HUMAN ACCURACY ANAYLSIS ON THE AMAZON MECHANICAL TURK

JASON CHEN, JUSTIN HSU, STEFAN WAGER

Platforms such as the **Amazon Mechanical Turk (AMT)** make it easy and cheap to gather human input for machine learning problems. The flipside, however, is that some human contributors can be confused or inaccurate. The challenge then becomes to extract trustworthy information from the answers of more or less accurate contributors.

Image-Net is an effort to organize images collected online into hierarchal synonym sets (synsets), such as *dog* or *jigsaw puzzle*. Image-Net then employs **AMT turkers** to label whether the images it has gathered are actually typical representatives of the synset they were grouped under.

Since the turker labels are not 100% accurate, Image-Net currently determines whether an image is correctly labeled by majority vote. In this project, we consider machine learning algorithms that could enable us to identify the more accurate turkers, and to recognize more typical representatives of synsets using the available AMT votes. First, we present an unsupervised EM algorithm that assigns accuracies to individual turkers and assumes that their votes are Bernoulli-distributed. Second, we present a supervised algorithm that identifies a few dozen key images, which we correctly label, and then uses this information to identify trustworthy turkers. The supervised approach produced some very promising results, especially in synsets with ambiguous definitions, such as *bishop* (the chess piece); the unsupervised results were a bit more mixed.

In both our approaches, we make the following assumptions: first, we assume that each of our voters has a consistent accuracy level; second, we assume that the majority is more often correct than not. We then attempt to identify turkers who vote with the majority most often, infer that they are our best labelers, and follow these trusted labellers in ambiguous cases.

Unsupervised Learning Approaches

First Approaches: Naive EM and Skill Scoring. Before settling on our final design, we attempted a Naive EM approach. Briefly, we assume each turker has a weight c_i corresponding to their accuracy. We set the weights to be the log odds of the fraction of the turker's votes that are correct, then we adjusted the labels of the pictures based on the weighted majority vote. Though this seemed promising at first, we encountered one of the main difficulties of this project: too many negative images. Since most of the images the turkers are presented with are negative examples, a turker who votes 'No' to all images will score very well. Thus, these people will have disproportionately high weights, skewing our labels, and leading to mediocre performance.

To remedy this, we considered using a **skill scoring** approach, in which we roughly estimate the fraction of positive images. With this number, we can calculate baseline performances of turkers, if they simply always vote Yes or always vote No. By taking this baseline performance into account, we can decrease the bias. The main problem with this approach is that it isn't clear how to normalize the performances. A turker's maximum performance depends not only on how many images she votes on, but also on the votes she casts, as it is easier to get a No vote correct than a Yes vote. After trying several normalizing schemes, we found that the results were much more promising than the first approach, with poor performance only in identifying false positives.

Final Approach: Specificity and Sensitivity EM. We went back to our first approach, and looked for a model in which voters were expected to score correctly at a high rate when voting No, and to correct for

Date: November 2010.

this bias. Our idea was to have different weights for when turkers voted Yes, and for when turkers voted No. This led to the following model (derivation closely follows [1]):

Let $y = 0, 1$ be the ground truth of a given image, and $y^j = 0, 1$ is voter j labeled the image as. For voter j, we associate parameters $\alpha^j = \Pr[y^j = 1 | y = 1], \beta^j = \Pr[y^j = 0 | y = 0]$, known as the **sensitivity** and the **specificity**, respectively. We also have one more hidden parameter, $p = \Pr[y = 1]$, or the fraction of typical images amongst all images. Now, if $\mathcal{D}$ is the observed votes, $y_i^{u_j}$ is the vote given to picture i by turker u_j, the jth turker to vote on this picture, and θ are the parameters α^j, β^j, p, we have maximum likelihood function:

$$\Pr[\mathcal{D}|\theta] = \prod_{i=1}^{N} \Pr[y_i^1, \ldots, y_i^{R_i} | \theta] = \prod_{i=1}^{N} \Big\{ \Pr[y_i^1, \ldots, y_i^{R_i} | y_i = 1, \alpha] \cdot p + \Pr[y_i^1, \ldots, y_i^{R_i} | y_i = 0, \beta] \cdot (1 - p) \Big\}$$

Independence of votes gives:

$$a_i := \Pr[y_i^1, \ldots, y_i^{R_i} | y_i = 1, \alpha] = \prod_{j=1}^{R_i} \Pr[y_i^j | y_i = 1, \alpha^j] = \prod_{j=1}^{R_i} (\alpha^j)^{y_i^j} (1 - \alpha^j)^{1 - y_i^j}$$

$$b_i := \Pr[y_i^1, \ldots, y_i^{R_i} | y_i = 0, \beta] = \prod_{j=1}^{R_i} \Pr[y_i^j | y_i = 0, \beta^j] = \prod_{j=1}^{R_i} (\beta^j)^{1 - y_i^j} (1 - \beta^j)^{y_i^j}$$

