crowdsourcing workflow control

- Nate Tucker and Perry Green

barriers to effective crowdsourcing

- Last time:
  - Ensure proper incentives:
    - Positive and negative
    - Social and economic
- This time:
  - How can we help people answer the question *correctly?*
  - How can we *aggregate* lots of responses into a single answer?
- Sound familiar? It's all about information aggregation.

dynamically switching between synergistic workflows for crowdsourcing

- Motivation: why would two workflows be better than one (even if, on average, one is known to yield more accurate results)?
  - Workers have different skillsets
  - Different errors in different workflows
- Examples?

how should we change the model?

- Before: Learn the best workflow. Use it.

$$a(d, \gamma_w) = \tfrac{1}{2}[1 + (1 - d)^{\gamma_w}]$$

- After: Dynamically choose workflows to maximize certainty of your answer.

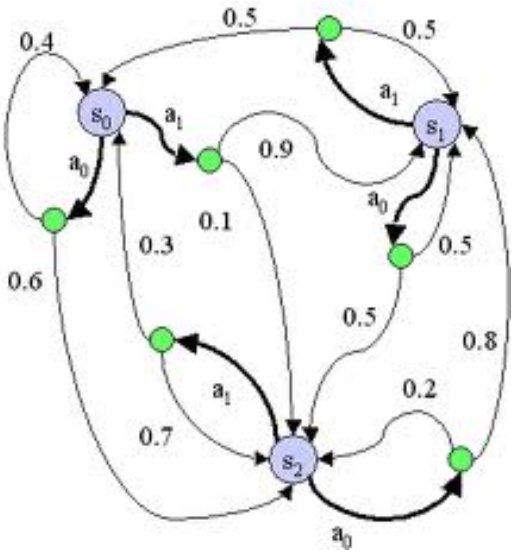
$$a(d^k, \gamma_w^k) = \frac{1}{2}\left(1 + (1 - d^k)^{\gamma_w^k}\right)$$

decision-theoretic agent

- So far we have a model for accuracy, but not how to actually *decide* what task to use/when to return an answer.
- We need to:
  - Given initial difficulty/error parameters, decide whether to make a new task, or return an answer
  - If we made a new task and got new information, update our parameters and repeat

decision given parameters - POMDP

- Partially observable Markov decision process:
  - Markov decision process:
    - Given states, actions, transition probabilities, and rewards, find the best policy (action to take in each state)
  - Partially observable:
    - You don't know what state you are in
- For our purposes, we have some black box to approximately solve POMDPs



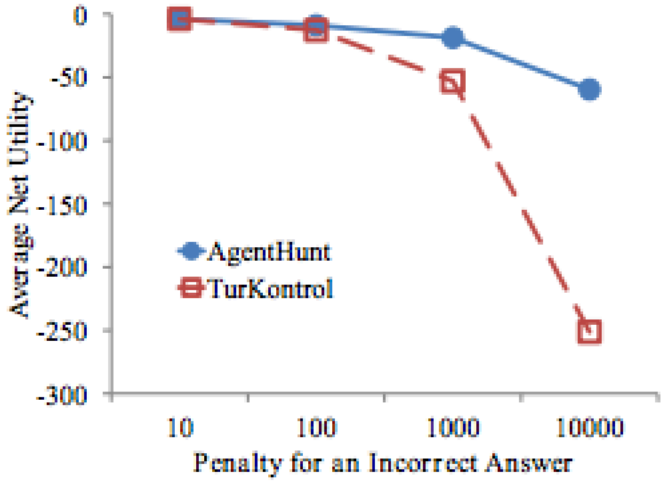
POMDP – AgentHunt details

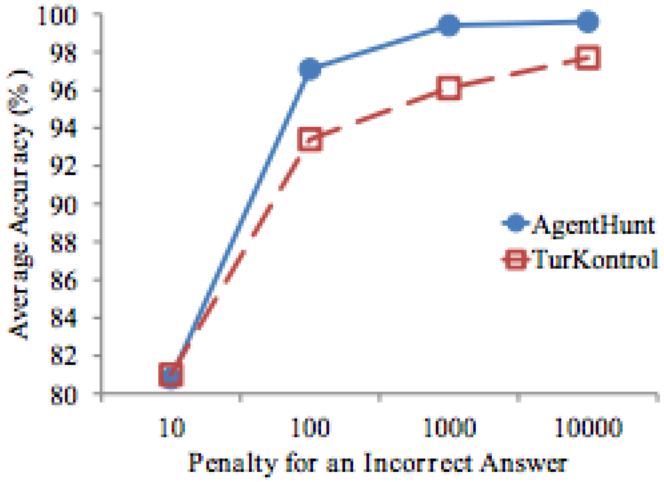
- Each state is $(d^1, d^2, \ldots d^k, v)$, where v is the correct answer
- Each action is either to make a new job for one of the workflows, or submit one of the two answers
- Reward function assigns some fixed cost for making a new task, and another (large) fixed cost for submitting the wrong answer
- Each transition involves updating all parameters

learning parameters

- Offline:
  - 1. Collect training data
  - 2. EM-algorithm, alternatively treating parameters and true value as fixed
  - 3. Use this to find average error parameter, initial estimate for d's
- Online:
  - 1. Start with uniform priors for d
  - 2. After some number of responses, update all parameters to define a new POMDP
  - 3. Exploration v Exploitation: randomly takes a suboptimal action with some small probability to explore

results


0
-50
-100
-150
-200
-250
-300
Average Net Utility
AgentHunt
TurKontrol
10
100
1000
10000
Penalty for an Incorrect Answer


100
98
96
94
92
90
88
86
84
82
80
Average Accuracy (%)
AgentHunt
TurKontrol
10
100
1000
10000
Penalty for an Incorrect Answer

crowdsourcing control: moving beyond multiple choice (abridged)

- Problem: hard to apply existing probabilistic models to open questions
- What do they want in a solution:
  - Given a correct answer and a difficulty, whether workers get it right is uncorrelated (i.e. no collusion)
  - If two workers get it wrong, their answers are correlated (i.e. there are common mistakes)
  - There is a single correct answer

chinese table

- With some probability, determined by difficulty and error parameter for worker, correct answer is given.
- If incorrect, they pick a table:
  - New table with probability $\Theta / (\Theta + N)$, where N is the number of people in restaurant, and Θ is the bandwagon parameter
  - Old table t with probability $f(t)/(\Theta+N)$, where f(t) is the number of people at t

summary

- LazySusan:
  - Decides whether to produce a new task, depending on the costs and predicted benefits.
  - Predicts answer based on chinese table model
- Results: