

CROWD-SOURCING

Simin Chen

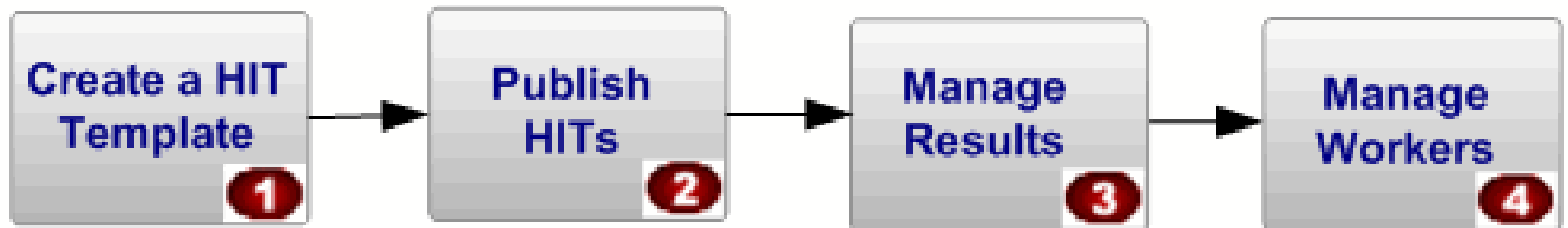
Amazon Mechanical Turk

- Advantages
 - ▣ On demand workforce
 - ▣ Scalable workforce
 - ▣ Qualified workforce
 - ▣ Pay only if satisfied

Terminology

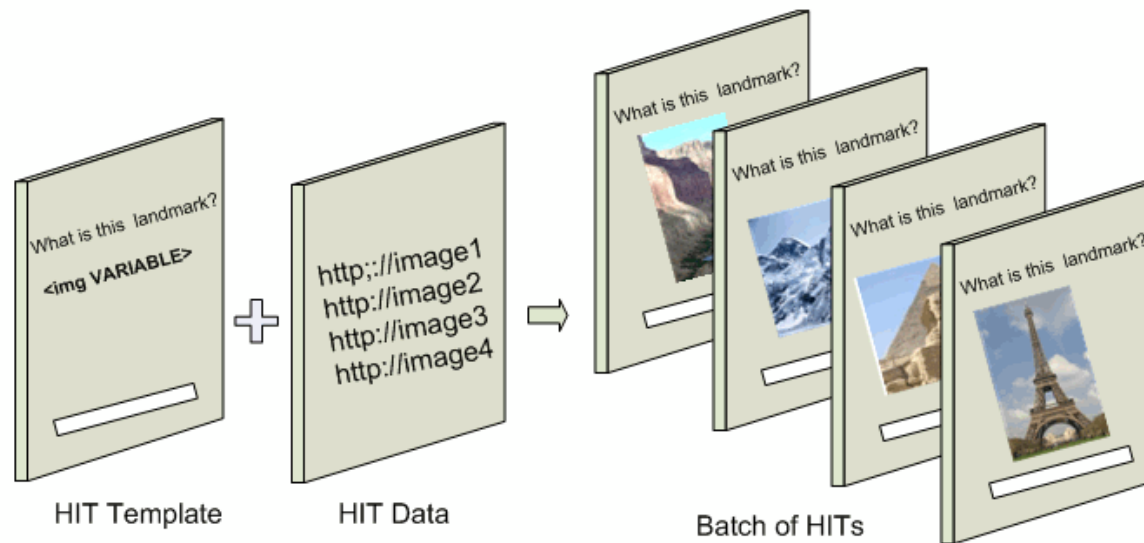
- Requestors
- HITs (Human Intelligence Tasks)
- Assignment
- Workers ('Turkers')
- Approval and Payment
- Qualification

Amazon Turk Pipeline



HIT Template

- HTML page that presents HITs to workers
 - ▣ Non-variable: all workers see the same page
 - ▣ Variable: every HIT has the same format, but different content



