits real-world applications. Prior work has attempted to compensate for this challenge by generating sets of labeled training data for machine learning algorithms using affordable human contributors. In this paper, we present ARchitect, a system that uses the crowd to extract context-dependent relational structure. We focus on activity recognition because of its broad applicability, high level of variation, and difficulty of training systems a priori. We demonstrate that using our approach, the crowd can accurately and consistently identify relationships between actions even over sessions containing different workers and varied executions of an activity. This results in the ability to identify multiple valid execution paths from a single observation, suggesting that one-off learning can be facilitated by using the crowd as an on-demand source of human intelligence in the knowledge acquisition process.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

Keywords

crowdsourcing, planning, activity recognition, constraints

1. BACKGROUND

Crowdsourcing is a type of human computation, which has been shown to be useful in many areas that automated systems find challenging, including: writing and editing [2], image description and interpretation [3, 7], and protein folding. Crowdsourcing leverages human "workers" (usually recruited off a marketplace such as Amazon's Mechanical Turk) who each contribute small pieces of work which typically require only basic skills, resulting in a dynamic, often unreliable source of answers. However, the benefits of this workforce's dynamic nature is that workers can be recruited at any time within seconds [3, 1], allowing ARchitect to work in nearly real-time. Most abstractions obtain quality work by introducing redundant work into tasks so that results are verified at each stage.To reduce the cost of crowd-powered systems, we want to use the crowd to train an automated system, which can then be used in place of the crowd.

K-CAP '13, June 23-26, 2013, Banff, Canada.

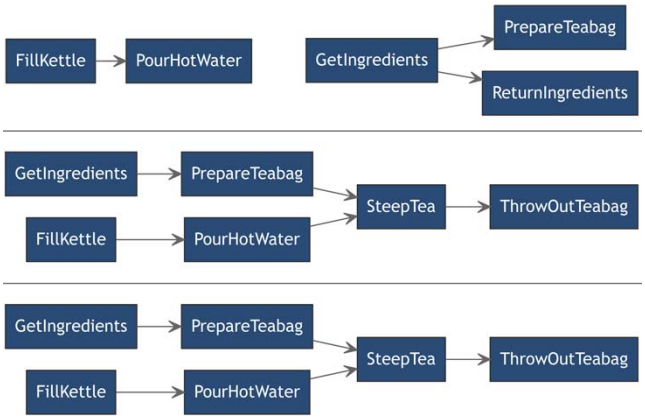

Figure 1: Dependency graph output for the 3 trials we ran. An average of 22.3 workers were asked each question. A threshold of 50% was used on the first two trials, while the last used 40%.

Legion:AR [6] uses the crowd to support and train an activity recognition (AR) system in real-time. Workers generate labels for a video stream, and these labels are used to train a Hidden Markov Model-based AR system. This paper extends the idea of learning from the crowd to include relational knowledge. Extracting generalized knowledge has been looked at by CrowdQ [5], which uses data mining and natural language processing, in addition to the crowd, to extract query semantics and generate templates. Cascade [4] looked at creating taxonomies from posts on Quora. It can extract hierarchies from responses by asking users similarity questions to find subset relationships.

2. SYSTEM

ARchitect begins by dividing the source activity video into smaller segments that contain one action each using Legion:AR [6]. To find the structure of the dependencies between actions, we ask workers to answer a set of questions for each video segment. These questions are automatically generated from the existing labels the system has observed. Workers simply have to answer 'Yes' or 'No' to one or more questions. We found that asking workers 'Is it possible to do [**action_in_video**] before [**action_in_question**] when doing [**high_level_activity**]?', resulted in a balance between the scope of the question and adding of new constraints.

The labels generated using Legion:AR are consistent between multiple executions of different tasks. We present the workers with video of the activity to encourage results that are specific to the instance of an action being observed, which helps to avoid missing details that cause some actions to be prerequisites of others only under in certain cases.

Using the information collected from workers, we can create a directed acyclic graph to represent the relational structure of the actions. For each 'Yes' response, we add 1 to the weight of the directed edge going from the prerequisite action to the given action. The resulting graph captures all of the pairwise constraints between actions. To increase reliability, we use a filtering threshold which removes edges with low agreement. This graph can be used to extract information for an activity recognition system. To calculate the number of valid possible execution paths, we generate a list of all action orderings that do not contain a dependent action as a parent of it's prerequisite. This path count provides a way to measure the potential learning difference between crowd labeling systems and ARchitect.

3. EXPERIMENTS

To test ARchitect, we looked at a common activity recognition domain: household monitoring. The goal of activity recognition in the home is to provide timely, task-relevant information and support to those who need it. For example, prompting systems that keep people with cognitive disabilities on track, or smart homes that detect when to summon help so that older adults can safely live independently longer. These examples require recognizing a whole range of household tasks. In this paper, we observe people making tea in a home kitchen environment (described in more detail below).