Thus we want to maximize $\Pr[\mathcal{D}|\theta] = \prod_{i=1}^{N} a_i p + b_i (1 - p)$. Denoting the ground truth as $y = [y_1, \ldots, y_N]$, we get that the log-likelihood is $\sum_{i=1}^{N} y_i \ln p_i a_i + (1 - y_i) \ln(1 - p_i) b_i$

E-step: We calculate the expected value of this log-likelihood (this will be our lower bound for the true log-likelihood). Taking the expectation with respect to $\Pr[y|\mathcal{D}, \theta] = Q_i(y)$, we have

$$\mathbb{E}[\ln \Pr[\mathcal{D}, y | \theta]] = \sum_{i=1}^{N} \mu_i \ln p_i a_i + (1 - \mu_i) \ln(1 - p_i) b_i \tag{1}$$

where $\mu_i = \Pr[y_i = 1 | y_i^1, \ldots, y_i^{R_i}, \theta]$. By Bayes, we update

$$\mu_i \propto \Pr[y_i^1, \ldots, y_i^{R_i} | y_i = 1, \theta] \cdot \Pr[y_i = 1] \qquad \Rightarrow \mu_i := \frac{a_i p}{a_i p + b_i (1 - p)}$$

M-step: We wish to maximize the lower bound calculated in the E-step with respect to the parameters α^j, β^j. By setting the α and β gradients of (1) to 0 and update p, we get:

$$\alpha^k := \frac{\sum_{i=1}^{N} \mu_i y_i^k}{\sum_{i=1}^{N} \mu_i} \qquad \beta^k := \frac{\sum_{i=1}^{N} (1 - \mu_i)(1 - y_i^k)}{\sum_{i=1}^{N} (1 - \mu_i)} \qquad p := \frac{1}{N} \sum_{i=1}^{N} \mu_i$$

By reestimating $p = \frac{1}{N} \sum_{i=1}^{N} \mu_i$ every iteration, we can run this until convergence, initializing μ_i to be the majority vote $\mu_i = \Pr[y_i = 1 | y_i^1, \ldots, y_i^{R_i}] = \frac{1}{N} \sum_{j=1}^{R_i} y_i^j$ To calculate our final labels, since μ_i is the probability that image i is typical, we can set a threshold γ and mark images that have $\mu_i > \gamma$ as typical.

Results. The results from our unserpervised approach were somewhat mixed. For some synsets, we managed to perform better than majority vote; on others, though, we generated a large number of false positives. Below are results for the following three synsets, as well as their definitions according to Image-Net:

- *Caterpillar, Cat*: a large tracked vehicle that is propelled by two endless metal belts; frequently used for moving earth in construction and farm work.
- *Bishop*: (chess) a piece that can be moved diagonally over unoccupied squares of the same color.
- *Horseshoe*: U-shaped plate nailed to underside of horse's hoof.

For each synset, we record the number of positives the algorithm recognized, the fraction of images that were unambiguously typical, and the false positive rate. We did not count ambiguous images in either of these categories.

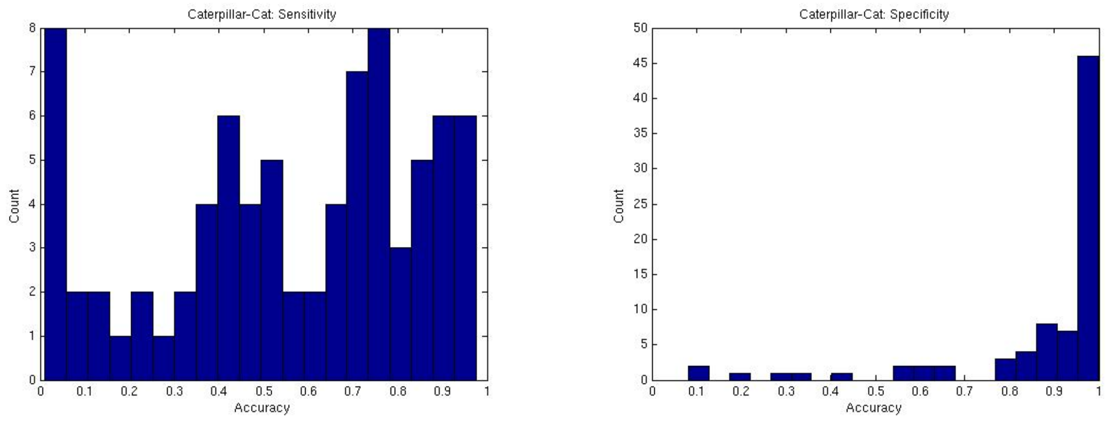

Figure 1. Sensitivity (left) and specificity (right) of users in the *Caterpillar, Cat* synset