HIT Template

- Define properties
- Design layout
- Preview

Sample HIT Templates		
HIT Template Name	HIT Title	
Basic Open-ended Question	Answer a Simple Question See an example	Start with this template
Blank Template	Default Title See an example	Start with this template
Categorization	Pick the best category See an example	Start with this template
Categorization using Masters	Pick the best category See an example	Start with this template
Data Collection	Find the Website Address for Restaurants See an example	Start with this template
Data Correction	Provide the correct spelling of search terms See an example	Start with this template
Data Extraction	Get Product Name from Image See an example	Start with this template
Image Filtering	Flag offensive content images (WARNING: This HIT may contain offensive content. Worker discretion is advised.) See an example	Start with this template
Image Tagging	Tag an image See an example	Start with this template

HIT Template

- Properties
 - ▣ Template Name
 - ▣ Title
 - ▣ Description
 - ▣ Keywords
 - ▣ Time Allowed
 - ▣ Expiration Date
 - ▣ Qualifications
 - ▣ Reward
 - ▣ Number of assignments
 - ▣ Custom options

HIT Template

- Design
 - HTML

The screenshot displays the 'Design Layout' tab of the HIT Template editor. At the top, there are three tabs: '1 Enter Properties', '2 Design Layout' (which is active), and '3 Preview and Finish'. Below the tabs, the 'Template Name' is set to 'Image Tagging' with a note that this name is not displayed to workers. The 'Frame Height' is set to '400' pixels, with a note explaining that this is the height in pixels of the frame where the HIT will be displayed to workers, and it should be adjusted to minimize scrolling. A rich text editor toolbar is visible, including options for Format, Font, and various text formatting tools like bold, italic, underline, and link. Below the toolbar is an 'Edit HTML Source' button. The main content area is titled 'Tag this image' and contains 'Guidelines' for tagging: at least 3 tags, single words, no more than 25 characters, and tags must describe the image or its context. Below the guidelines is a large text area labeled 'Image:' with a placeholder 'image_url'. At the bottom, there are three input fields for 'Tag 1:', 'Tag 2:', and 'Tag 3:'. The interface concludes with 'Save' and 'Preview and Finish' buttons.

1 Enter Properties 2 Design Layout 3 Preview and Finish

Template Name: Image Tagging This name is not displayed to workers.

Frame Height 400 Height in pixels of the frame your HIT will be displayed in to workers. Adjust the height appropriately to minimize scrolling for workers.

Format Font U I B T

Edit HTML Source

Tag this image

Guidelines:

- You must provide at least 3 tags for this image.
- Each tag must be a single word
- No tag can be longer than 25 characters
- The tags must describe the image, the contents of the image, or some relevant context.

Image:

image_url

Tag 1: Tag 2: Tag 3:

Save Preview and Finish

HIT Template

- Design

- ▣ Template Variables

- Variables are replaced by data from a HIT data file

```

```

HIT Template

□ Design

▣ Data File

■ .CSV file (Comma Separated Value)

The screenshot shows a web interface for a HIT template. A yellow-bordered window titled "HIT Data File" is overlaid on the main content. The window contains a table with two columns, A and B, and six rows. Row 1 contains the variable names "continent" and "image_url". Rows 2 through 5 contain example data: "North America", "Europe", "Africa", and "North America" in column A, and corresponding URLs in column B. Row 6 is empty. In the background, the main interface has a section titled "Tag this image" with guidelines. One guideline says "Name the landmark, for example, the G". Another says "Specify the country the landmark is in, for example, the US". Below this, there is a text input field with the placeholder "This landmark is in \${continent}.". There is also a placeholder for an image labeled "image_url". At the bottom, there are two input fields labeled "Landmark" and "Country".

	A	B
1	continent	image_url
2	North America	http://www.myServer.com/images/image1.gif
3	Europe	http://www.myServer.com/images/image2.gif
4	Africa	http://www.myServer.com/images/image3.gif
5	North America	http://www.myServer.com/images/image4.gif
6		

Row 1: Variable Names
Rows 2-5: Variable for
each HIT

HIT Template

□ Result

▣ Also .CSV

	A	B	C	D
1	CompanyName	City	State	Country
2	Amazon	Seattle	WA	USA
3	Zappos	Las Vegas	NV	USA
4	Starbucks	Seattle	WA	USA
5	Coke	Atlanta	GA	USA
6	Facebook	Menlo Park	CA	USA

Table rows separated by line breaks.
Columns separated by commas.

First row is a header with labels for each column.

```
1 CompanyName, City, State, Country
2 Amazon, Seattle, WA, USA
3 Zappos, Las Vegas, NV, USA
4 Starbucks, Seattle, WA, USA
5 Coke, Atlanta, GA, USA
6 Facebook, Menlo Park, CA, USA
```

HIT Template

□ Accessing assignment details in JavaScript

```
var assignmentId = turkGetParam('assignmentId', '');  
if (assignmentId != '' && assignmentId != 'ASSIGNMENT_ID_NOT_AVAILABLE') {  
    var workerId = turkGetParam('workerId', '');
```

```
function turkGetParam( name, defaultValue ) {  
    var regexS = "[\\?&]" + name + "=(^[&#]*)";  
    var regex = new RegExp( regexS );  
    var tmpURL = window.location.href;  
    var results = regex.exec( tmpURL );  
    if( results == null ) {  
        return defaultValue;  
    } else { return results[1]; }  
}
```

Function automatically included
by Amazon

Also commonly see a gup function
used for the same purpose

Publishing HITs

❑ Select created template

Select HIT Template ① Select HIT Template ② Upload Input Data ③ Preview ④ Confirm and Publish

All of the HITs in a batch will use the same HIT template. The HIT template describes the layout and properties of the HITs.

Your HIT Templates

	<u>HIT Template Name</u>	<u>HIT Title</u>	<u>Creation Date</u> ▼
Select	Image Tagging	Identify landmarks See an example	October 01, 2010 8:26 AM PDT

Publishing HITs

□ Upload Data File

Upload Input Data

1 Select HIT Template 2 Upload Input Data 3 Preview 4 Confirm and Publish

To specify the data that will replace the variables in your HIT template, choose a Comma Separated Values (CSV) file that contains the input data. If you want to include images in your HIT, you will need to provide the URL. ([learn more](#))

Image Tagging

Locate a New File

[Download](#) a sample of the input file for this HIT template or learn more about [acceptable file formats](#)

Or Choose an Existing File

Your Existing Files

	<u>File Name</u>	<u>Input Lines</u>	<u>Creation Date</u> ▼	
--	------------------	--------------------	------------------------	--

Publishing HITs

□ Preview and Publish

amazonmechanical turk

REQUESTER

HomeDesignPublishManageDeveloperHelp

Create HITs individually

Confirm and Publish Batch

1 Select HIT Template2 Upload Input Data3 Preview4 Confirm and Publish

Please review the information about the HIT batch, then click "Publish HITs".

Image Tagging

Batch Summary

Batch Name: 'Image Tagging' @ 22 Jun 20:37Description:

Batch Properties

Title:

Tag an image

Description:

Please view and tag the image contained in this HIT

Batch expires in:

7 Days

Results are automatically approved after:

3 Days

Master Qualification:

Photo Moderation Masters

Workers must meet the following Qualifications to work on these HITs:

HIT approval rate (%) score greater than or equal to 95

HITs

Number of HITs in this batch:

5

Number of assignments per HIT:

x

1

Total number of assignments in this batch:

5

Cost

Reward per Assignment:

\$0.030

x

5

(total number of assignments in this batch)

Estimated Total Reward:

\$0.150

Estimated Fees to Mechanical Turk:

+ \$0.025

(fees paid to Mechanical Turk) (fee details)

Estimated Total Cost:

\$0.175

(this is the amount that will be deducted from your Available Balance when you click "Publish HITs")

Your Available Balance:

\$53,194.021

(before clicking "Publish HITs")

Your Projected Balance:

\$53,193.846

(after clicking "Publish HITs")

BackPublish HITs

Qualification

□ Qualification

- ▣ Make sure that a worker meets some criteria for the HIT
 - 95% Approval rating, etc.
- ▣ Requester User Interface (RUI) doesn't support Qualification Tests for a worker to gain a qualification
 - Must use Mechanical Turk APIs or command line tools

Masters

- Workers who have consistently completed HITs of a certain type with a high degree of accuracy for a variety of requestors
 - ▣ Exclusive access to certain work
 - ▣ access to private forum
- Performance based distinction
- Masters, Categorization Masters, Photo Moderation Masters – superior performance for thousands of HITs

Command Line Interface

- Abstract from the “muck” of using web services
- Create solutions without writing code
- Allows you to focus more on solving the business problem and less on managing technical details
- mturk.properties file for keys and URLs
- Input: *.input, *.properties, and *.question files
- Output: *.success, and *.results

*.input

- Tab delimited file
- Contains variable names and locations

Image1	Image2	Image3
Image1.jpg	Image2.jpg	Image3.jpg

Image1	Image2	Image3
Image1.jpg	Image2.jpg	Image3.jpg

*.properties

- Title
- Description
- Keywords
- Reward
- Assignments
- Annotation
- Assignment duration
- Hit lifetime
- Auto approval delay
- Qualification

*.question

- XML format
- Define the HIT layout
- Consists of:
 - ▣ <Overview>: Instructions and information
 - ▣ <Question>
- Can be a QuestionForm, ExternalQuestion, or a HTMLQuestion

<Question>

- *QuestionIdentifier
- DisplayName
- IsRequired
- *QuestionContent
- *AnswerSpecification
 - ▣ FreeTextAnswer, SelectionAnswer, FileUploadAnswer

<Question>

```
<Question>
  <QuestionIdentifier>my_question_id</QuestionIdentifier>
  <DisplayName>My Question</DisplayName>
  <IsRequired>true</IsRequired>
  <QuestionContent> [...] </QuestionContent>
  <AnswerSpecification> [...] </AnswerSpecification>
</Question>
```

<QuestionContent> (and <Overview>) can contain:

- <Application>: JavaApplet or Flash element
- <EmbeddedBinary>: image, audio, video
- <FormattedContent> (later)

*.success and *.results

- *.success: tab delimited text file containing HIT IDs and HIT Type IDs
 - ▣ Auto-generated when HIT is loaded
 - ▣ Used to generate *.results
- Submitted results in the last columns
 - ▣ generate *.results with getResults command
 - ▣ tab-delimited file, last columns contain worker responses

Command Line Operations

- ApproveWork
- getBalance
- getResults
- loadHITs
- reviewResults
- grantBonus
- updateHITs
- etc

Loading a HIT

- ❑ `loadHITs -input *.input -question *.question -properties *.properties -sandbox`
- ❑ `-sandbox` flag to create HIT in sandbox to preview
- ❑ `-preview` flag also available
 - ▣ requires XML to be written in a certain way

FormattedContent

- Use FormattedContent inside a QuestionForm to use XHTML tags directly
 - No JavaScript
 - No XML comments
 - No element IDs
 - No class and style attributes
 - No <div> and elements
 - URLs limited to http:// https:// ftp:// news:// nntp://
mailto:// gopher:// telnet://
 - Etc.

