Improving workflow, reducing speed, and achieving human-like understanding in data annotation

Van Trinh 2024

Data annotation is a must-have for training AI models. With a focus on improving labeling, models achieve high levels of accuracy and precision in performing their intended tasks. On the other hand, models progressively degrade and correcting their mistakes may require much more time and resources than initially intended. Below is a review based on published articles on some of the many tools available for improving the labeling process and thus enhancing AI model performance.

I. Workflow designs

Human-assisted computation algorithms, data labeling in particular, increase performance when there's a focus on workflow design. Without it, there is an increased risk of error and the quality of work in the labeling process may be affected. Below are several design patterns studied in the published literature, and how they may be applied to AI training projects.

1. Implement a multi-stage process

There are multiple ways to conduct a multi-stage workflow, and one of which is Find-Fix-Verify. Find-Fix-Verify is a design pattern used by Soylent, a tool designed by MIT researchers to be embedded in Microsoft Word to shorten text while preserving the content of the text. When a user selects an area of text in Word and presses a button in the Soylent ribbon tab, Soylent launches a series of tasks on the Amazon Mechanical Turk platform. This platform recruits human workers to do the tasks and subsequently outputs the results back to Soylent, which then computes the combination of trimmings by humans that most closely match the desired text length given by the user (Bernstein et al., 2010).

Find-Fix-Verify splits the tasks into three stages, completed by three different groups of workers. “Find” workers identify areas to shorten in the text, “Fix” workers suggest ideas for revisions, and “Verify” workers vote to choose option (Figure 1) (Bernstein et al., 2010). This design has more accuracy compared to letting an individual worker perform all three stages for each task.

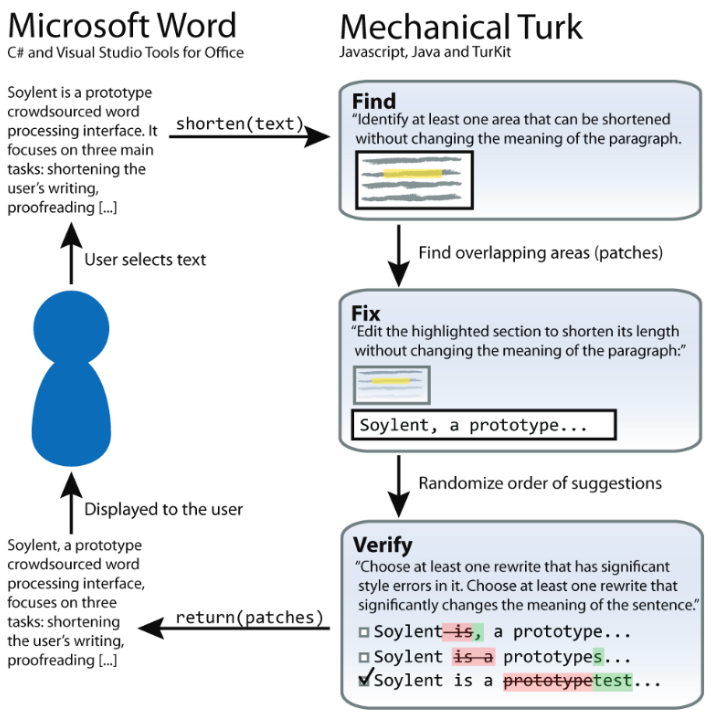
Microsoft Word
C# and Visual Studio Tools for Office
Mechanical Turk
Javascript, Java and TurKit
Soylent is a prototype crowdsourced word processing interface. It focuses on three main tasks: shortening the user's writing, proofreading [...]
shorten(text)
Find
"Identify at least one area that can be shortened without changing the meaning of the paragraph.
User selects text
Find overlapping areas (patches)
Fix
"Edit the highlighted section to shorten its length without changing the meaning of the paragraph:"
Soylent, a prototype...
Randomize order of suggestions
Displayed to the user
Verify
"Choose at least one rewrite that has significant style errors in it. Choose at least one rewrite that significantly changes the meaning of the sentence."
Soylent ~~is~~, a prototype...
Soylent ~~is a~~ prototypes...
✓ Soylent is a ~~prototype~~test...
Soylent, a prototype crowdsourced word processing interface, focuses on three tasks: shortening the user's writing, proofreading [...]
return(patches)

Figure 1. Find-Fix-Verify workflow identifies areas to change, suggests revisions, and votes to choose the best option. (Bernstein et al., 2010)

2. Breaking down tasks: Partition-Map-Reduce

Partition-Map-Reduce pattern suggests that instead of viewing workers separately, you can also view each worker as node in a coupled distributed computing system, in which each worker is analogous to a computer processor that can solve a task requiring human intelligence (Kittur et al., 2011). Therefore, we can apply distributed computing solutions to managing human-assisted computation. MapReduce was inspired by functional programming languages in which a large array of data is processed in parallel through a two-step process: first, key/value pairs are each processed to generate a set of intermediate key/value pairs (the "Map" phase). Next, values with the exact same intermediate keys are merged (the "Reduce" phase). Map and Reduce processes may be nested. As a result, tasks that do not depend on each other are broken down, local improvements can be made with more ease compared to keeping tasks in chunks, and the process is more streamlined.

