Integrating Machine Learning and Crowdsourcing

Crowdsourcing and Human Computation
Lecture 14
Instructor: Chris Callison-Burch
TA: Ellie Pavlick

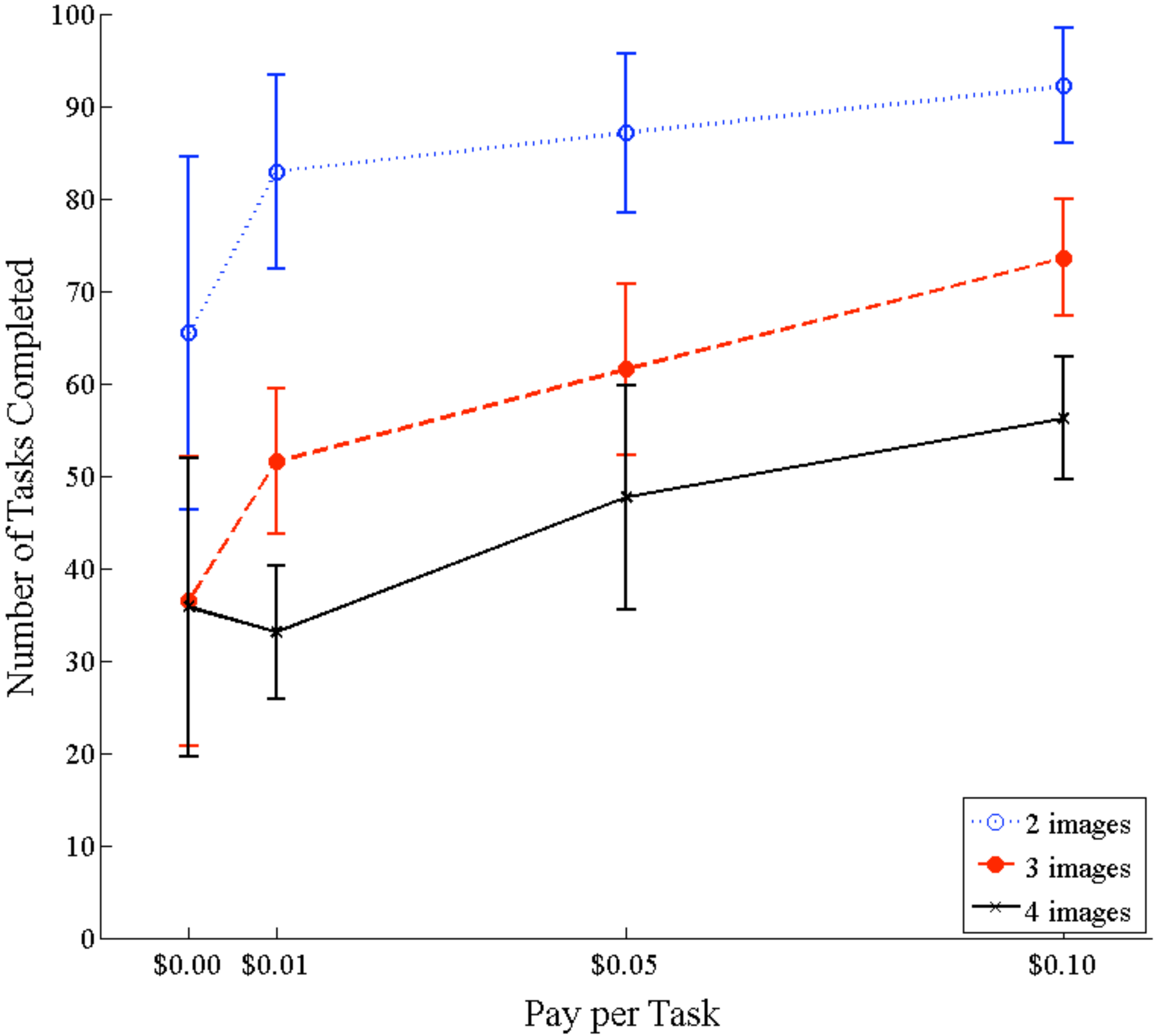
Website: crowdsourcing-class.org

Speed-Cost-Quality Tradeoffs

- Crowdsourcing usually considers speed-cost-quality tradeoffs
- If we want to get it done faster, we pay the crowd more
- If we want higher quality, we get multiple judgments and take a consensus
- If we want lower cost, we take single judgments

Factors affecting price

- Base HIT price on MTurks
- Number of assignments in the HIT group
- Amount of redundancy in judgments
- Cost of creating gold standard data
- Fraction of an item that is gold standard
- Cost of second-pass quality control HITs

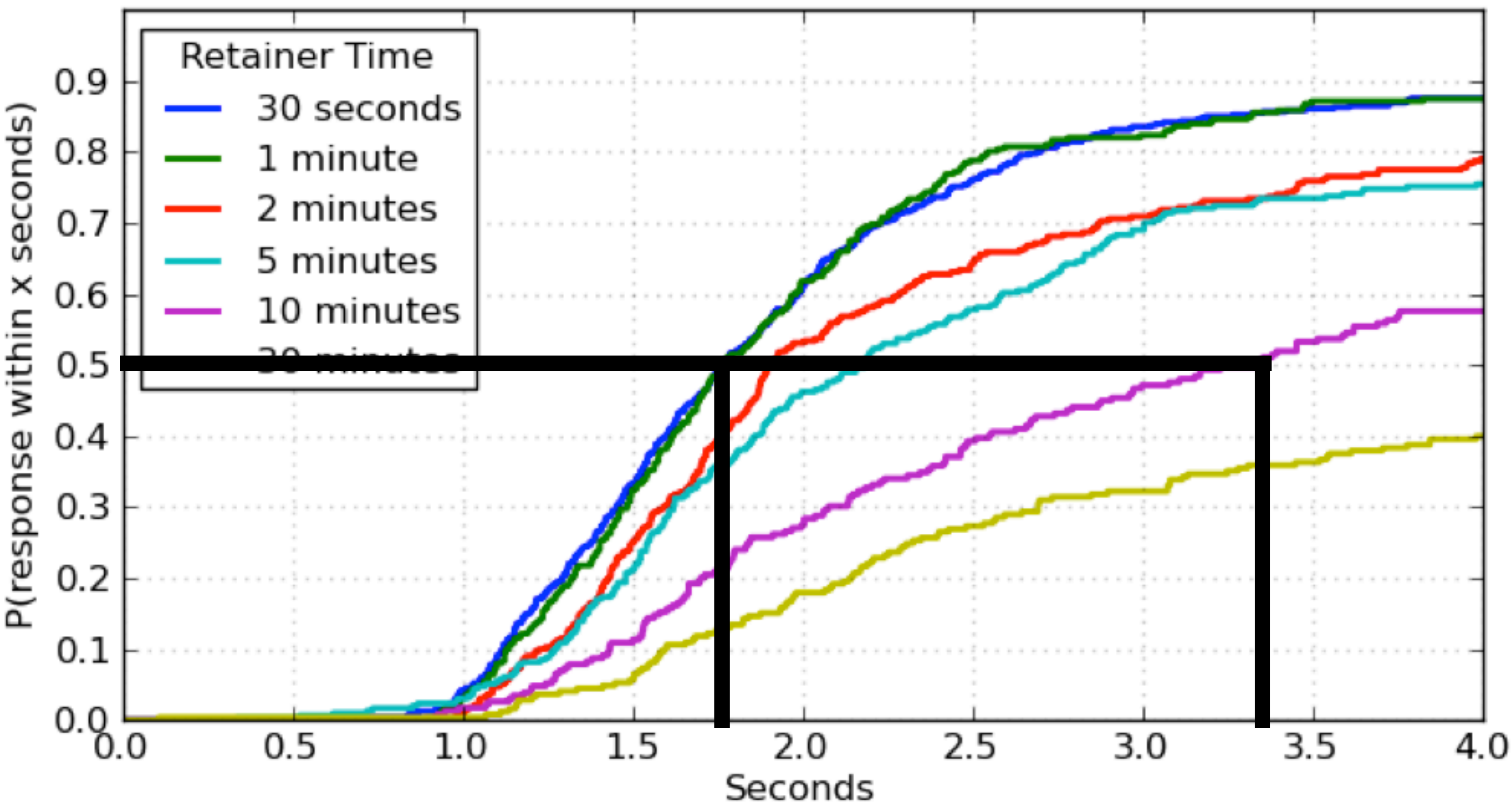


100
90
80
70
60
50
40
30
20
10
0
Number of Tasks Completed
$0.00
$0.01
$0.05
$0.10
Pay per Task
2 images
3 images
4 images

Factors affecting speed

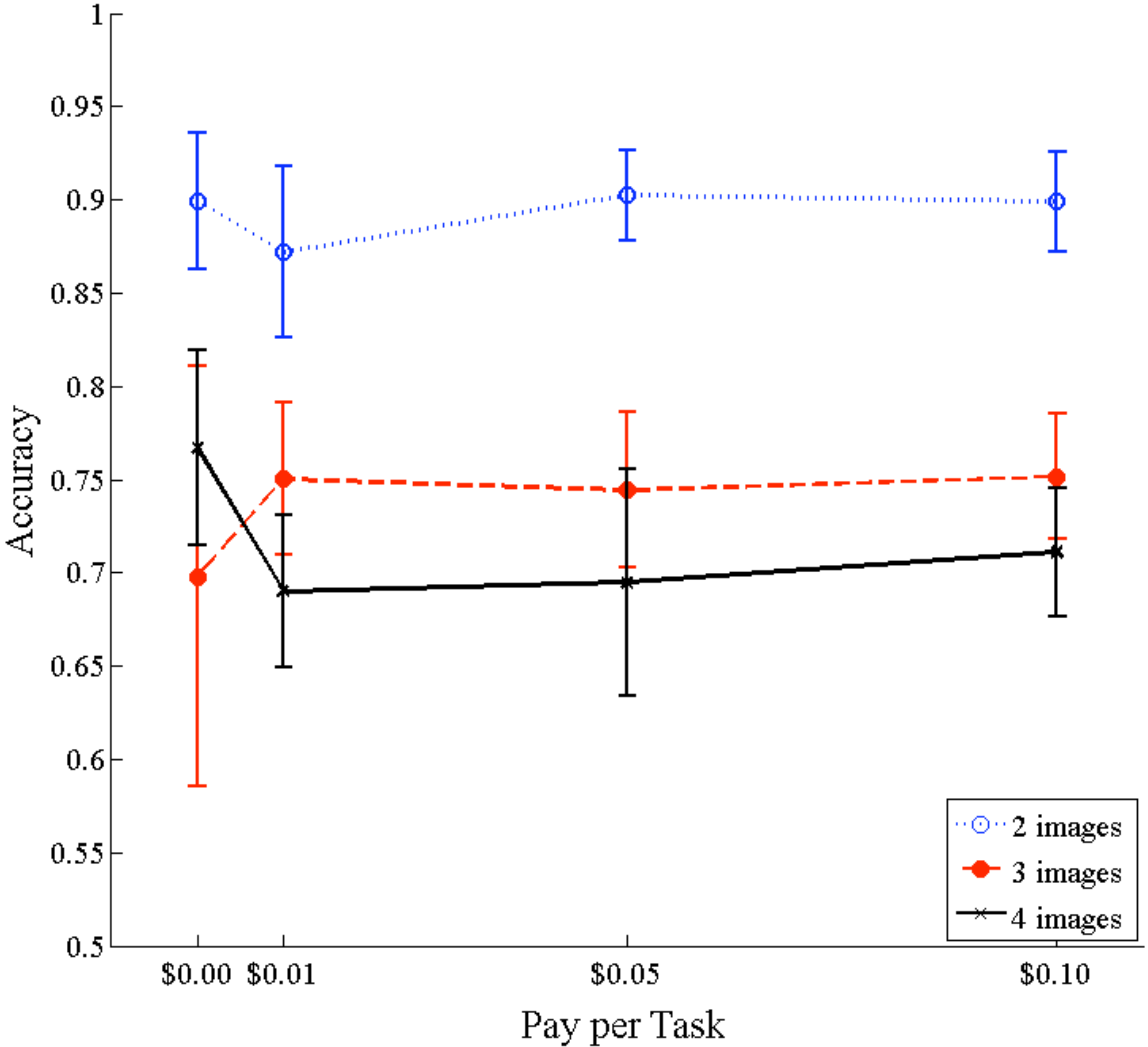
- Time for workers to discover the work
  - SEO on the MTurk listing
  - One-off HITs versus continuous HITs
  - Adrenaline's retainer model
- Time to complete each item
- Size of the pool of qualified workers / how much parallelization is possible
- Retention of workers
- Attractiveness of HIT to workers ($$$)

Retainer Reduces Response time



Factors affecting quality

- Inherent difficulty of the task, whether it requires expertise
- Clarity of the instructions
- Design of the task
- Quality control strategy
    - Qualification test / requirements
    - Redundancy + agreement
    - Embedded gold standard


Accuracy
1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0.5
$0.00
$0.01
$0.05
$0.10
Pay per Task
2 images
3 images
4 images

