Turkit: Human Computation Algorithms on Mechanical Turk

Greg Little, Lydia B. Chilton, Max Goldman, Robert C. Miller

Human computation Algorithms

Tasks that build on each other.

Human computation Algorithms

Tasks that build on each other.

- Please describe the image **factually**.
- You may use the provided text as a starting point, or delete it and start over.
- Use no more than 500 characters.

Lightening strike in a blue sky near a tree and a building.

character count: 59/500

Submit

Human computation Algorithms

Tasks that build on each other.

Iteration 1: Lightening strike in a blue sky near a tree and a building.
Iteration 2: The image depicts a strike of fork lightening, striking a blue sky over a silhoutted building and trees. (4/5 votes)
Iteration 3: The image depicts a strike of fork lightning, against a blue sky with a few white clouds over a silhouetted building and trees. (5/5 votes)
Iteration 4: ~~The image depicts a strike of fork lightning, against a blue sky- wonderful capture of the nature.~~ (1/5 votes)
Iteration 5: This image shows a large white strike of lightning coming down from a blue sky with the tops of the trees and rooftop peaking from the bottom. (3/5 votes)
Iteration 6: This image shows a large white strike of lightning coming down from a blue sky with the silhouettes of tops of the trees and rooftop peeking from the bottom. The sky is a dark blue and the lightening is a contrasting bright white. The lightening has many arms of electricity coming off of it. (4/5 votes)

- Please describe the image **factually**.
- You may use the provided text as a starting point, or delete it and start over.
- Use no more than 500 characters.

Lightening strike in a blue sky near a tree and a building.

character count: 59/500

Submit

Crash-and-rerun programming model

- Model for when local computation is cheap and remote work is costly
- Managing states over a long running program is challenging
  - Examples: Computer restarts? Errors?
- Solution: store states in the database (in case)
- If an error happens, just crash the program and re-run by following the history in DB
  - Throw a “crash” exception; the script is automatically re-run.
- New keyword “once”:
  - Remove non-determinism
  - Don’t need to re-execute an expensive operation (when re-run)
- But why should we re-run???

Turkit Script

Built on top of Javascript

Supports the crash-and-rerun programming model

Allows for simple parallelism via the **fork** and **join** operators.

Provides wrapper functions for AMT REST API

Building blocks Voting and Sorting

Written in Java, using Rhino to interpret JavaScript code, and E4X to handle XML results from MTurk.

Turkit Script

```
quicksort(A)
    if A.length > 0
        pivot ← A.remove(A.randomIndex())
        left ← new array
        right ← new array
        for x in A
            if compare(x, pivot)
                left.add(x)
            else
                right.add(x)
        quicksort(left)
        quicksort(right)
        A.set(left + pivot + right)

compare(a, b)
    hitId ← createHIT(...a...b...)
    result ← getHITResult(hitId)
    return (result says a < b)
```

Turkit Script

```
quicksort(A)
    if A.length > 0
        pivot ← A.remove(A.randomIndex())
        left ← new array
        right ← new array
        for x in A
            if compare(x, pivot)
                left.add(x)
            else
                right.add(x)
        quicksort(left)
        quicksort(right)
        A.set(left + pivot + right)

compare(a, b)
    hitId ← createHIT(...a...b...)
    result ← getHITResult(hitId)
    return (result says a < b)
```

```
quicksort(A)
    if A.length > 0
        pivot ← A.remove(once A.randomIndex())
        left ← new array
        right ← new array
        for x in A
            if compare(x, pivot)
                left.add(x)
            else
                right.add(x)
        quicksort(left)
        quicksort(right)
        A.set(left + pivot + right)

compare(a, b)
    hitId ← once createHIT(...a...b...)
    result ← once getHITResult(hitId)
    return (result says a < b)
```