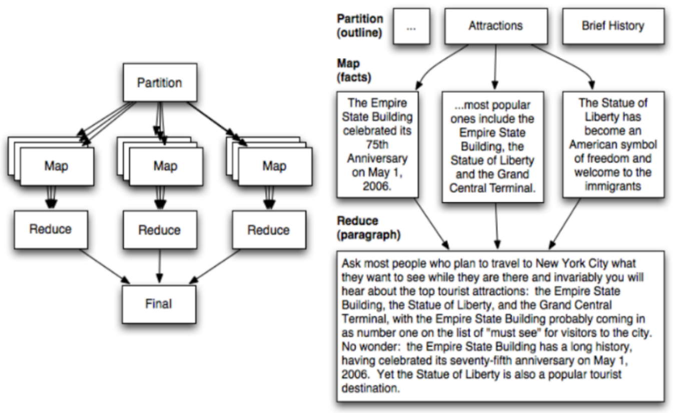
Partition
Map
Map
Map
Reduce
Reduce
Reduce
Final
Partition (outline)
...
Attractions
Brief History
Map (facts)
The Empire State Building celebrated its 75th Anniversary on May 1, 2006.
...most popular ones include the Empire State Building, the Statue of Liberty and the Grand Central Terminal.
The Statue of Liberty has become an American symbol of freedom and welcome to the immigrants
Reduce (paragraph)
Ask most people who plan to travel to New York City what they want to see while they are there and invariably you will hear about the top tourist attractions: the Empire State Building, the Statue of Liberty, and the Grand Central Terminal, with the Empire State Building probably coming in as number one on the list of "must see" for visitors to the city. No wonder: the Empire State Building has a long history, having celebrated its seventy-fifth anniversary on May 1, 2006. Yet the Statue of Liberty is also a popular tourist destination.

Figure 2. Partition-Map-Reduce workflow design (Kittur et al., 2011).

II. Managing AI models.

1. Algorithm to reduce labeling speed while still maintaining quality.

Data labeling, the process of tagging or marking datasets, can be approached through the lens of classification - dividing data with different labels or classes. Binary classification models predict a binary outcome, that is, one of two classes. For example, in a yes or no question, there are only two outcomes, “yes” and “no”. Multi-class classification models predict one class from three or more classes, used to perform text and image disambiguation, recognition, identification, and various other purposes.

In data annotation projects, many tasks compare between responses A and B to output a rating, such as: “A and B are the same”, “A is slightly better than B”, “A is much better than B”, etc. Each rating category can be viewed as a class. Thus, a multi-class classification model can be used to predict which class of rating is the most likely for a certain task. A recent study by researchers at Stanford and Yahoo Labs (Krishna et al., 2016) proposes a class-optimized approach for multi-class classification built upon a binary classification technique that reduces data labeling speed by an order of magnitude while still maintaining the same quality. First, let’s discuss the class-optimized approach for multi-class classification, a simple method that may be applied immediately to data annotation projects. Afterwards, we’ll go into the details of the binary classification technique.

Theoretically, all multi-class classification can be broken down into a series of binary verifications. For example, if there are N classes, there can be N binary questions of whether an item is in each class. Given a list of items, one item can be classified at a time by asking each class or bucket a yes-or-no question. After every iteration, all items classified for a particular class are removed. The rest of the items can be classified for the next class. The order in which classes are considered should not matter when all the classes contain an equal number of items. A simple baseline approach would choose a class at random and attempt to detect all items for that class first. However, if classes end up with uneven sizes, such a method would require more time. Consider the case where items are to be classified into 5 classes. The sizes of the classes are initially unknown, but at the end of the process, one class has 1000 items, while all others have 10 items each. In the worst case, if items that belong to the class that ends up having 1000 examples are classified last, each item would go through the interface 5 times, once for every class, for a total of 5000 times (Figure 3). Instead, if all items that belong to this class are detected first, those 1000 images would only go through the interface once. With this intuition, the authors propose a class-optimized approach that classifies the class with the most items first. The number of items classified at every iteration is maximized, reducing the total number of binary verifications. The authors demonstrate this approach by classifying a dataset of 2,000 images with 10 categories, where each category contains 100 to 250 examples. The class-optimized approach completes the task at a speed of 11,700 seconds, while the baseline approach does so in 12,342 seconds, a difference that will likely be widened as the dataset grows larger. One caveat is that in the study, the identity of the class with the most items is known in advance. For a dataset with unknown sizes, it might be good to do a run to first gauge the frequency of classes, which can then be used to inform future runs.

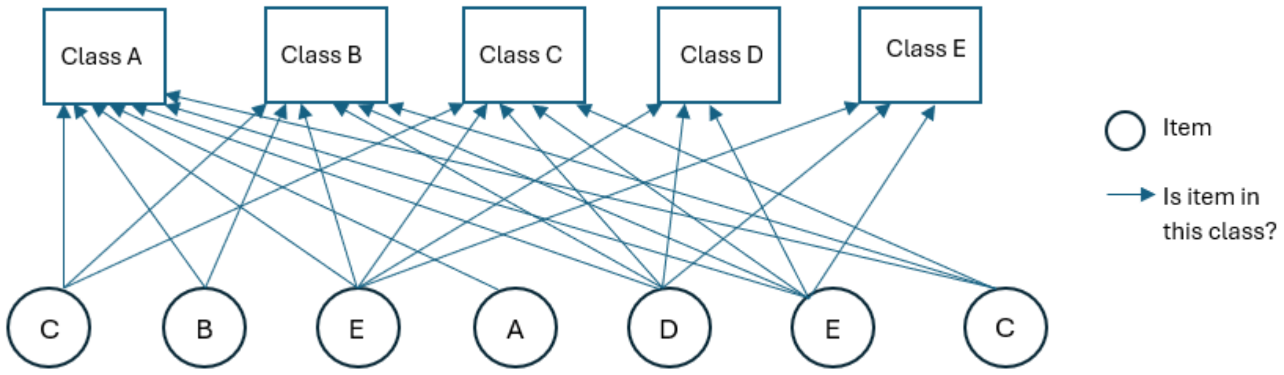

Figure 3. Each item queries each class until it finds its own class and is removed. Class sizes are initially unknown. If items in the largest class are classified last, all of them must query all other classes. Classifying the class with the most items first should reduce the total number of queries.