In our tests, we use a tea making dataset in which participants were asked to prepare a cup of tea in our kitchen lab. We recruited 3 participants to record a video of them stepping through the process of making tea. Participants were not asked to take a particular course of action. There were 6 distinct actions observed in our 'Make Tea' domain: `FillKettle`, `GetIngredients`, `PrepareTeabag`, `ReturnIngredients`, `PourHotWater`, `SteepTea`, and `ThrowOutTeabag`.

4. RESULTS

To validate ARchitect, we are interested in testing two aspects: (*i*) the crowd's ability to identify dependencies in observed action orderings, and (*ii*) the accuracy and validity of the resulting activity execution paths identified using this approach. The latter allows us to better understand the tradeoff in accuracy versus the cost savings of requiring fewer observations of a given activity to reliably identify it later. We ran 3 sets of trials using the videos of the tea making activity performed by 3 different individuals, collecting between 1 and 5 responses each from 288 unique crowd workers in all. In order to help workers better understand the task that they were being asked to perform, we first asked each worker to do a quick 2-question tutorial in which they received feedback on each of their answers (why it was correct or incorrect). During ours tests, we paid workers between \$0.05 and \$0.20 per task, with an average of \$0.15.

Crowd workers may misunderstand the task, not be able to infer the effects of prior actions, or not be willing to complete the task with high accuracy for the offered price. Thus, we begin by testing how accurately ARchitect can extract individual relationships from a video of a given activity.

Handling noisy data is a necessary step when collecting data from the crowd. In order to use only the action dependencies which we have high confidence in, we filter the data so that a link between two nodes is only included if multiple workers agreed on it. We found that at around 40% agreement we had over 75% in both precision and recall. In addition, we found that at above 50% agreement all identified valid paths were correct.

Due to the question format that we used, the input from workers tended towards not adding links in order to prevent over-constraining the possible solutions. This means that erroneous edges from reliable workers who were uncertain about a particular answer are more rare, but we still need to handle lazy or malicious workers who provide potentially incorrect answers regardless of the understandability of the task. The maximum number of possible valid paths increases exponentially as fewer constraints are included. For Video #1 there were at most 120 distinct paths, for Video #2 there were 720, and for Video #3 there were 120. We found an average of 10.7 possible paths for each of our videos, compared to only 1 without our approach.

5. CONCLUSION

We have presented ARchitect, an activity recognition system training tool that is capable of extracting structural information from video using the crowd by asking workers to identify dependency relationships between actions. Our experiments demonstrate the feasibility of this approach. More generally, ARchitect takes a first step towards using the crowd to provide on-demand human intelligence in the knowledge acquisition process and suggests the this approach could be used to train automated systems from one-off examples. Human intelligence is well-suited to these types of tasks because people can easily apply common sense reasoning to problems that even the current most sophisticated automated systems would struggle with. Our goal in future work is to extract even more detailed semantic information from the crowd.

6. REFERENCES

[1] M. S. Bernstein, J. R. Brandt, R. C. Miller, and D. R. Karger. Crowds in two seconds: Enabling realtime crowd-powered interfaces. In *Proc. of UIST '11*, p.33–42, 2011.

[2] M. S. Bernstein, G. Little, R. C. Miller, B. Hartmann, M. S. Ackerman, D. R. Karger, D. Crowell, and K. Panovich. Soylent: a word processor with a crowd inside. In *Proc. of UIST '10*, p.313–322, 2010.

[3] J. P. Bigham, C. Jayant, H. Ji, G. Little, A. Miller, R. C. Miller, R. Miller, A. Tatarowicz, B. White, S. White, and T. Yeh. Vizwiz: nearly real-time answers to visual questions. In *Proc. of UIST '10*, p.333–342, 2010.

[4] L. B. Chilton, G. Little, D. Edge, D. S. Weld, and J. A. Landay. Cascade: Crowdsourcing taxonomy creation. In *Proc. of CHI '13*, To Appear, 2013.

[5] G. Demartini, B. Trushkowsky, T. Kraska, and M. Franklin. CrowdQ: Crowdsourced query understanding. In *Proc. of CIDR '13*, 2013.

[6] W. S. Lasecki, Y. C. Song, H. Kautz, and J. P. Bigham. Real-Time Crowd Labeling for Deployable Activity Recognition. In *Proc. of CSCW '13*, p.1203–1212, 2013.

[7] L. von Ahn and L. Dabbish. Labeling images with a computer game. In *Proceedings of the conf. on Human factors in computing systems*, CHI '04, p.319–326, 2004.