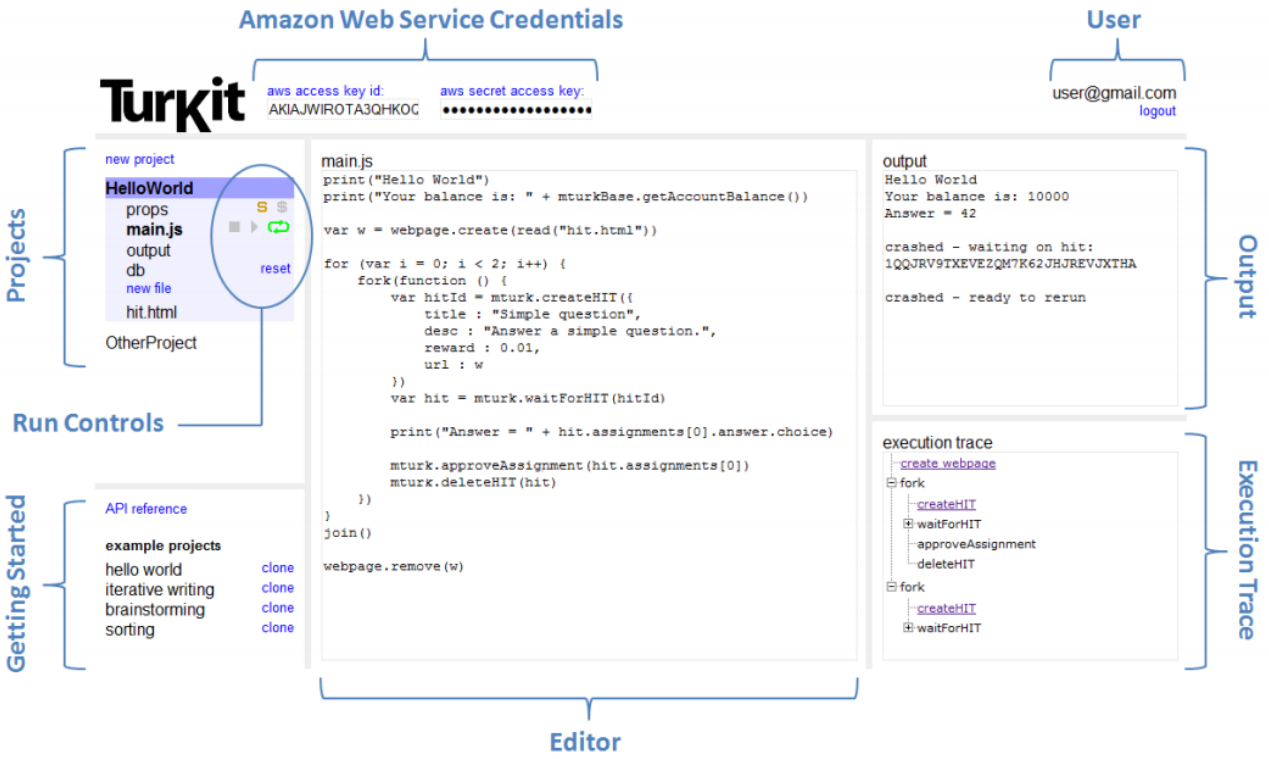
Turkit Script

```
quicksort(A)
    if A.length > 0
        pivot ← A.remove(once A.randomIndex())
        left ← new array
        right ← new array
        for x in A
            if compare(x, pivot)
                left.add(x)
            else
                right.add(x)
        quicksort(left)
        quicksort(right)
        A.set(left + pivot + right)

compare(a, b)
    hitId ← once createHIT(...a...b...)
    result ← once getHITResult(hitId)
    return (result says a < b)
```

```
quicksort(a) {
    if (a.length == 0) return
    var pivot = a.remove(once(function () {
        return Math.floor(a.length * Math.random())
    }))
    var left = []
    var right = []
    for (var i = 0; i < a.length; i++) {
        fork(function () {
            if (vote("Which is best?",
                    [a[i], pivot]) == a[i]) {
                right.push(a[i])
            } else {
                left.push(a[i])
            }
        })
    }
    join()
    fork(function () {
        quicksort(left)
    })
    fork(function () {
        quicksort(right)
    })
    join()
    a.set(left.concat([pivot]).concat(right))
}
```