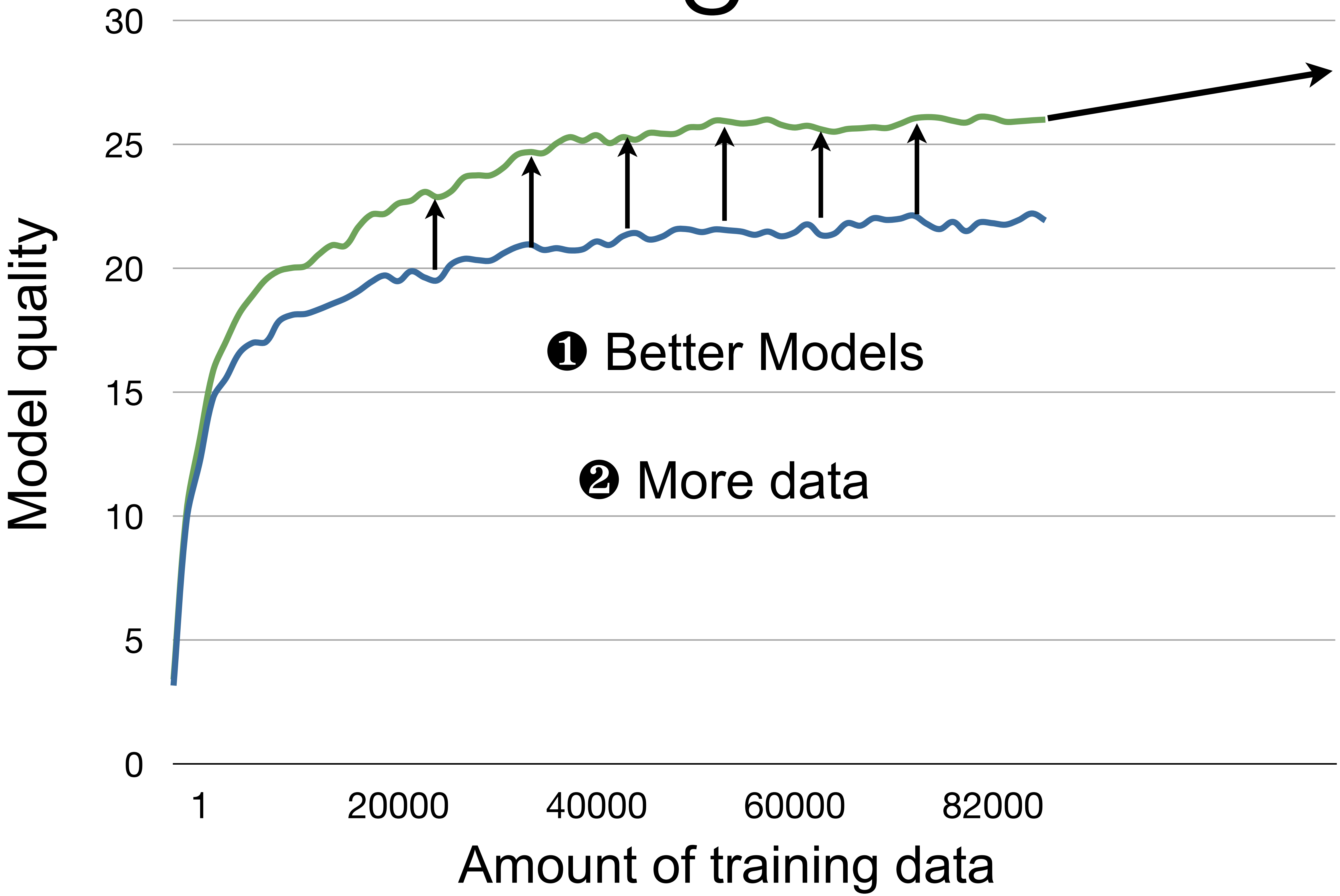
Humans v. Machines

- Many problems can be solved by either human computation or machine learning
  - Visual tasks: Recognizing faces, objects
  - NLP: Translation, sentiment, summarization
- Humans are more accurate than machines
- Machines are cheaper and faster than people

Factors affecting quality of machine learning

- How good is the model? Did we choose the right representation? Did we do good feature engineering?
- How many labeled training data was the model trained on? Are they representative of the test set? Do they have a weird bias? How noisy are they?

Learning curves



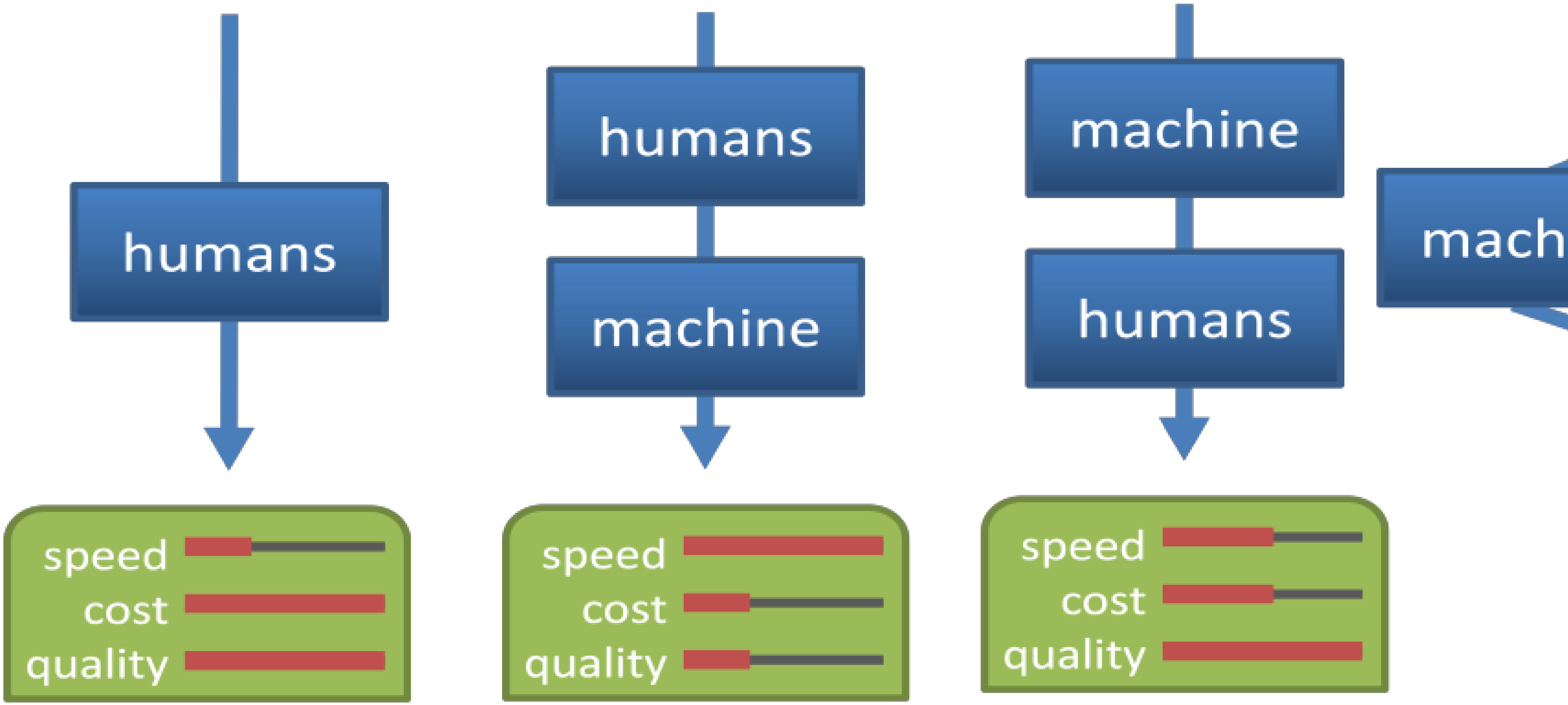
How should we combine crowdsourcing and machine learning?