Binary classification technique proposed by Krishna et al. (2016) can potentially be applied to future data annotation projects. To reduce cost, many previous studies have examined the tradeoff between speed and accuracy in the annotation process. Some methods estimate speed and accuracy to jointly model errors, other methods vary speed and accuracy to calculate the tradeoff between the two, and another predicts the redundancy of non-expert labels to match the level of expert labels (Wah et al., 2014; Branson et al., 2010; Wah et al., 2011). Below is a simple yet powerful method to reducing speed while still maintaining the highest quality.

Krishna et al. (2016) first discusses how current methods of evaluating annotation work usually punish all errors. Meanwhile, in their study, errors are fully expected and even used to predict the correct outcomes. Their method encourages a margin of human error in the interface that is later rectified by inferring true labels algorithmically.

In the past, studies have explored different ways to use workers in a model: workers themselves have been modeled as projections into open space, their skills have been modeled in a generative method, and work quality has been modeled in an unsupervised method (Whitehill et al., 2009; Zhou et al., 2012; Ipeirotis et al., 2010). This study proposes a method not requiring many algorithms yet allowing the prediction of correct outcomes from errors: creating a normal probability distribution for each keypress and then modeling the keypress latency to predict the correct labels.

In the study, each worker sees an image that is displayed for 400 ms alongside a binary question. The worker presses the space bar when the answer is “yes” to the question. The worker is only given 400 ms to react, so errors are expected. There is a delay between when the image appears and when the worker presses the space bar. Given a keypress, this delay is unknown. Thus, each keypress is treated as a normal probability distribution within 400 ms before the keypress (Figure 4).

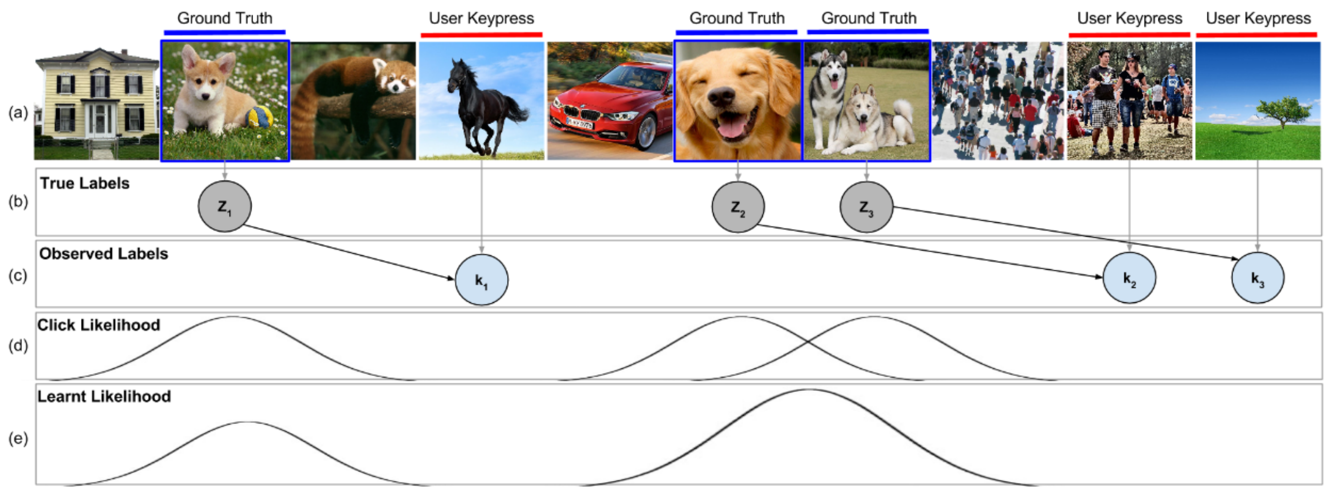

Figure 4. (a) Images are shown, and workers react whenever they see a dog. (b) The ground truth dog images are the true labels. (c) The workers’ keypresses occur several images after the dog images have already passed. These keypresses are recorded as the observed labels. (d) Each keypress is modeled as a delayed Gaussian to predict (e) the probability of an image containing a dog from these observed labels. (Krishna et al., 2016)

Because of the worker’s reaction delay, the data from each worker has much uncertainty. The same set of images is shown to multiple workers in random orders, and independent sets of keypresses are collected. This randomization allows for unbiased estimates of probabilities and cleaner signal in aggregate during majority voting, a previously studied method that this new method is compared to.

The goal is to translate keypresses that may be erroneous into the identifications of positive items. To do so, Krishna et al. (2016) calculates the probability that a particular image is correctly identified given a worker's keypress. Next, the mean and the variance of the distribution of each keypress are used to train a model. The probabilities across several workers are summed up to calculate the total probability for each image. All images above a certain threshold are set as "positive", that is, correctly identified. This threshold is a hyperparameter in the model and can be tuned.