Results	Images Recognized	Definitely Typical	False Positives
Cat-Cat: Unsupervised	531	66%	13%
Cat-Cat: Majority Vote	579	45%	39%
Bishop: Unsupervised	280	52%	43 %
Bishop: Majority Vote	167	91%	3%
Horseshoe: Unsupervised	1422	75%	16%
Horseshoe: Majority Vote	1248	72%	13%

Our algorithm worked fairly well for *Caterpillar, Cat.* However, it performed poorly for *Bishop.* We suspect this is because a very small fraction of the images presented to the AMT are in fact typical: under 10%. This means that voting on no-images is very easy, while getting a high score on yes-images is hard. Thus, most turkers have very high specificities but low sensitivities.

Such numbers tend to push split votes toward a yes-decision, since our high specificity and low sensitivity reflect an assumption by the model that false positives are quite unlikely, whereas false negatives are common. If we try to fix this problem by computing only a single credibility for each turker (as we did in our first model), we run into another related problem: all the easy no-votes give most turkers an artificially high credibility.

A challenge for unsupervised learning approaches to the Image-Net dataset is that the interesting images are swamped by a large mass of easy no-images from which it is difficult to learn anything new. In the next section, we present an alternative approach that only learns turker accuracies from contentious, or difficult images.

Supervised Learning Approach

We suspect our work unsupervised learning algorithms ran into difficulties for the following reasons:

- Learning contributor accuracies from all images is not very effective, because interesting, difficult images are swamped by easy images on which everyone votes correctly.
- Learning contributor accuracies from the most difficult images in an unsupervised way can be dangerous, since even a large majority can be wrong on difficult images.

One way to overcome these problems is to automatically find the most difficult and contentious images, and then have an accredited expert provide the ground truth on these images. Though this seems to defeat the purpose of using AMT input, we think that this approach is valuable if a researcher really needs accurate labels on a set of tens of thousands of images and has time to manually label a few dozen images per synset to train the supervised learning algorithm. After these labels are set, the full collection of AMT votes can then be used effectively.

Our idea is an example of a **collaborative filtering** approach, in that we estimate of how each turker might have voted on our small labeled training set by comparing them to turkers that actually voted on our training set. We then compute weights for each turker using a Bernoulli maximum likelihood estimate, and infer labels for our images using a weighted majority vote.

Specifically, we first pick 16 images with a negative majority vote and 16 images with a positive majority vote, while maximizing the quantity (#yes-votes $\times$ #no-votes) for each image. These are our contentious images form our **key images**, which we label manually: for each key image k_l we provide a definite label $L(k_l) = 0, 1$.

To compute weights for all turkers, we first need to to obtain predictions $\tilde{A}_{ik_l}$ for how each turker i would have voted on key image k_l. Clearly, if i voted on k_l, then $\tilde{A}_{ik_l} = A_{ik_l}$. However, if i did not vote on k_l, we need to somehow guess how i would have voted on k_l. To do this, we first compute the **similarity coefficient** u_{ij} for all turkers i and j:

$$u_{ij} = \frac{\sum_{k \in images} A_{ik} A_{jk}}{\sum_{k \in images} |A_{ik} A_{jk}|}$$

Assuming that turkers $\{l_j\}$ voted on key image k_l, we use these parameters to give a guess for the vote of $i \notin \{l_j\}$ on k_l: $\tilde{A}_{ik_l} = \sum_j u_{il_j} A_{l_j k_l}$. We are now ready to compute weights for each turker. Assuming that the votes by turker i are Bernoulli-distributed such that they are accurate with probability c_i, we get the following experssion to maximize:

$$\prod_{i,k_l} c_i^{\max(L(k_l) A_{ik_l}, 0)} \cdot (1 - c_i)^{\max(-L(k_l) A_{ik_l}, 0)}$$

With a little math (see milestone), we get that $c_i = \frac{\sum_{k_l} L(k_l) A_{ik_l}}{\sum_{k_l} |L(k_l) A_{ik_l}|}$. Taking the log of the likelihood function, we can also find additive weights $w_i = \log(\frac{c_i}{1 - c_i})$. We do not let additive weights drop below zero; instead, we ignore turkers with negative weights. Having computed additive weights for each turker as described, we proceed to a weighted majority vote with which we infer labels for the remaining images.