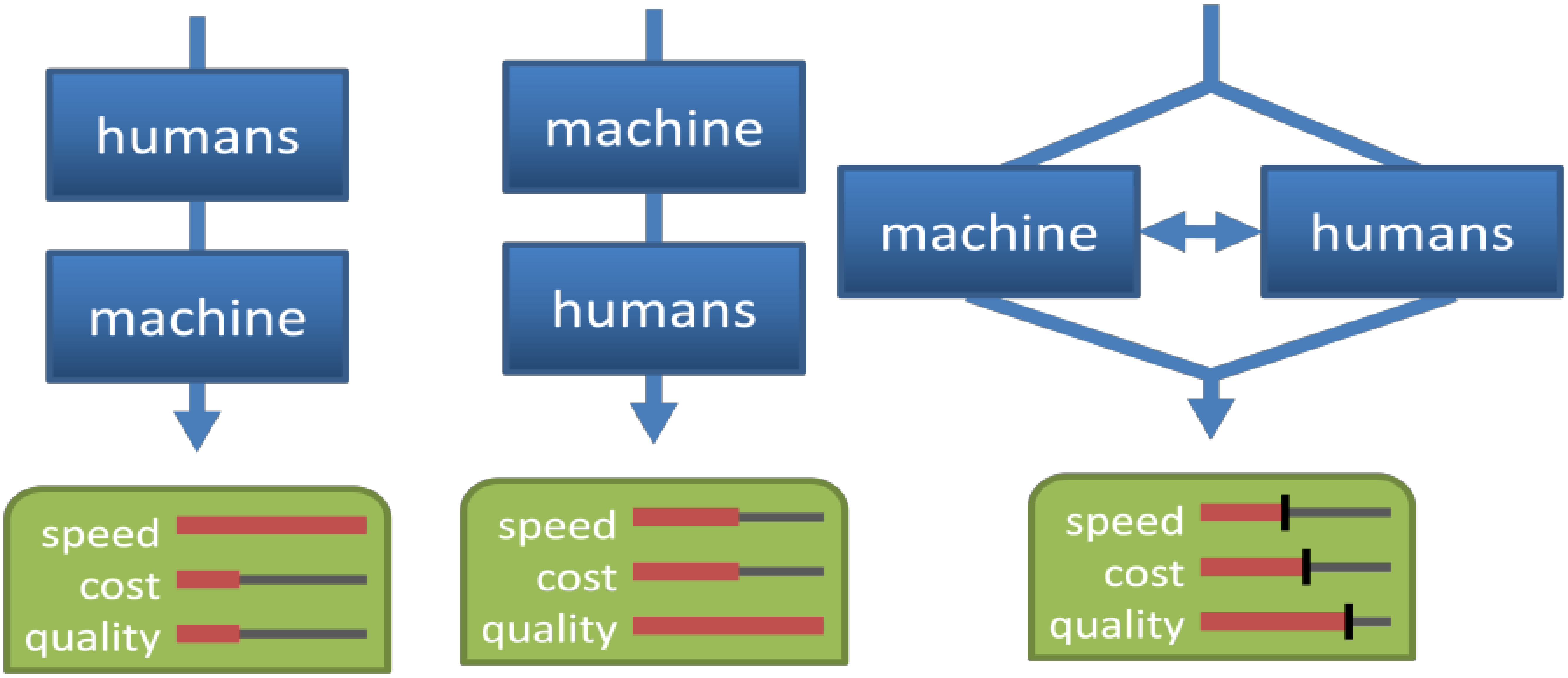
Lots of options

- Use crowdsourcing to create a labeled training set for our machine learning algorithm
- Use a mixture of crowdsourcing and machine learning to produce output
  - Have workers edit ML output
- Use machine learning to predict the quality of the crowd in order to reduce costs
- Use crowd when ML is not confident

New Speed-Cost-Quality Tradeoffs

- Adding machine learning, we have new speed-cost-quality tradeoffs
- Machine predictions are less costly and faster than human labels
- Quality is typically less than humans
- What tradeoffs do we want to make?


humans
speed
cost
quality
humans
machine
speed
cost
quality
machine
humans
speed
cost
quality
mach


humans
machine
speed
cost
quality
machine
humans
speed
cost
quality
machine
humans
speed
cost
quality

CrowdFlow

- Alex Quinn, Ben Bederson, Tom Yeh, and Jimmy Lin proposed CrowdFlow
- Framework for combining humans and machines that lets users specify their speed-cost-quality preferences
- Propose a toolkit to combine MTurk and ML in Python, allowing for different configurations

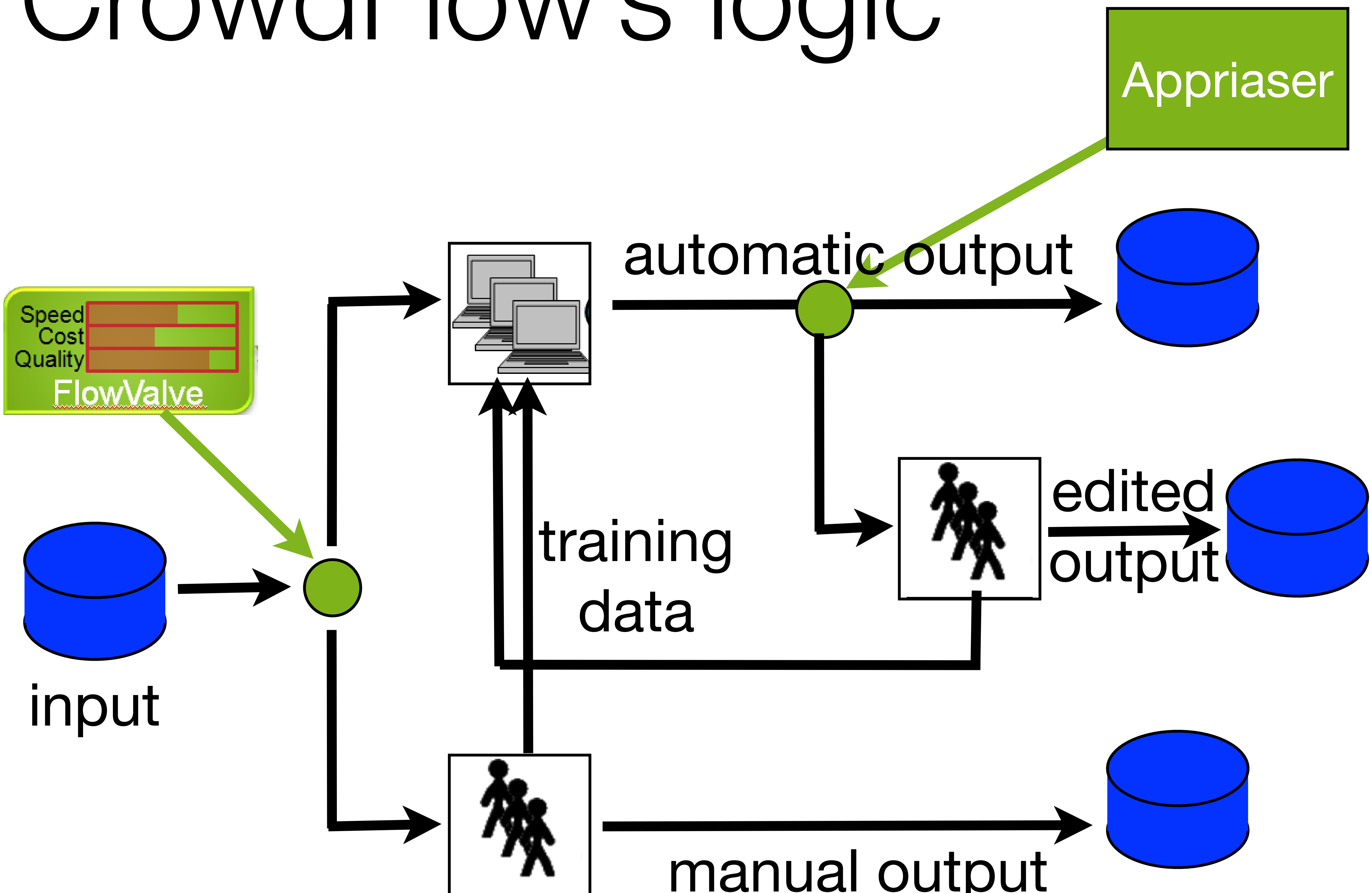
CrowdFlow

- User specifies speed by setting time limit for completing the job
- User sets maximum amount she willing to pay to Turkers
- Quality is measured relative to some objective measure
- For the system to balance the user's desired time / cost / quality preference, what do we need to estimate?

Estimates

- Accuracy of the human workers
- The machine’s accuracy
  - Compare machine’s predicted labels against the workers’ annotations
  - Use objective fn + cross-fold evaluation
- Time it will take to collect labels from workers
- Which type of human work will be most effective

CrowdFlow's logic



Feedback 1: Creating training data

- Until some minimum performance threshold is reached by the machine learner, have people annotate the data to train the learner
- At this point the workers do the complete task

Feedback 2: Fixer

- At some point, it may be quicker for a worker to correct the labels produced by the learner
- If the cognitive cost of fixing an incorrect result is low, then the fixer role is preferred
- It allows Turkers to benefit from the machine results, thus reducing their effort
- Potentially reduces costs

How should we choose which to do?

Automatically?

- **Validator** – looks at an answer from a human or machine, and predicts if it is correct or not
- **Appraiser** – takes an answer believed to be wrong and decides whether it would be easier to fix it or replace it with a new answer
- **Corrector** – takes an answer predicted to be incorrect, and either automatically fixes it, or replaces it with a new correct answer

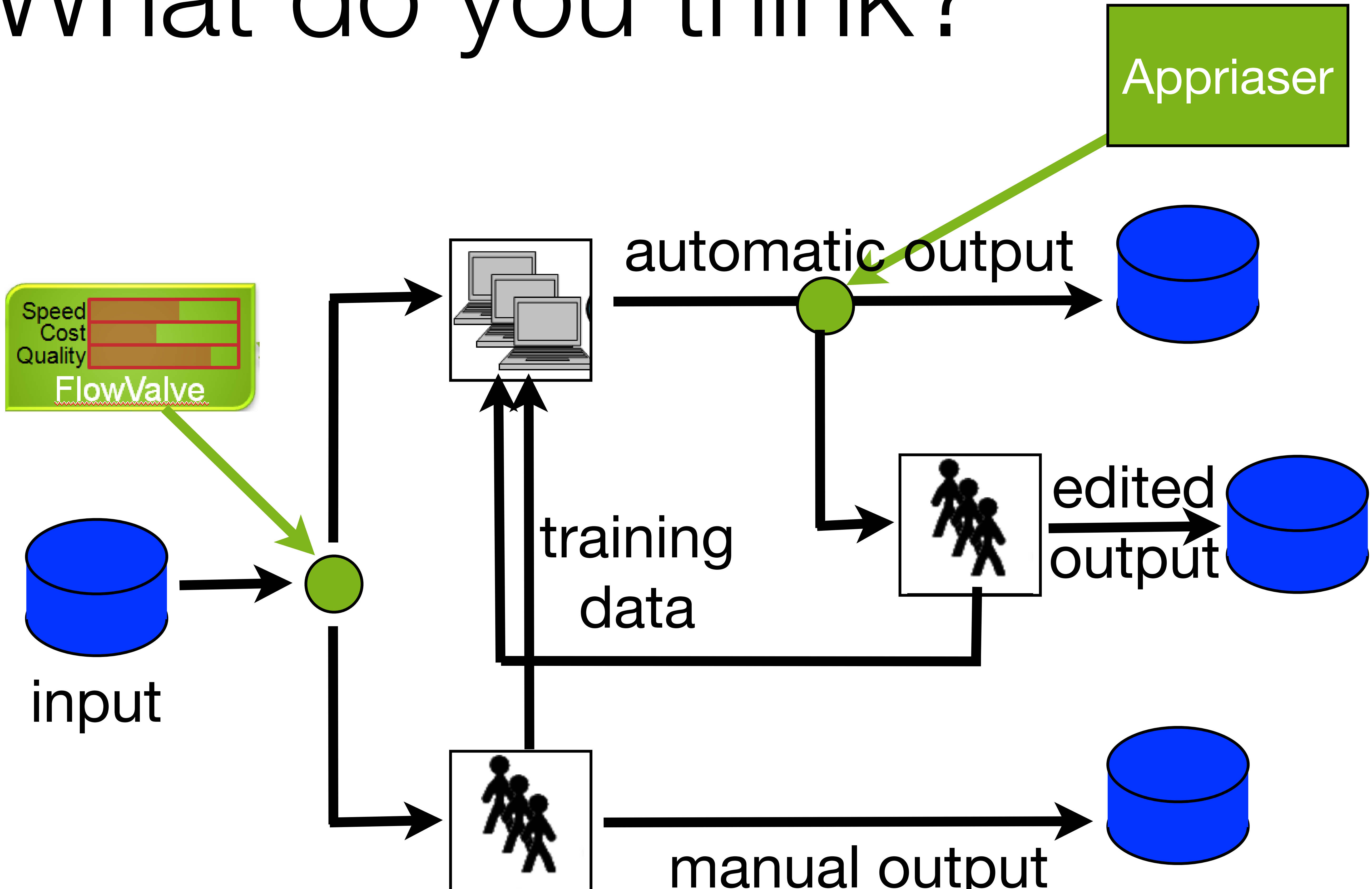
CrowdFlow implementation

- **Valve**: The controller responsible for allocating tasks to machines/humans, submitting HITs to MTurk, and ensuring accuracy/cost/time constraints are met
- **Machine**: An abstract class representing a generic classifier with *train(question,answer), evaluate(question), init()*, and *terminate()* methods
- **Task** and **HITSpecification**: Defined by the user to provide the appropriate info to MTurk