Here are the model details. A set of items $I = \{I_1, \ldots, I_n\}$ are sent to W workers in a different random order for each worker. From each worker w, a set of keypresses $C^w = \{C_1^w, C_2^w, \ldots, C_k^w\}$ is collected, where $w \in W$ and k is the total number of keypresses from worker w. The aim is to calculate the probability of an item being correctly identified given a set of keypresses, $P(I_i \mid C^w)$.
Using Bayes' theorem, the likelihood that an item I_i is positive given the set of key presses C^w is:

$$P(I_i \mid C^w) = \frac{P(C^w \mid I_i)P(I_i)}{P(C^w)}$$

- $P(C^w) = \prod_k P(C_k^w)$ is the probability that keypresses exist, i.e. the probability that a set of items has keypresses from worker w.
  $\prod_k P(C_k^w) = P(C_1^w) \times P(C_2^w) \times \ldots . \times P(C_k^w) = P(C_k^w)^k = constant$, assuming that these probabilities are equal, i.e. it's equally likely that a worker reacts to any item.
- $P(I_i)$ is the probability of an item being correctly identified in any case. Using the law of total probability, the probability of correctly identifying item I_i in any case is:

  $$\begin{aligned} P(I_i) &= P(I_i) \times 1 = P(I_i) \times [P(C^1) + P(C^2) + \cdots + P(C^w)] \\ &= P(I_i) \times P(C^1) + P(I_i) \times P(C^2) + \cdots + P(I_i) \times P(C^w) \\ &= \sum_{w=1}^{W} P(I_i \mid C^w)P(C^w) \end{aligned}$$

  $P(I_i)$ can be a constant, or it can be an estimate from a machine learning algorithm specific to a domain. For example, if the goal is to scale up a dataset of "dog" profiles, one could use a small set of known "dog" profiles to train a binary classifier and use that to calculate $P(I_i)$ for all the unknown profiles. For images, the authors use a pretrained convolutional neural network to extract image features and train a linear support vector machine to calculate $P(I_i)$.
- $P(C^w \mid I_i)$ is a set of independent keypresses for item I_i:

$$P(C^w | I_i) = P(C_1^w, C_2^w, \ldots, C_k^w \mid I_i) = \prod_k P(C_k^w \mid I_i)$$

Each keypress C_k^w is modeled as a Gaussian distribution $\mathcal{N}(\mu, \sigma)$ given a positive item. The authors train the mean μ and variance σ by running rapid crowdsourcing on a small set of items for which the positive items are already known.

The model works by treating each keypress as creating a Gaussian "footprint" of probability of an image being correctly identified about 400ms before the keypress (Figure 4). The alignment of the peak of the Gaussian distribution under each ground truth image is arbitrary. Now that there are a set of probabilities for each item, the next step is to decide which ones should be classified as being correctly identified. Each set of items I is ordered according to likelihood of being in the correctly identified class $P(I_i)$. All items above a certain threshold are set as correctly identified. This threshold is a hyperparameter that can be tuned to trade off precision versus recall. In total, this model has two hyperparameters: (1) the threshold above which images are classified as correctly identified and (2) the speed at which items are displayed to the user. Both hyperparameters are modeled in each topic, for example, image verification or sentiment analysis. For a new topic, first, an estimate is obtained regarding how long it takes to label each item in the conventional setting with a small set of items. Next, the speed at which each item is displayed is reduced in portions until the model is unable to achieve the same precision as the untimed case.

To evaluate this method's performance, Krishna et al. (2016) examines its precision and recall values. Precision is the proportion of true positives among true and false positives. Recall is the proportion of true positives that are correctly identified. As an example, when labeling a dataset such as that of ten thousand articles about housing, precision is often the highest priority: articles labeled as on-topic by the system must in fact be about housing. Recall, on the other hand, is of lower priority, because there is often much unlabeled data: even if the system misses some on-topic articles, the system can label more items until it reaches the desired dataset size. Thus, the goal of this approach is to produce high precision at high speed, sacrificing some recall if necessary. Indeed, this approach achieves very high precision - the same as the approach of punishing all errors during annotation (Table 1). This approach also achieves the speed of about 6 to 11 orders of magnitude times more than the known approach.

Task		**Majority voting**			**Krishna et al. (2016)**			**Speedup**
		Time (s)	*Precision*	*Recall*	*Time (s)*	*Precision*	*Recall*	
Image Verification	Easy	1.50	0.99	0.99	0.10	0.99	0.94	**9.00×**
	Medium	1.70	0.97	0.99	0.10	0.98	0.83	**10.20×**
	Hard	1.90	0.93	0.89	0.10	0.90	0.74	**11.40×**
	All Concepts	1.70	0.97	0.96	0.10	0.97	0.81	**10.20×**
Sentiment Analysis		4.25	0.93	0.97	0.25	0.94	0.84	**10.20×**
Word Similarity		6.23	0.89	0.94	0.60	0.88	0.86	**6.23×**
Topic Detection		14.33	0.96	0.94	2.00	0.95	0.81	**10.75×**

Table 1. Image verification: “easy” – dogs, “medium” – person riding motorcycle, “hard” – eating breakfast. Sentiment analysis – find positive tweets, word similarity – find synonyms of a word, topic – match articles with topics, such as “housing” or “gas”. Speedup column is written in terms of order of magnitude M in the form $M = 10^n$. E.g. 9.00× speedup means $M = 10^n = 9.00$, thus $n = \log_{10} 9.00 = 0.95$. Similarly, 10.20× speedup means $M = 10^n = 10.20$, thus $n = \log_{10} 10.20 = 1.009$. Subsequently, approximately an order of magnitude increase in speed was achieved (Krishna et al., 2016).

