Ashwin: Plug-and-Play System for Machine-Human Image Annotation

Anand Sriraman, Mandar Kulkarni, Rahul Kumar, Kanika Kalra, Purushotam Radadia, Shirish Karande
TCS Research, Tata Research Development and Design Center, Pune, India - 411013
{anand.sriraman, mandar.kulkarni3, kalra.kanika, rahul14, purushotam.radadia, shirish.karande}@tcs.com

Abstract

In this paper, we present an end-to-end machine-human image annotation system where each component can be attached in a plug-and-play fashion. These components include Feature Extraction, Machine Annotation, Task Sampling and Crowd Consensus.

Introduction

Crowdsourcing has been a major research area for the past few years with the rise of online labour marketplaces like Amazon's Mechanical Turk, CrowdFlower, etc. This has led to many algorithms being proposed to tackle problems in crowdsourcing like task allocation, crowd consensus, spam detection, etc. Crowdsourcing has also gained impetus due to its usefulness in generating large-scale annotated data for machine learning. Since re-training learning algorithms can be expensive, algorithms for active learning have also been extensively studied. Systems for collaborative human-machine workflows have also been studied (Russakovsky et. al., 2015). We focus on seeing how such systems can be decomposed and implemented in a plug-and-play fashion.

Every algorithm proposed in the above areas have their advantages depending on the application. Hence, applying the right algorithm for the task is crucial. Towards this goal, we demonstrate a plug-and-play system for collaborative machine-human image annotation where users can upload their own algorithms and choose which ones to use for a specific job. We also include integration with external crowd platforms like Amazon's Mechanical Turk and CrowdFlower through a coordination layer.

System Workflow

Figure 1 shows the entire workflow of our system. The orchestration of this workflow is described below. Throughout this paper we differentiate between the requester (person who wants the images annotated), the researcher (person who uploads programs for different parts of the pipeline) and the workers (people who annotate the images).

Algorithm Upload

Researchers upload their programs which implement an algorithm for one of the machine stages: Feature Extraction, Machine Labelling, Task Sampling, and Crowd Consensus. These programs can be marked Public (visible to all) or Private (visible to only the uploader). Programs uploaded by researchers are first verified by authorized users to make sure they’re safe to run.

Job Configuration

Requesters upload their Job, consisting of images to be labelled, type of label required (class/bounding box) and some training images. They also select which algorithm to use for each machine stage from the available list.

Figure 1 System Workflow

Figure 2 Screenshot of Machine Stage Mapping for Job
**Feature Extraction**
The images are sent to the selected Feature Extractor. This could be a Deep Network (Sharif Razavian et al. 2014) or any other program that extracts image features, as selected by the requester. The output is a feature vector which is stored in the database.

**Machine Labelling - Training**
The training images and their feature vectors are forwarded to the selected Machine Labelling program to train the model. The generated model file is stored on disk and its location stored in the database. A web based API is generated for this trained model for future use.

**Machine Labelling**
Once trained, the machine labelling program is run on all the images uploaded by the requester. The machine labelling program returns the label and the label probability for each image. If the requester wants to improve the results, they can request for a round of crowd annotations.

**Task Sampling (Active Learning)**
When new crowd annotations are required, the label probabilities are sent to the selected Task Sampling program which returns a list of images that should be annotated next. A simple algorithm which could be used here is uncertainty based sampling as proposed by (Cohn et al., 1996).

**Crowd Annotation - Coordination Layer**
A batch of tasks is created for these images and posted to the various external crowd platforms through the Coordination Layer. The coordination layer posts these as a Survey Job where crowd workers come to the system’s URL and work on the custom UI provided by the system. The coordination layer also takes care of generating the survey code for workers to enter and platform specific timers (CrowdFlower has a 30min limit for survey jobs). Supported annotation types by the crowd are: (i) Classification, (ii) Bounding Box, (iii) Object Contour and (iv) Image Comparison (Are two images same?).

**Crowd Consensus**
The crowd annotations are sent to the selected Crowd Consensus program which determines the agreed label, its probabilities and worker reputations for that batch. Algorithms here could be simple majority voting or EM-based approaches like (Welinder and Perona, 2010).

**Machine Re-training**
The labels received from the Crowd Consensus are sent back to re-train the selected machine labelling program and the process continues.

**System Architecture**

Our system allows researchers to upload their algorithm implementations as Python programs. They can upload a ZIP file containing all required files, including any model files. They must specify the main Python program that should be called. The main file must contain methods matching the signature specified below:

- **Feature Extraction**
  - `getModel()` -> model object
  - `getFeatureVector(image,model) : float list`

- **Machine Labelling**
  - `doTrain(images,image_labels) : model object`
  - `doRun(image, model) : image_label`

- **Task Sampling**
  - `getNextSamples(images,image_labels) : image list`

- **Crowd Consensus**
  - `getConsensus(images,crowd_labels) : consensus_labels`

Our server application is developed using Java. The Python programs are run as an external process using Java’s ProcessBuilder. They are started from a new Thread since program execution may not be real-time. The communication between the Java and Python processes happens through temporary JSON files. A wrapper Python script parses the JSON file, executes the selected program and writes the output back to another temporary JSON file. It’s location is written to standard output which is read by Java.

**Future Work**
Currently, we have developed a plug-and-play image labelling platform which can be used to train learning algorithms for image labelling whose models are available as Web APIs. These models can be re-trained on the fly using workers from crowd workers. In the future, we plan on exploring how to extend the design of our system for speech and language.
References

Cohn, David A., Zoubin Ghahramani, and Michael I. Jordan. "Active learning with statistical models." Journal of artificial intelligence research (1996).

Russakovsky, Olga, Li-Jia Li, and Li Fei-Fei. "Best of both worlds: human-machine collaboration for object annotation." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015.

Sharif Razavian, Ali, et al. "CNN features off-the-shelf: an astounding baseline for recognition." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. 2014.

Welinder, Peter, and Pietro Perona. "Online crowdsourcing: Rating annotators and obtaining cost-effective labels." 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops. IEEE, 2010.