Pick a constraint: {time, cost, accuracy}

- If time is picked, it computes how many tasks it can post based on its estimate of how how long Turkers take to finish a HIT
- If cost, then it computes how many tasks it can post for the fixed budget
- If accuracy is picked then it takes the user's estimates the Turkers and the machine's accuracy, and attempts to get their average accuracy to user's level

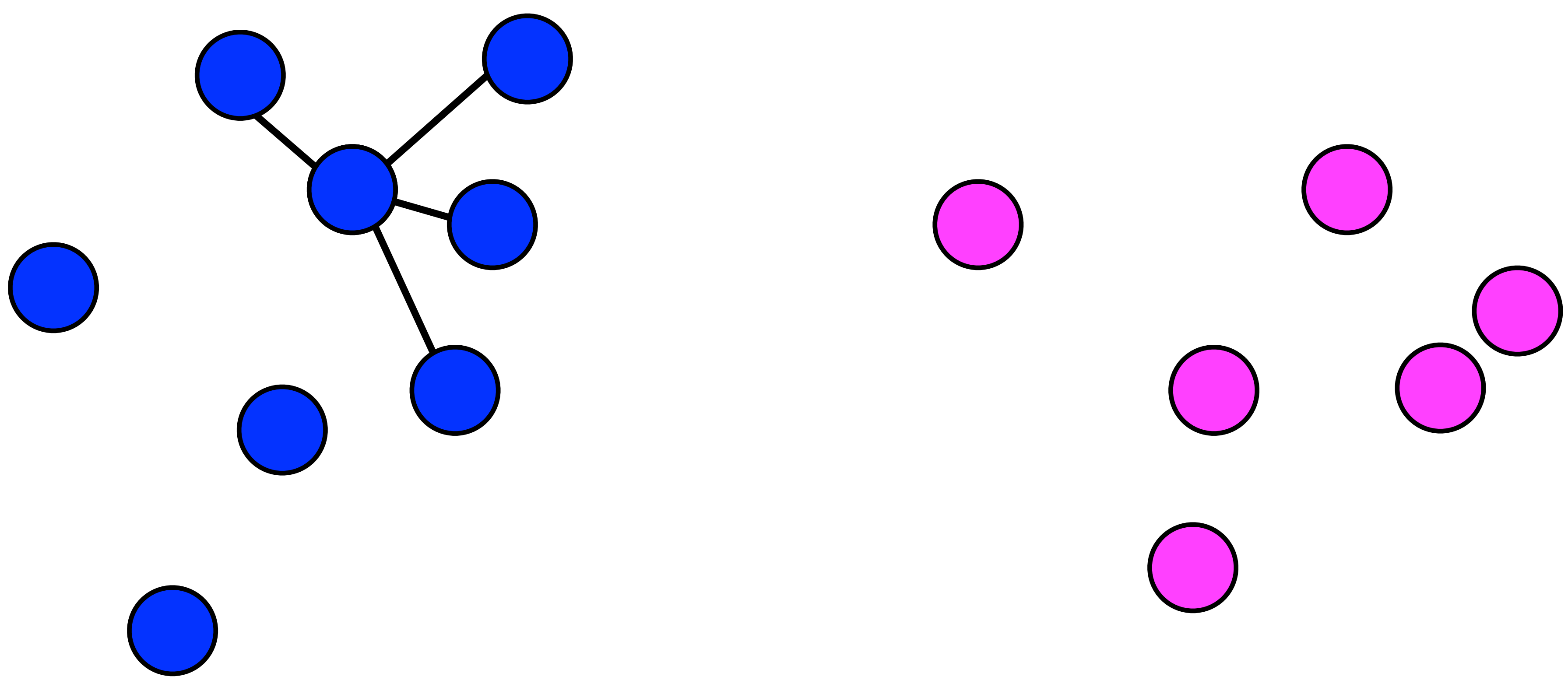
What do you think?



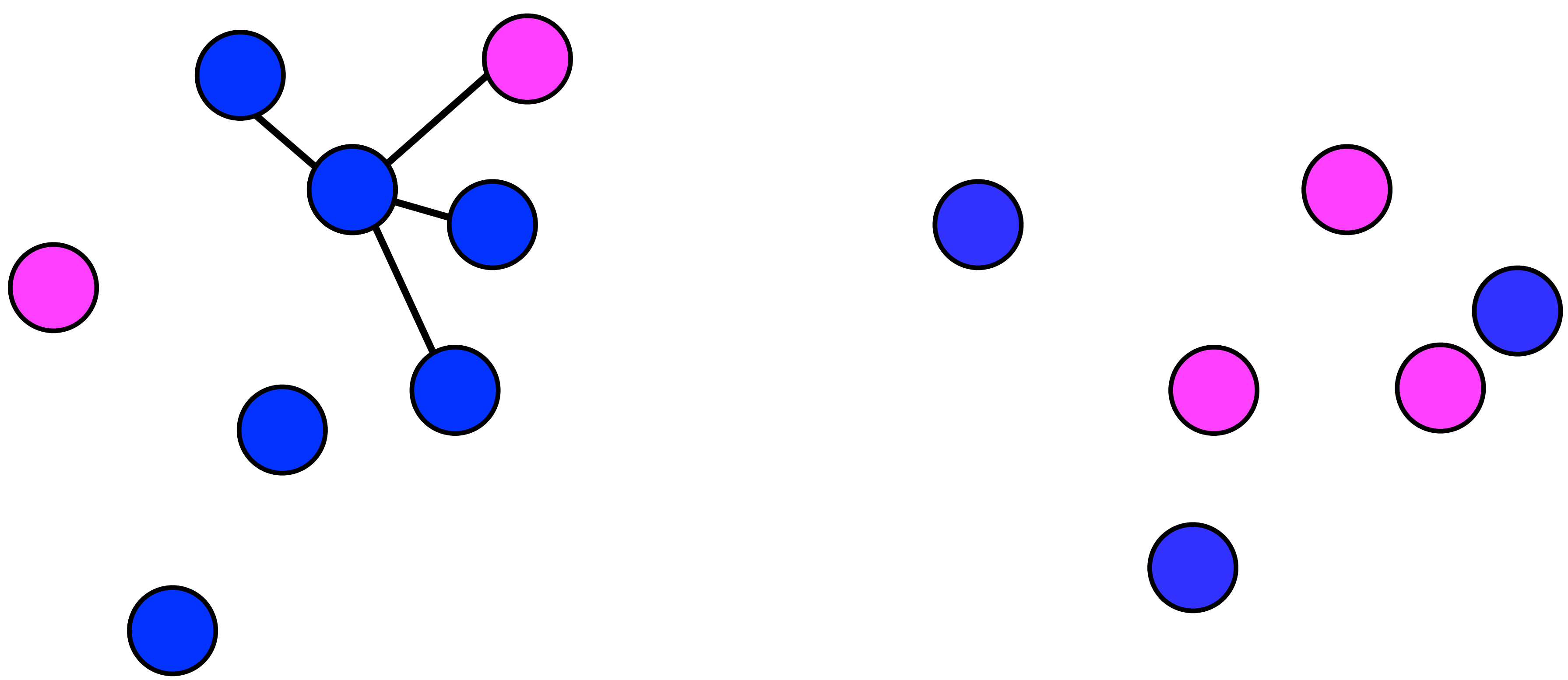
QC considerations

- Is our goal to produce high quality outputs that are consumed by people through some app?
- Or is our goal to create as much training data as possible, and then train a ML on it?
- Should that change our decision on how to do quality control?