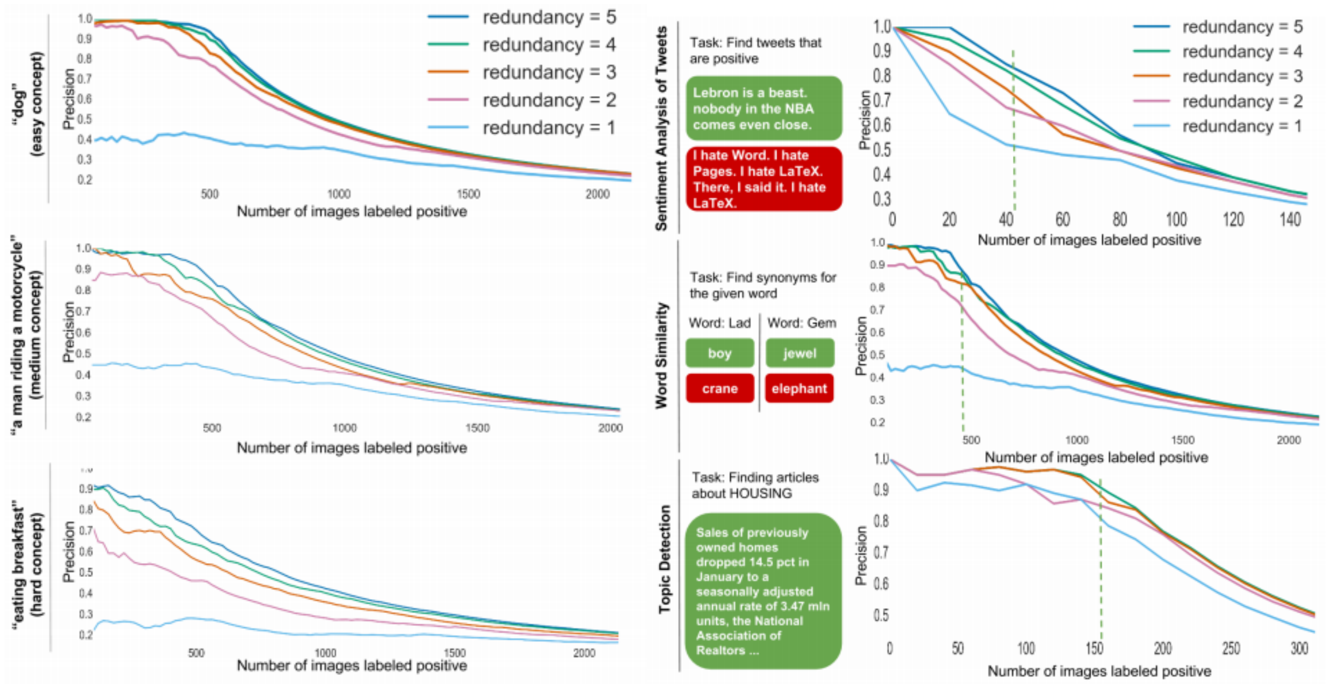

Figure 5. Precision of the method in binary classification tasks (Krishna et al., 2016).

The cognitive load on workers was measured using the NASA Task Load Index (TLX) [12] on all tasks, including image verification. TLX measures the perceived workload of a task. A survey was run on 100 workers who in majority voting approach and 100 workers who used Krishna et al. (2016) technique across all tasks. The cognitive workload for their method was slightly elevated but not statistically significant (61.1 vs. 70.0).

This method requires few algorithms yet achieves good results enough to be applied to many annotation projects for the purpose of increasing speed while still maintaining quality, as well as turning errors into correct labels. We can do so by collecting the amount of time a worker does each task, treating this duration as a Gaussian distribution, and training a model to predict the correct labels using this duration.

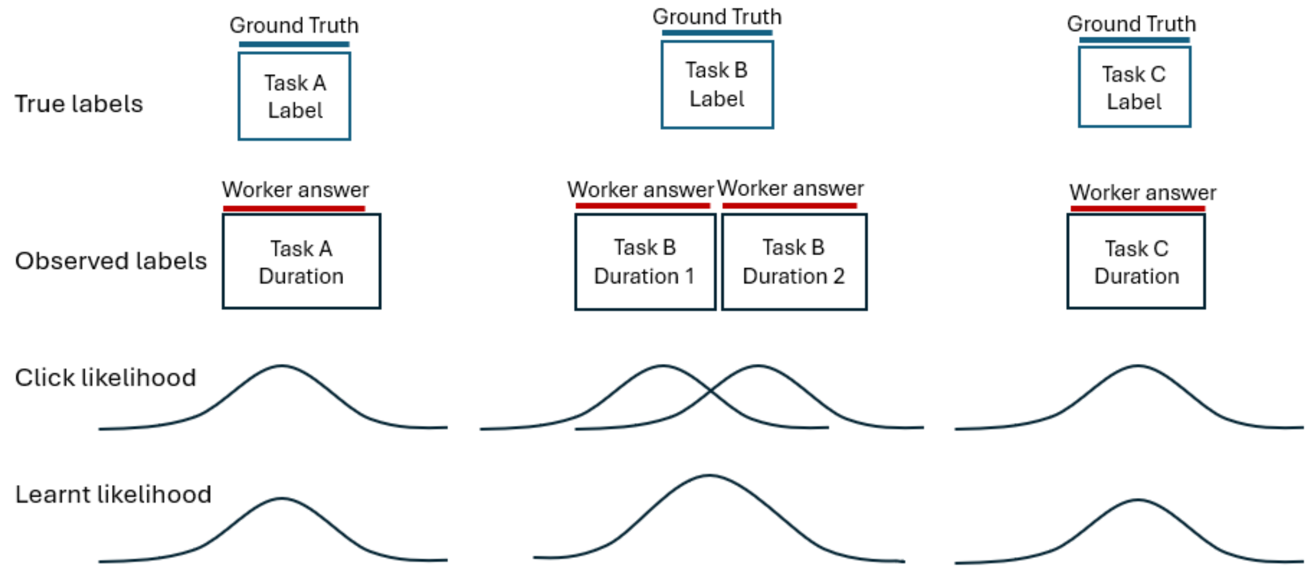

Figure 6. Applying Krishna et al. (2016) method to general data labeling tasks.