FormattedContent

- Specified in XML CDATA block inside a FormattedContent element

```
<QuestionContent>
  <FormattedContent><![CDATA[
    <font size="4" color="darkblue" >Select the image below that best represents:
      Houses of Parliament, London, England</font>
  ]]></FormattedContent>
</QuestionContent>
```

Qualification Requirements

- qualification.1: qualification type ID
- qualification.comparator.1: type of comparison (greaterthan, etc.)
- qualification.value.1: integer value to be compared to
- qualification.locale.1: locale value
- qualification.private.1: public or private HIT
- Increment the *.1 to specify additional qualifications

*.properties

□ *.properties example

```
qualification.1:0000000000000000000LO  
qualification.comparator.1:greaterthan  
qualification.value.1:25  
qualification.private.1:false
```

Qualification TypeId
for percent
assignments approved

- Worker must have 25% approval rate and HIT can be previewed by those that don't meet the qualification

External HIT

□ Use an ExternalQuestion

```
<ExternalQuestion
xmlns="http://mechanicalturk.amazonaws.com/AWSMechanicalTurkDataSchemas/2006-07-14/ExternalQuestion.xsd">
  <ExternalURL>http://s3.amazonaws.com/mturk/samples/sitecategory/externalpage.htm?url=${helper.urlencode($urls)}</ExternalURL>
  <FrameHeight>400</FrameHeight>
</ExternalQuestion>
```

- `${helper.urlencode($urls)}` to encode urls from `*.input` to show in `externalpage.htm`

External HIT

□ In the external .htm:

```
<form id="mturk_form" method="POST"  
action="http://www.mturk.com/mturk/externalSubmit">  
(...question...)
```

And then submit the assignment to Mturk

```
if (gup('assignmentId') == "ASSIGNMENT_ID_NOT_AVAILABLE") {  
    ...  
} else {  
    var form = document.getElementById('mturk_form');  
    if (document.referrer && ( document.referrer.indexOf('workersandbox') != -1 ) ) {  
        form.action = "http://workersandbox.mturk.com/mturk/externalSubmit";  
    }  
}
```


Other Useful Options

- *.question
 - ▣ Create five questions, where the first 3 are required

```
#set( $minimumNumberOfTags = 3 )
#foreach( $tagNum in [1..5] )
<Question>
<QuestionIdentifier>tag${tagNum}</QuestionIdentifier>
#if( $tagNum <= $minimumNumberOfTags )
<IsRequired>true</IsRequired>
#else
<IsRequired>false</IsRequired>
#end
```

Qualification Test

- Given a request for a qualification from a worker, you can:
 - ▣ Manually approve qualification request
 - ▣ Provide answer key and Mturk will evaluate request
 - ▣ Auto-grant qualification
- Qualifications can also be assigned to a worker without a request

Qualification Test

- *.question, *.properties, *.answer
- Define the test questions in *.question and answers in *.answer

```
createQualificationType -properties qualification.properties  
                        -question qualification.question  
                        -answer qualification.answer  
                        -sandbox
```

Qualification Test (Question)

```
<QuestionForm
xmlns="http://mechanicalturk.amazonaws.com/AWSMechanicalTurkDataSchemas/2005
-10-01/QuestionForm.xsd">
  <Overview>
    <Title>Trivia Test Qualification</Title>
  </Overview>
  <Question>
    <QuestionIdentifier>question1</QuestionIdentifier>
    <QuestionContent>
      <Text>What is the capital of Washington state?</Text>
    </QuestionContent>
    <AnswerSpecification>
      ...
    </AnswerSpecification>
  </Question>
</QuestionForm>
```

Qualification Test (Answer Key)

```
<?xml version="1.0" encoding="UTF-8"?>
<AnswerKey
xmlns="http://mechanicalturk.amazonaws.com/AWSMechanicalTurkDataSchemas/2005-
10-01/AnswerKey.xsd">
  <Question>
    <QuestionIdentifier>question1 </QuestionIdentifier>
    <AnswerOption>
      <SelectionIdentifier>1b</SelectionIdentifier>
      <AnswerScore>10</AnswerScore>
    </AnswerOption>
  </Question>
</AnswerKey>
```

Auto-assign qualification and score with answer key

Qualification Test Properties

- ❑ name
- ❑ description
- ❑ keywords
- ❑ retrydelayinseconds
- ❑ testdurationinseconds
- ❑ autogranter

Matlab Turk Tool

```
aws_access_key = ;  
aws_secret_key = ;  
sandbox = true;
```

Initialize with keys and sandbox option

```
turk = InitializeTurk(aws_access_key, aws_secret_key, sandbox);
```

```
result = RequestTurk(turk, 'GetAccountBalance',  
{'ResponseGroup.0','Minimal','ResponseGroup.1','Request'});  
result.GetAccountBalanceResponse.GetAccountBalanceResult.AvailableBalance.Amount  
.Text
```

Command line operation

[Parameters](#)

[Operations](#)

Matlab Turk Tool

```
<GetAccountBalanceResult>
  <Request>
    <IsValid>True</IsValid>
  </Request>
  <AvailableBalance>
    <Amount>1 0000.000</Amount>
    <CurrencyCode>USD</CurrencyCode>
    <FormattedPrice>$10,000.00</FormattedPrice>
  </AvailableBalance>
</GetAccountBalanceResult>
```

result.GetAccountBalanceResponse.**GetAccountBalanceResult**.**AvailableBalance**.**Amount**.Text

Paid By Bonus

- Approve individually or by batch
- Reject individually or by batch
- Give bonuses to good workers
- Can download batch into a .CSV, mark accept/reject, then upload updated .CSV to the Mechanical Turk

TurkCleaner

- Have the user select a subset of images that satisfy certain rules.

Is this a **office cubicles**?

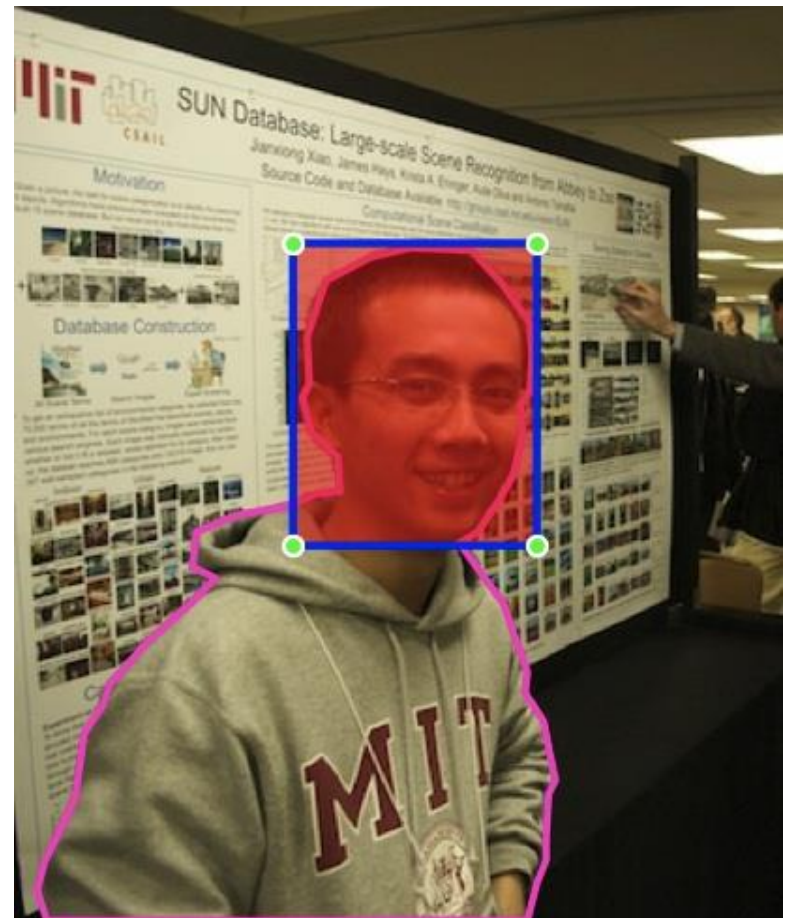
Yes



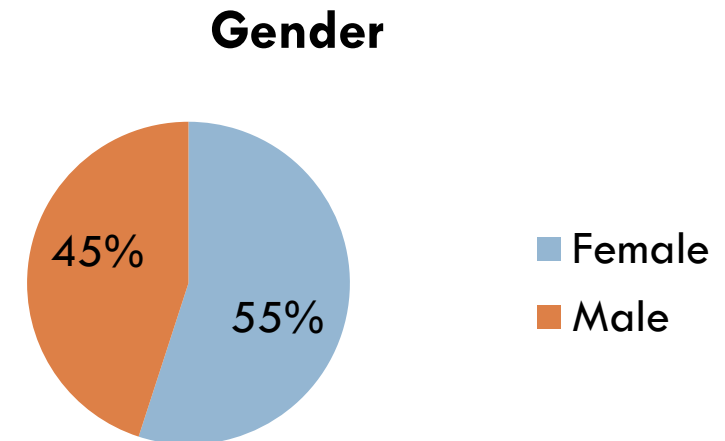
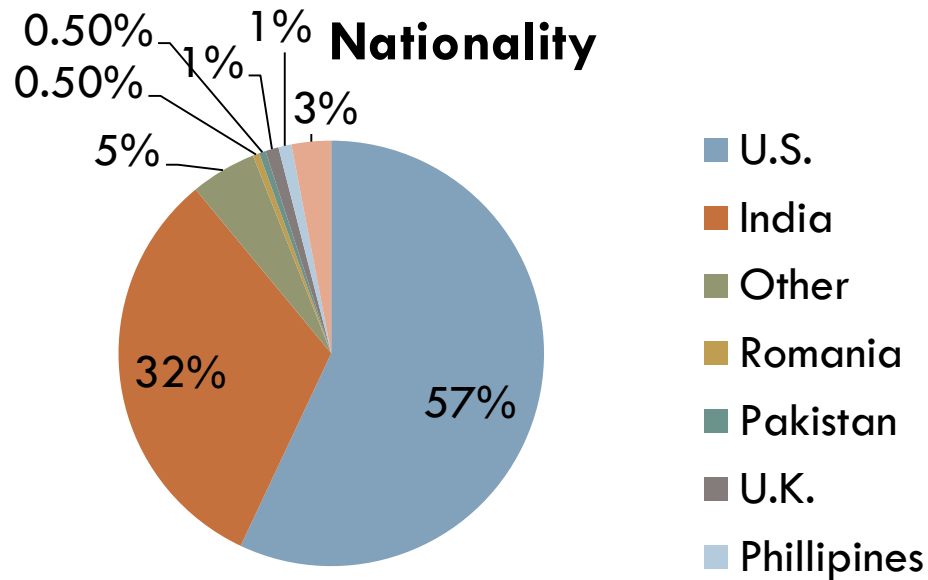
Copy .html into template, parse .CSV into Matlab readable format