Online Web Interface



Example Applications

Iterative Writing

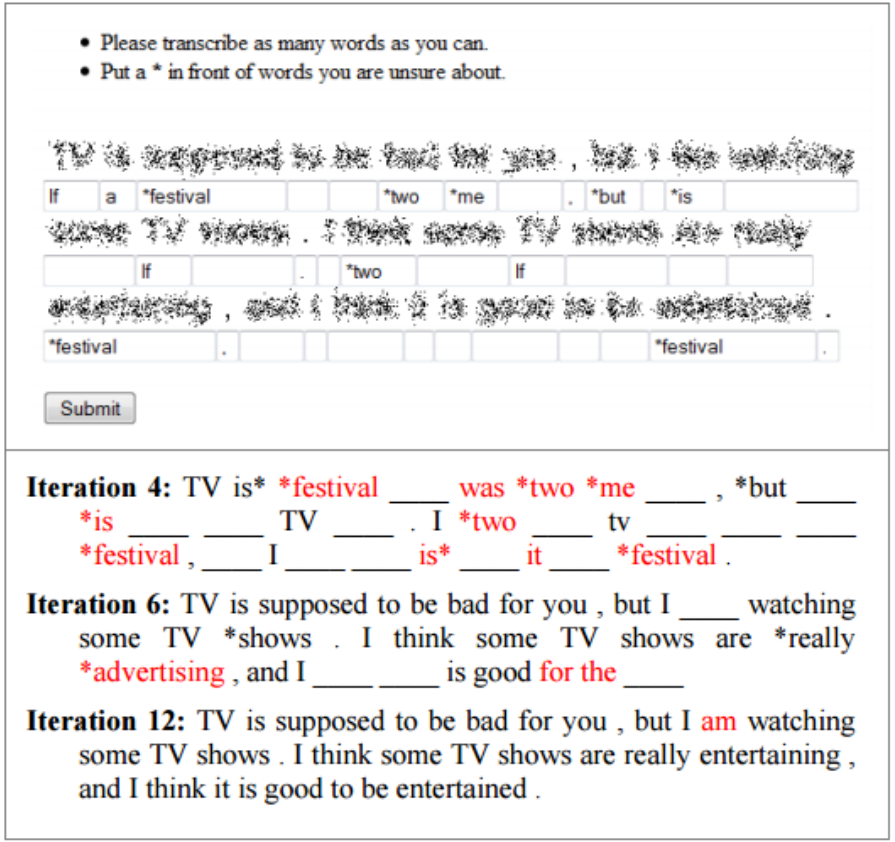
```
// generate a description of X
// and iterate it N times
var text = ""
for (var i = 0; i < N; i++) {
    // generate new text
    var newText = mturk.prompt(
        "Please write/improve this paragraph
         describing " + X + ": " + text)

    // decide whether to keep it
    if (vote("Which describes " + X + " better?",
        [text, newText]) == newText) {
        text = newText
    }
}
```