First, we need a subset of ground truth labels. These ground truth labels serve two purposes. The first purpose is to filter out low-quality work. Tasks with ground truth labels will be mixed with task of unknown labels. Workers who get zero of the correct labels over a period of time will be rejected. The second purpose is to train a model to calculate a duration time given a correct label. This value is the $P(I_i)$ value discussed above. The ground truth labels can be created and verified by annotators, admins, and reference materials.

Second, we need a subset of labels for tasks with unlimited timing. This subset is used to determine the lowest duration possible given a task. The workers are allowed to work without a time limit to determine an initial average duration. This duration is then successively reduced to until workers start making mistakes. This is the error point, which will later be used in the model.

Third, we set up the model. Here are the model details. A set of labels $L = \{L_1, \ldots, L_n\}$ are sent to W workers. The labels in the set are in a different random order for each worker. From a worker w, a set of durations $D^w = \{d_1^w, d_2^w, \ldots, d_k^w\}$ is collected, where $w \in W$ and k is the total number of keypresses from worker w. The aim is to calculate the probability of a label being correct given a set of durations, $P(L_i \mid D^w)$.

Using the law of total probability: if D_1^w, D_2^w,…, D_k^w are partitions of the sample space S, then the probability of correctly identifying label L_i in any case is:

$$P(L_i) = P(L_i) \times 1 = P(L_i) \times [P(D_1^w) + P(D_2^w) + \cdots + P(D_k^w)] = \sum_w P(L_i \mid D^w)P(D^w)$$

According to Bayes' theorem, the likelihood of a label L_i being correct given the set of key durations D^w is:

$$P(L_i \mid D^w) = \frac{P(D^w \mid L_i)P(L_i)}{P(D^w)}$$

- $P(D^w) = \prod_k P(D_k^w)$ is the probability that durations exist, i.e. the probability that a set of labels has durations from worker w.
  $P(D^w) = P(D_1^w) \times P(D_2^w) \times \ldots . \times P(D_k^w) = P(D_k^w)^k = \mathit{constant}$, assuming that these probabilities are equal, i.e. it's equally likely that a worker reacts to any label.
- $P(L_i)$ is the probability of a label being correct in any case. $P(L_i)$ models the estimate of label l_i being correct. It can be a constant, or it can be an estimate from a machine learning algorithm specific to a domain. There are several options to calculate $P(I_i)$, such as using a linear support vector machine.
- $P(D^w \mid L_i)$ is a set of independent durations for label L_i:

$$P(D^w \mid L_i) = P(d_1^w, d_2^w, \ldots, d_k^w \mid L_i) = \prod_k P(D_k^w \mid L_i)$$

  Each duration d_k^w is modeled as a Gaussian distribution $\mathcal{N}(\mu, \sigma)$ given a correct label. We will train the mean μ and variance σ by using annotators on a small set of labels for which the correct labels are already known.

2. Achieve human-like understanding.

A second problem for AI to overcome is machine doesn't understand concepts and struggle to adapt or generalize to a new situation. Machines are very good at deducing patterns, but not so good at understanding abstractions and concepts. This is because supervised machine learning often focuses on training and test data, which requires function-fitting interpolation. Even if a model gets perfect accuracy on the test data set, it can still overfit. In addition to the ability to interpolate, humans and most other animals are able to extrapolate—that is, to adapt what they have learned to diverse situations. This adaptability is accomplished via the abilities to build representations of concepts, and to make analogies mapping these representations to new situations (Mitchell 2020).

How do we build this ability to extrapolate into AI models? A recent study (Johnston and Fusi, 2023) shows learning of multiple tasks, aka multi-task learning, causes these representations to emerge. The study describes this method using classification tasks (Figure 1a).

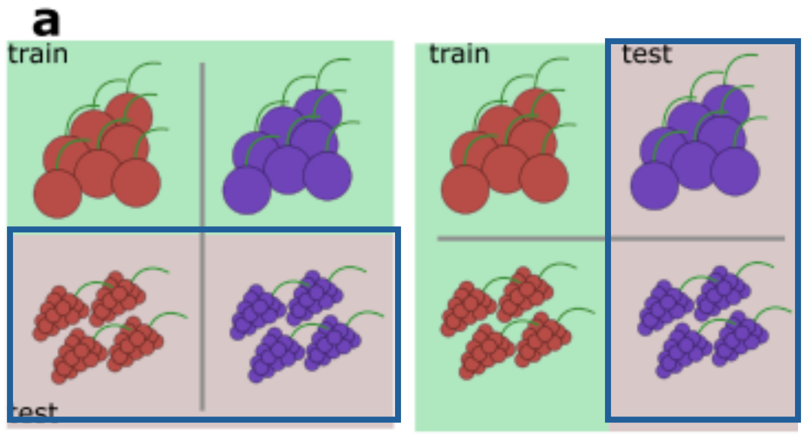

Figure 1a. Berries can be described in different ways: color and shape. For example, we can describe berries with the same shape as having a higher probability of having similar texture than berries of different shapes, regardless of their color (left figure). Berries with the same color has a higher probability of having the same taste than berries of different colors, regardless of their shape (right figure) (Johnston and Fusi, 2023).

The generalizability of concepts can be shown as geometric shape in space, also known as representational geometry (Fig. 1b).

In the left, the fact that berries of the same shape are placed on a straight line opposite to each other across the texture axis means those with the same shape have a higher probability of having the same texture. Similarly, the fact that berries of the same color are placed on a straight line opposite to each other across the taste axis means those with the same color have a higher probability of having the same taste. The straight line means the concept is generalizable across contexts. That is, a texture classification learned using berries of one shape should generalize to other shapes, and a taste classification learned using berries of one color should generalize to other colors.

In the right, the fact that berries of the same shape are placed on a curvy line opposite to each other across the texture axis means the same shape have a higher probability of having the same texture. Similarly, the fact that berries of the same color are placed on a curvy line opposite to each other across the taste axis means those with the same color have a higher probability of having the same taste. The curvy line means the concept is not generalizable across contexts. That is, a texture classification learned using berries of one shape is not generalizable to other shapes, and a taste classification learned using berries of one color should is not generalizable to other colors.

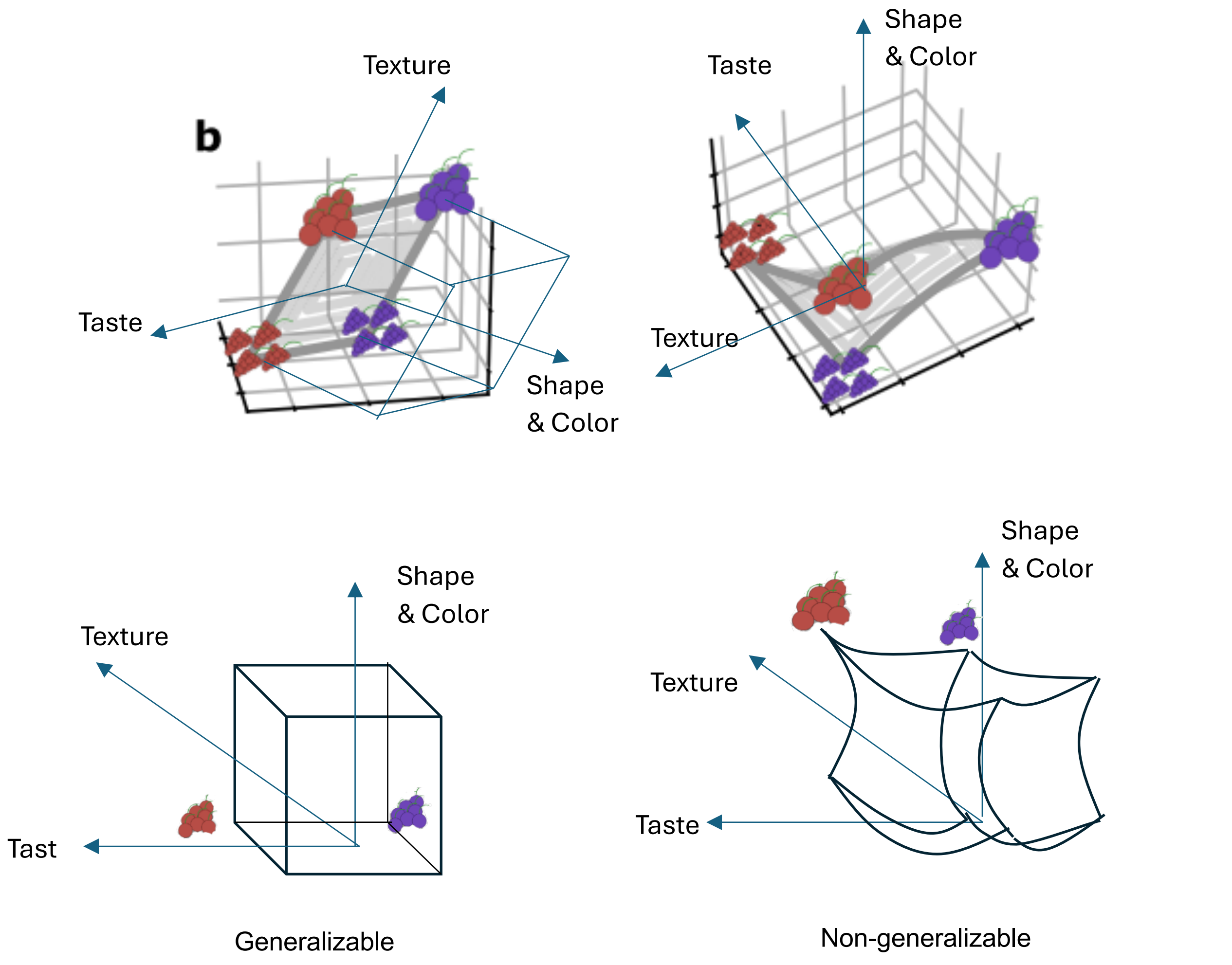

Figure 1b. Representations of knowledge in space (image modified from Johnston and Fusi, 2023). The left figures demonstrate the concept of texture is generalizable across different shapes, and the concept of taste is generalizable across different colors. The right figures demonstrate the concept of texture is non-generalizable across different shapes, and the concept of taste is non-generalizable across different colors. The bottom figures are 3D representations of the top figures. The red and blue berries images on the bottom figures help show the direction of the representation; in particular, the corner with red berries on the bottom right figure shows a curve upwards seen on the top right figure. “Texture” arrows going through the middle of the line connecting berries of the same shape demonstrate the texture concept being considered between berries of the same shape. “Taste” arrows going through the middle of the line connecting berries of the same color demonstrate the taste concept being considered between berries of the same color.

There are several steps to training a multi-tasking model from which representations of concepts have been shown to emerge. First, representations of latent variables such as representations seen above are used as input going forward (Fig. 1c), aka the standard input. Next, the multi-tasking model receives these representations as input and then trained to perform P random binary classification tasks on the latent variables. Three different squares describe different partitions of the same sample space used to train the model. Finally, after the multi-tasking model is fully trained, the level of abstraction developed in the representation layer (rep layer) is quantified using two abstraction metrics: classifier and regression. If the generalization performance is greater than chance, then a category learned in one part of latent variable space generalizes to another part of latent variable space, and thus the ability to extrapolate in the like of humans emerge.