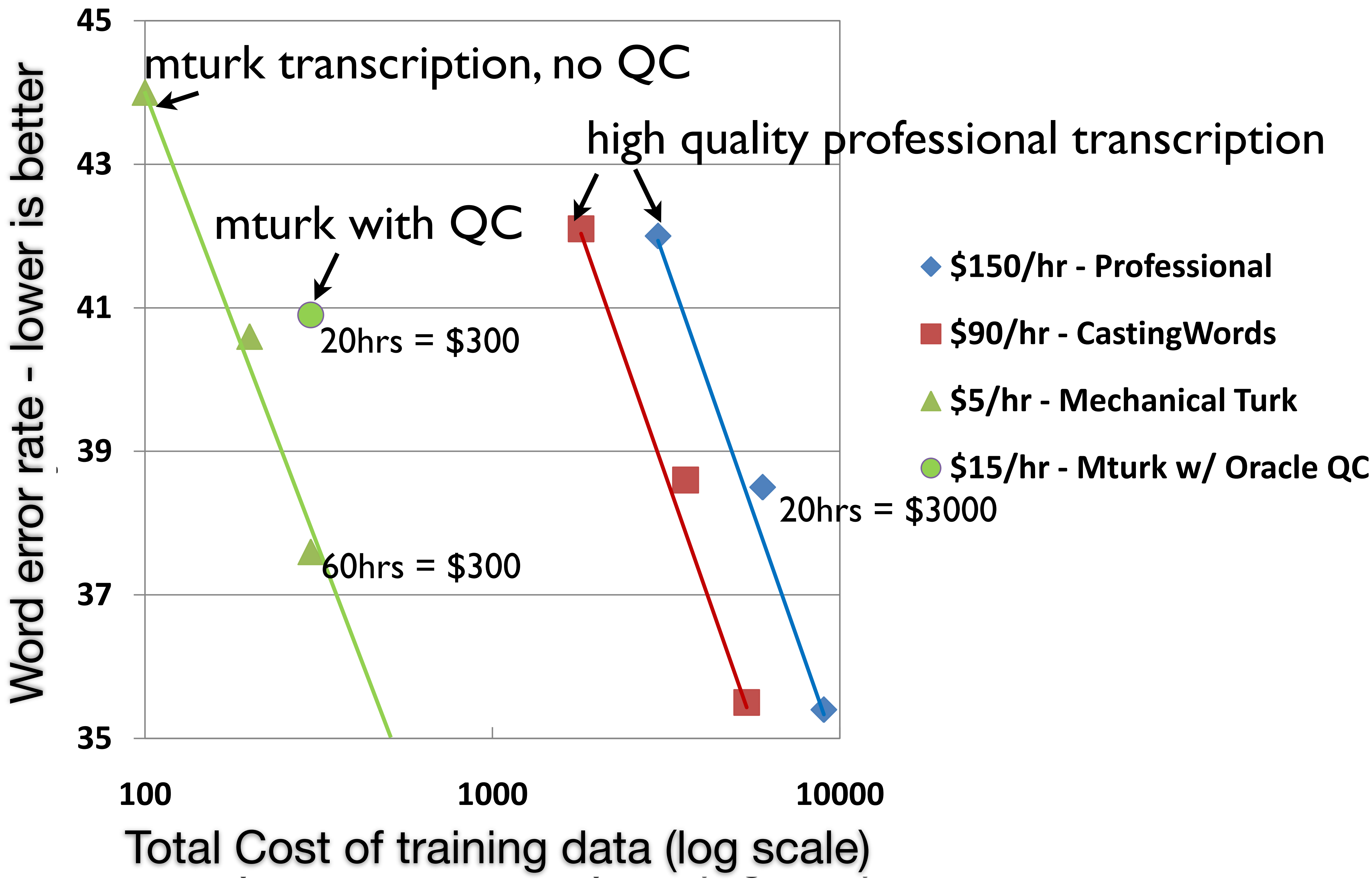
How much do high quality labels matter?



How much do high quality labels matter?