Example Applications

Iterative Writing

Blurry Text Recognition



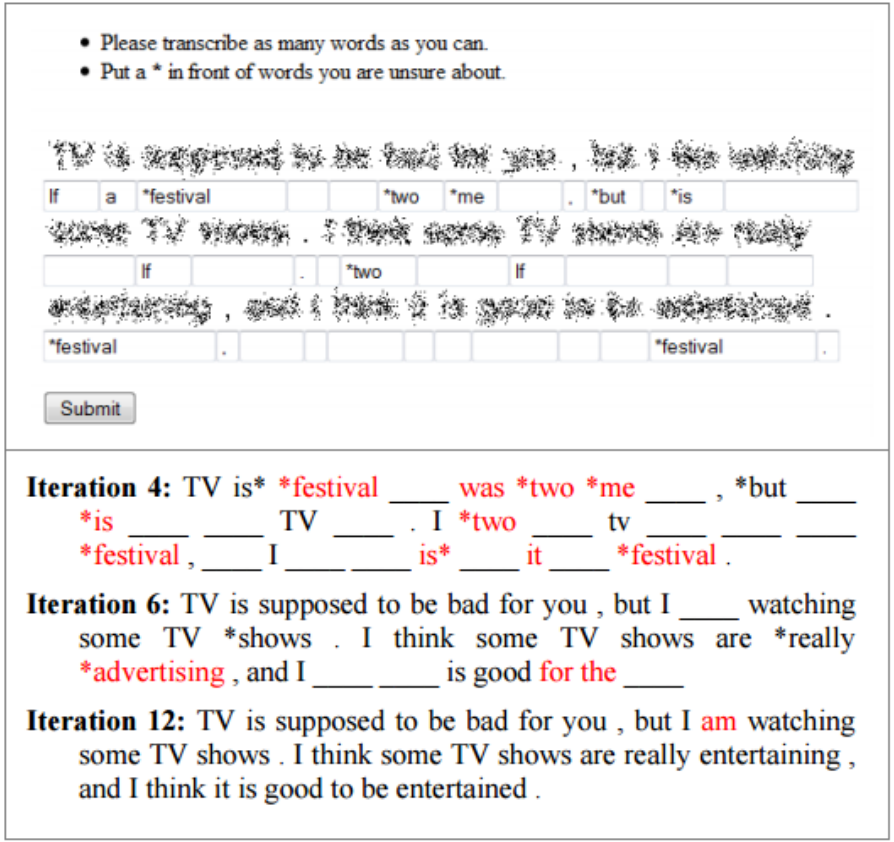
Example Applications

Iterative Writing

Blurry Text Recognition

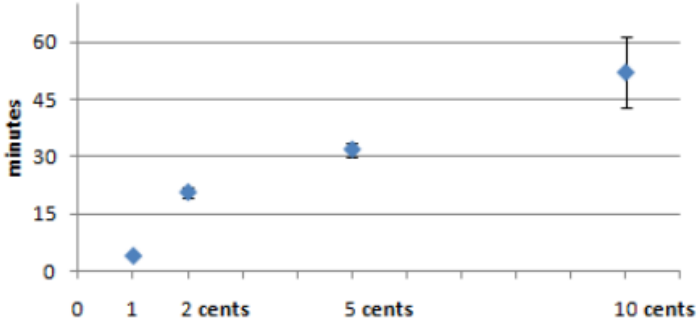
Decision Theory Experimentation

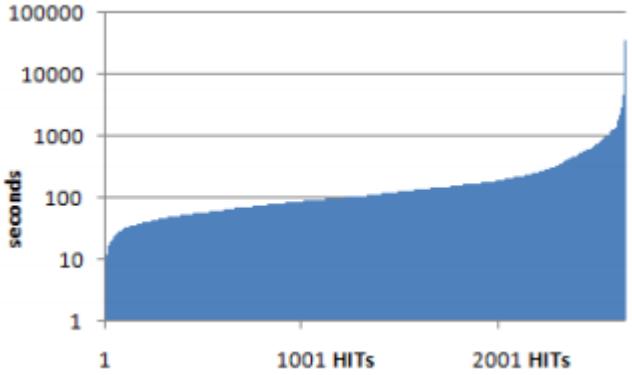
Psychophysics Experimentation



Performance Evaluations

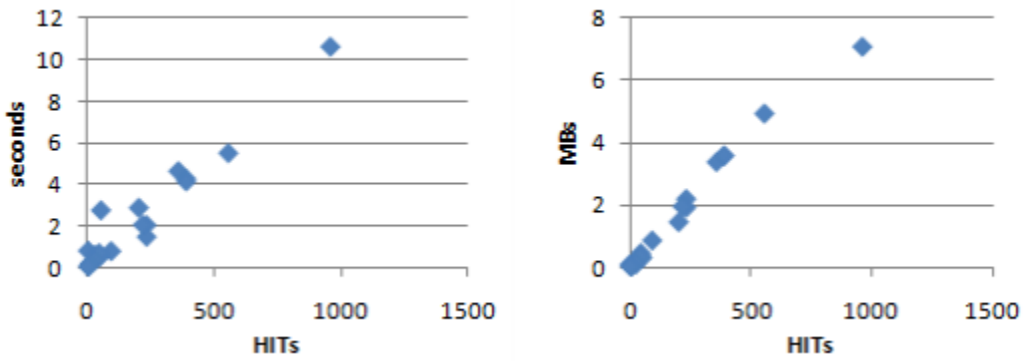
- Dataset : Corpus of 20 TurKit experiments over a year
  - taks: iterative writing, blurry text recognition, website clustering, brainstorming, and photo sorting.
  - $364.85 for 29,731 assignments across 3,829 HITs.
- Round-trip-time for first assignment completion
  - avg. 4 min for $0.01 tasks,





Performance Evaluations

- Dataset : Corpus of 20 TurKit experiments over a year
  - taks: iterative writing, blurry text recognition, website clustering, brainstorming, and photo sorting.
  - $364.85 for 29,731 assignments across 3,829 HITs.
- Turkit execution time & memory consumption
  - At most takes 11 seconds to run full trace



Discussion

Usability vs Scalability

Parallel Programming Model Limitations

Experimental Replication

Conclusion

- TurKit is a toolkit for exploring human computation algorithms on Mechanical Turk.
- Uses crash-and-rerun programming model for writing fault-tolerant scripts
- TurKit Script: An API for writing algorithmic MTurk tasks using crash-and-rerun programming.
- TurKit Online: A public web GUI for running and managing TurKit scripts.
- TurKit performance evaluated on a corpus of 20 scripts posting almost 30,000 tasks, shows its reasonably fast for most HITs