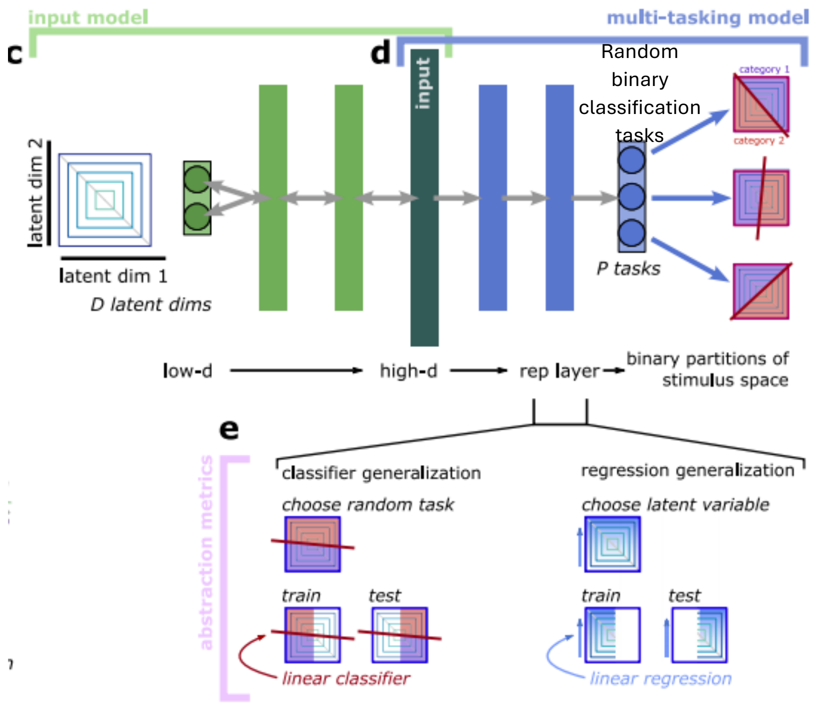

Figure 1c-e. Multi-tasking model used to develop abstraction (Johnston and Fusi, 2023).

In conclusion, data annotation process can be improved using various techniques as shown in the published literature. Selecting the best tool depends on the nature of the project and the purpose of the model. Some directions include adjusting the workflow such that there are multiple stages (Find-Fix-Verify) and breaking down larger tasks into smaller tasks (Partition-Map-Reduce). Gaussian distributions can be created for each worker keypress or task duration and then the keypress latency or task duration can be modeled to predict the correct labels. Developing the ability to understand knowledge like humans in machines may be accomplished using multi-task learning, in which the generalizability of knowledge is presented using geometric shapes and abstract representations can naturally emerge. Continuous study and development of the data labeling process leads to a more streamlined workflow, higher quality at a lower speed, and improved machine's ability to achieve human-like understanding.

References

Bernstein, M., Little, G., Miller, R., Hartmann, B., Ackerman, M., Karger, D., Crowell, D., Panovich, K. (2010). Soylent: a word processor with a crowd inside. *UIST' 10*. https://up.csail.mit.edu/other-pubs/soylent.pdf

Branson, S., Wah, C., Schroff, F., Babenko, B., Welinder, P., Perona, P., Belongie, S. (2010). Visual recognition with humans in the loop. *Computer vision - ECCV 2010.* Springer. 438-451.
Wah, C., Branson, S., Perona, P., Belongie, S. (2011). Multiclass recognition and part localization with humans in the loop. *Computer vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2524-2541.

Deng, J., Russakovsky, O., Krause, J., Bernstein, M., Berg, A., Fei-Fei, L. (2014). Scalable multi-label annotation. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 3099-3102.

Ipeirotis, P., Provost, F., Wang, J. (2010). Quality management on Amazon Mechanical Turk. *Proceedings of the 14th ACM SIGKDD workshop on human computation.* ACM, 64-67.

Jain, S., Graumann, K. (2013). Predicting sufficient annotation strength for interactive foreground segmentation. *Computer vision (ICCV), 2013 IEEE International Conference on*. IEEE, 1313-1320.

Mitchell, M. (2020). On Crashing the Barrier of Meaning in AI. *AI Magazine, 41(2)*, 86-92.
Johnston, W.J., Fusi, S. (2023). *Nature Communications, 14(1)*.

Kittur, A. Smus, B., Khamkar, S., Kraut, R. (2011). CrowdForge: Crowdsourcing complex work. *UIST' 11*. https://static.googleusercontent.com/media/research.google.com/vi//pubs/archive/39980.pdf

Krishna, R., Hata, K., Chen, S., Kravitz, J., A Shamma, D., Fei-Fei, L., & S Bernstein, M. (2016). Embracing error to enable rapid crowdsourcing. *CHI 2016*. https://cs.stanford.edu/~kenjihata/papers/Embracing_CHI_2016.pdf

Sheng, V., Provost, F., Ipeirotis, P. (2008). Get another label? improving data quality and data mining using multiple, noisy labelers. *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 614-622.

Wah, C., Horn, G., Branson, S., Majie, S., Perona, P., Belongie, S. (2014). Similarity comparison for interactive fine-grained categorization. *Computer vision and pattern recognition (CVPR), 2014 IEEE International Conference on*. IEEE, 859-866.

Whitehill, J., Wu, T., Bergsma, J., Movellan, J., Ruvolo, P. (2009). Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. *Advances in neural information processing systems*. 2035-2043.

Zhou, D., Basu, S., Mao, Y., Platt, J. (2012). Learning from the wisdom of crowds by minimax entropy. *Advances in Neural Information Processing Systems*. 2195-2203.