Quality v quantity when training a speech recognizer



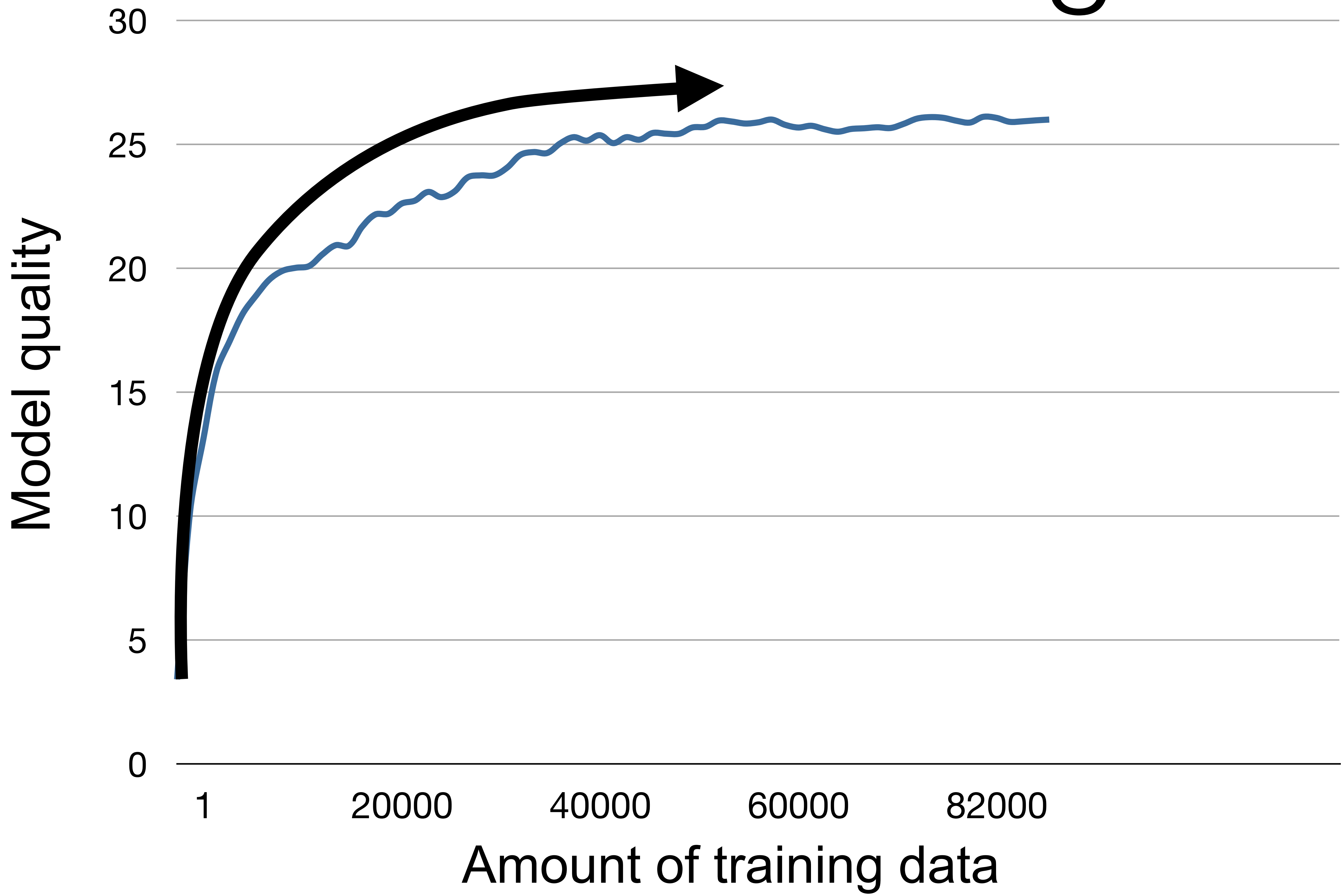
Active Learning

- In machine learning, the training data that the learner encounters is typically a fixed set that is given at the outset
- If we integrate crowdsourcing and machine learning, then the training set grows over time
- If our goal is higher quality accuracy, how could the machine learning model help us to achieve that?

Active Learning

- Active learning is a subfield of machine learning where the ML algorithm picks what data to have labeled
- Goal is to choose unlabeled training items that will be maximally useful to the model when labeled
- “Selective sampling” instead of random sampling

Active learning

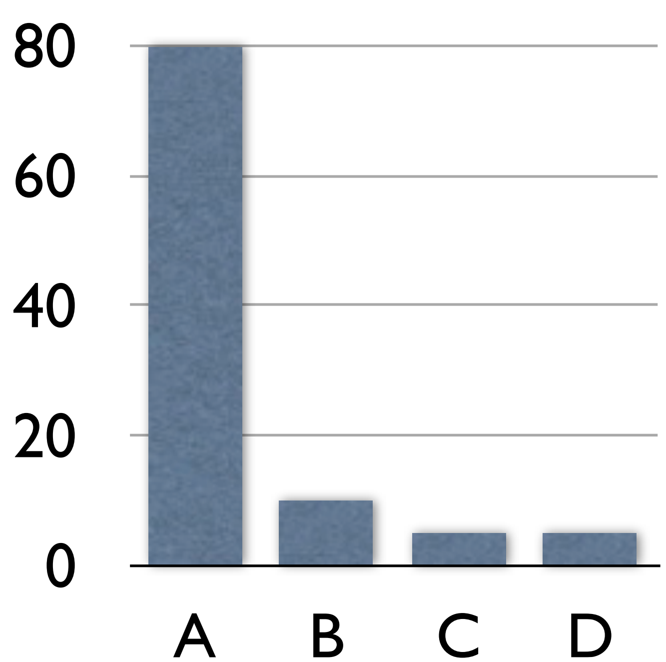


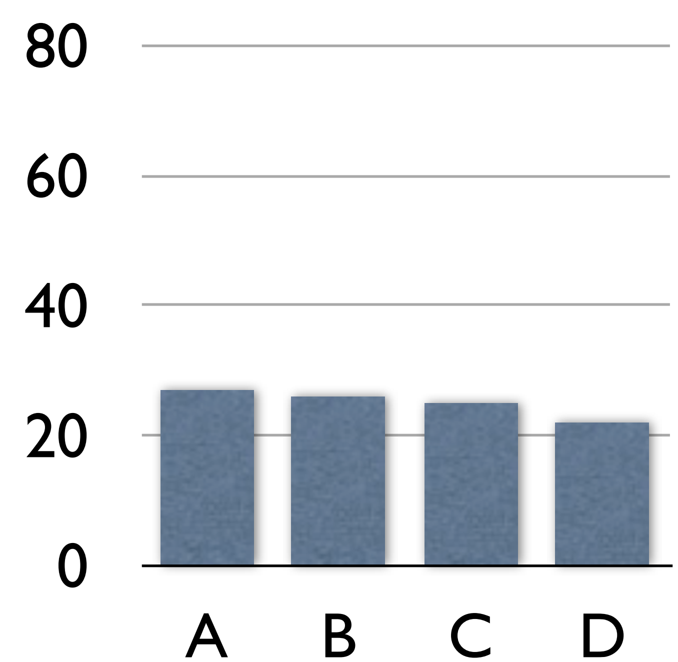
Selective sampling strategies

- **Committee based selection** – create an *ensemble* of different classifiers, and skip items where they agree on the label. Pick unlabeled items with maximal disagreement

Selective sampling strategies

- **Confidence-based selection** – Look how a single learner assigns probability mass across the different possible labels. Choose items with maximum entropy.





Active Learning

- Almost all current active learning studies are *simulated*
- They take an existing set of labeled training data, hide the labels and then choose what order to reveal the labels to the classifier
- Report cost savings as a % of items needed to reach a fixed performance value
- What deficiencies does this have?

Active Learning Pitfalls

- In a non-simulated setting the pool of unlabeled data is much much larger – this changes the potential set quite a bit
- The assumption that each item costs the same to label may or may not be true – Items that are hard for classifiers to label might also have higher cognitive load for humans

Active Learning Pitfalls

- Typically active learning is viewed as an iterative process
    - The learner picks the next item to have labeled,
    - Retrains on previous training set plus the new item
    - Runs over the entire unlabeled set, making predictions and looking for the next item to request a label for

Active Learning Pitfalls

- The length of time that it takes to re-train a classifier can be non-trivial
- *Online learners* can be better in that regard, since the update their parameters after seeing each new item
- Length of time for classifying large unlabeled set could be very long
- We might not want our annotators sitting around doing nothing, so choose batches of items instead of on individual items

Active Learning Pitfalls

- The items selected by one type of machine learning classifier might not be the same as the items selected by another classifier
- Could result in worse performance than random sampling when training a new classifier

Take aways

- There are a lot of different ways that we can integrate machine learning and crowdsourcing
- CrowdFlow is one example of how to think about routing tasks, and what constraints we might consider
- Non-simulated active learning is an exciting potential research topic to explore, if our ultimate goal is better machine learners