Crowd Formalization of Action Conditions

Walter S. Lasecki, Leon Weingard, Jeffrey P. Bigham, George Ferguson
University of Rochester
Rochester, NY 14627
{wlasecki, weingard, jbigham, ferguson}@cs.rochester.edu

Abstract

Training intelligent systems is a time consuming and costly process that often limits their application to real-world problems. Prior work in crowdsourcing has attempted to compensate for this challenge by generating sets of labeled training data for machine learning algorithms. In this work, we seek to move beyond collecting just statistical data and explore how to gather structured, relational representations of a scenario using the crowd. We focus on activity recognition because of its broad applicability, high level of variation between individual instances, and difficulty of training systems a priori. We present ARchitect, a system that uses the crowd to ascertain pre and post conditions for actions observed in a video and find relations between actions. Our ultimate goal is to identify multiple valid execution paths from a single set of observations, which suggests one-off learning from the crowd is possible.

Introduction

Decades of artificial intelligence (AI) research has resulted in a range of systems that are capable of reliably generating answers when provided with sufficient prior knowledge. However, training these systems is a time consuming and costly process that often limits their application to real problems. Crowdsourcing has been used as a means of solving problems that automated systems cannot yet handle robustly to generate sets of labeled training data for machine learning algorithms. Recent work has investigated how the crowd can be used to support automated systems that can be deployed with little or no training to begin with, then scale towards being fully automated in the future. This approach still requires learning numerous instances, but solves many scalability problems of the crowd and greatly reduces the cost of developing intelligent systems for real-world use.

In this paper, we seek to move beyond collecting statistical data and explore how to gain logical understanding of a situation using the crowd. We focus on activity recognition because of its broad applicability, high level of variation between instances, and difficulty of training systems a priori. We present ARchitect, a system that uses the crowd to find pre and post conditions for actions observed in a video.

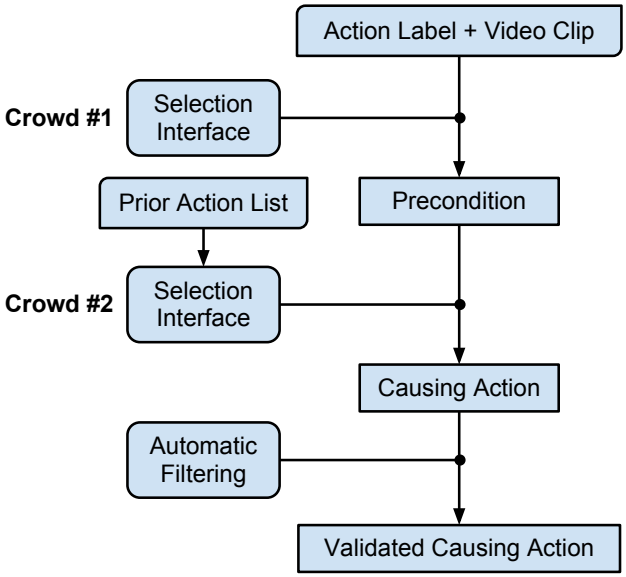

Figure 1: ARchitect's crowd formalization approach.

This allows systems to identify multiple valid possible execution paths from a single observation, suggesting that one-off learning from the crowd is possible.

Background

We present a means of extracting planning pre and post conditions using the crowd. 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 (Bernstein et al. 2010), image description and interpretation (Bigham et al. 2010; von Ahn and Dabbish 2004), and more. Most abstractions obtain reliably high quality work despite the dynamic nature of the crowd by introducing redundancy into tasks so that multiple workers contribute and verify results at each stage (Little et al. 2010; Bernstein et al. 2010). However, this also contributes to difficulties scaling crowd-powered systems to a larger audience.

To help address this problem, prior work has looked at using the crowd to provide training data to an automated system, which can then be used in place of the crowd (Raykar et al. 2010). Legion:AR (Song et al. 2012) is a system that uses the crowd to support and train a deployed activity recogni-

tion system in real-time using crowd-generated labels for a video stream. A learning model can then be trained online using this data. The model enables systems that can gradually scale from fully crowd-powered to fully automated. This paper extends this idea beyond statistical data, to learn relational and logical aspects of the observed activities.

Extracting structural information from the crowd has been looked at in the context of database queries in CrowdQ (Demartini et al. 2013), which uses a combination of query log mining and natural language processing, in addition to the crowd, to extract query semantics and generate query templates that can then be used for lookup searches if a matching query is made in the future. In this paper, we similarly use the crowd to create a more general abstraction, with the goal of eventually creating a crowd-powered interface between human and machine understandable languages.

System

Our algorithm for finding pre and post conditions with the crowd (Figure 1) has four main components. Since this process can be run on sequential actions as they are observed, it can be run online in nearly real-time. The first step is to divide the source activity video into smaller segments that contain one action each. In the general case where it may be impossible to separate actions in this way, we can extract multiple action labels from matching sub-segments, a 'taboo' list such as the one used in the ESP Game (von Ahn and Dabbish 2004) can be used to discourage workers from labeling the same action twice. To find consistent segment times and labels, we use Legion:AR (Song et al. 2012). Legion:AR consists of two steps: the first asks workers to press a button each time they see an action come to an end, and the second asks a potentially different group of workers to propose and vote on labels for each segment. Because Legion:AR is designed to work in real-time, it adds only a few seconds of latency to our condition-extraction process.

Finding Preconditions

For each of the labeled actions, we then find the preconditions that are needed to perform it. We ask workers to select these requirements from a list of crowd-generated options. If the list does not already contain a worker's intended answer, they can add it to the list. Because there are an unbounded number of potential preconditions to any given action, we need to filter the results we get down to only the interesting ones. We do this filtering in two ways: (i) we select only the top k suggestions from the previous step, this prevents a majority of overly-general conditions (i.e. "the Earth must still exist") from being selected because more specific and useful conditions will be more clearly useful to a larger set of workers; and (ii) we only look at preconditions that were caused by actions that we have observed. By looking at the top k choices, we are able to include an incentive mechanism that rewards workers for contributing high-quality answers that are eventually selected for use.

Finding Effects

Workers are then asked to select the most likely cause of the precondition to the action in the video from a known list of potential prior actions, or indicate no such action exists. This works because we are interested in learning relations between different actions, meaning that any precondition which we cannot observe the cause of is out of our scope and assumed to be a prior state of the world.

Finally, since the effects of the n^{th} action are extracted by finding the preconditions of a future action $(n + m)$ that is caused by action n, a final pass is needed to find the final effects. This is handled in the same manner as all of the other cases, except workers are now shown a video of the state following the final action has completed.

Combining Results

The predicates formed by the pre-condition and preceding-action filtering steps are reduced to a useful, well specified, form that can be used to generate a graphical model similar to a plan graph. This graph defines constraints between the ordering of actions, and thus allows us to assess whether or not future sequences of actions are potentially valid instances of the same high-level activity, greatly improving the learning power of the system from a single instance.

Conclusion and Future Work

In this paper we have presented ARchitect, a system that allows the crowd to formalize pre and post conditions to actions observed in a video in nearly real-time. While the systems and approaches that ARchitect is built on have previously been proven to be effective, our next step is to use our current implementation to determine how best to present the task of defining a set of preconditions to workers.

ARchitect allows an automated system to leverage a structured representation of the plan executed in a scene to find multiple valid action execution orders, allowing the system to be trained using one-off examples. Using this approach, our goal is to create deployable intelligent systems can learn on-the-fly in real-world situations, enabling them to work using the crowd today, while scaling towards fully automated in the future.

References

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

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

Demartini, G.; Trushkowsky, B.; Kraska, T.; and Franklin, M. 2013. Crowdq: Crowdsourced query understanding. In *Proc. of CIDR '13*.

Little, G.; Chilton, L. B.; Goldman, M.; and Miller, R. C. 2010. Turkit: human computation algorithms on mechanical turk. In *Proc. of UIST '10*, 57–66.

Raykar, V. C.; Yu, S.; Zhao, L. H.; Valadez, G. H.; Florin, C.; Bogoni, L.; and Moy, L. 2010. Learning from crowds. *J. Mach. Learn. Res.* 99:1297–1322.

Lasecki, W. S.; Song, Y. C.; Bigham, J. P.; and Kautz, H. 2013. Training activity recognition systems online using real-time crowdsourcing. In *Proc. of CSCW 2013*.

von Ahn, L., and Dabbish, L. 2004. Labeling images with a computer game. In *Proc. of CHI '04*, 319–326.