Results. The results using our supervised algorithm were quite promising. Especially for very confusing synsets, such as *Caterpillar, Cat*, our algorithm did a good job at weeding out turkers working under a wrong definition. Below are some detailed results; the Yes and No on the second row indicate the decisions by our algorithm.

	Majority Vote No		**Majority Vote Yes**	
Caterpillar Cat	**No**	**Yes**	**No**	**Yes**
Total Number of Images	12232	159	297	282
Sample Size	100	100	100	100
Number of Definite Positive	2	55	8	78
Number of Ambiguous	3	38	13	20
Number of Definite Negative	95	7	79	2

Many images, such as vehicles with wheels manufactured by Cat, or vehicles with tracks but not directly useable for farming, were ambiguous as to whether they should be considered to represent *Caterpillar, Cat* or not; we labeled these ambiguous. The sample size row indicates how many images we looked at in each category to evaluate our algorithm. Evidently, in all the cases where our algorithm disagreed with the majority vote, it had less than 1 chance in 10 of making a non-ambiguous mistake.

From the point of view of Image-Net, which is trying to accumulate a large and accurate database of images, the two most important metrics are the number of positive images recognized, and the false positive rate. We provide such statistics below for the 3 synsets we worked with:

Results	**Images Recognized**	**Definitely Typical**	**False Positives**
Cat-Cat: Supervised	464	70%	4%
Cat-Cat: Majority Vote	579	45%	39%
Bishop: Supervised	190	89%	0%
Bishop: Majority Vote	167	91%	3%
Horseshoe: Supervised	1259	73%	12%
Horseshoe: Majority Vote	1248	72%	13%

Our algorithm performed quite well on both *Caterpillar, Cat* and *Bishop.* We notice that both of these synsets have somewhat confusing definitions: some might mistakenly believe that the first applies to larvae of butterflies, and the second to Catholic clergy. Our hypothesis is that our algorithm managed to eliminate confused voters, and thus clean up the vote dataset.

Our algorithm does not, however, perform nearly as well on *Horseshoe*. The difficulty in picking good images of horseshoes is not so much due to confusing definitions, but rather in learning what to do with limit cases such as horseshoe shaped necklaces. These results suggest that our algorithm is good at picking out confused turkers, but not so good at picking out turkers without a keen attention to detail.

As support for the hypothesis that our algorithm works essentially by eliminating confused labellers, consider the following numbers: With *Caterpillar, Cat*, we gave 20 out of 80 turkers a weight of 0, and with *Bishop*, 10 out of 28 voters got no weight. On the other hand, with *Horseshoe*, we only gave 4 out of 156 turkers a weight of 0. This suggests that we did poorly on the latter synset because we couldn't identify the turkers who were consistently wrong on a majority of the key images.

Discussion and Further Works

Our results show that machine learning techinques can successfully be applied to improve the value of votes gathered on AMT. Our supervised approach, in particular, makes it easy for a researcher who is willing to spend about a minute labelling images from a synset to eliminate data from turkers who do not share her understanding definitions of the synsets.

Our unsupervised approach presents some interesting challenges. The central difficulty is that, for many synsets, the number of bad images swamps the number of interesting images. If we give turkers a simple Bernoulli accuracy, the high number of trivially bad images makes this accuracy artificially high. Conversely, if we try to give each turker a separate sensitivity and specificity, we end up giving each turker an extremely high specificity. This results in unwanted false positives, since having very high specificities makes it unlikely that any image having received many yes-votes could be a negative.

The success of the supervised approach suggests that it is most useful to infer turker accuracies from their performance on the most difficult images. An interesting problem for further study would be to work on unsupervised learning algorithms that focus more closely on contentious images. One way this could be implemented is by ignoring all images with unanimous votes while learning accuracies. Another approach could be to introduce a parameter for the difficulty of each image–the probability of a turker making a mistake could then depend both on her accuracy and on the difficulty of the image. See, for example, [2] for an idea similar to the latter.

Our results show that there is much promise and much room for improvement in automatic AMT vote analysis. Since AMT is one of the most promising ways of collecting machine learning data on a large scale, it is critical that we find good ways of analyzing AMT data as efficiently as possible. We look forward to seeing what approaches will surface in the literature over the next years.

References

[1] V. Rayker, et al. *Learning From Crowds*, Journal of Machine Learning Research, **11** (2010), 1297–1322
[2] J. Whitehill, et al. *Whose Vote Should Count More: Optimal Integration of Labels from Labelers of Unknown Expertise*, Advances in Neural Information Processing Systems (forthcoming), 2009.