DrawMe

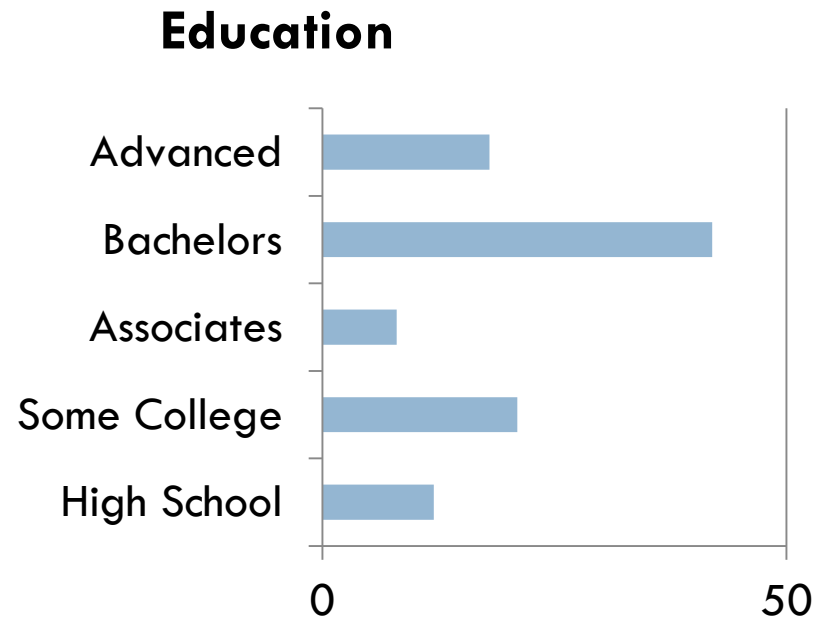
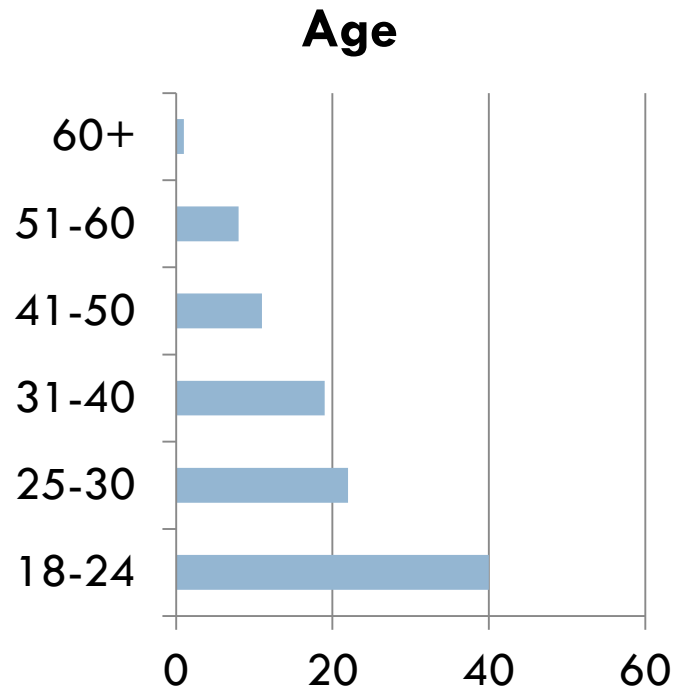
- Line drawing on an image.
- Copy .html into Mturk template
- .CSV file can be parsed into Matlab cell arrays for processing



Demographics



Demographics



Best Practice

- Motivation
 - ▣ Incentives: entertainment, altruism, financial reward
- Task Design
 - ▣ Easy to understand visuals, design interface such that accurate task completion requires as much effort as adversarial task completion, financial gain for amount of work tradeoff for worker
 - ▣ Creation task vs. Decision task
- High Quality Results
 - ▣ Heuristics such as gold standard and majority vote
- Cost Effectiveness

Creation Task vs Decision Task

- Creation:

- ▣ Write a description of an image

- Decision:

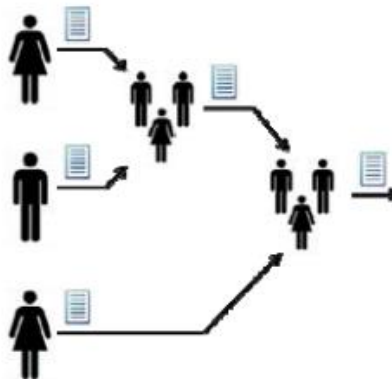
- ▣ Given two descriptions for the same image, decide which description is best

Iterative and Parallel

- Iterative: sequence of tasks, where each task's result feeds into the next task (better average response)



- Parallel: workers are not shown previous work (better best response)



Task Design

(A) Easy for Humans



Chair? Airplane? ...

(B) Hard for Humans



Finch? Bunting?...

(C) Easy for Humans



Yellow Belly? Blue Belly? ...

Gold Standard

- Present workers with control questions where the answer is known to judge the ability of the worker.
- Requires keeping track of workers over time or presenting multiple questions per task.



Majority Vote

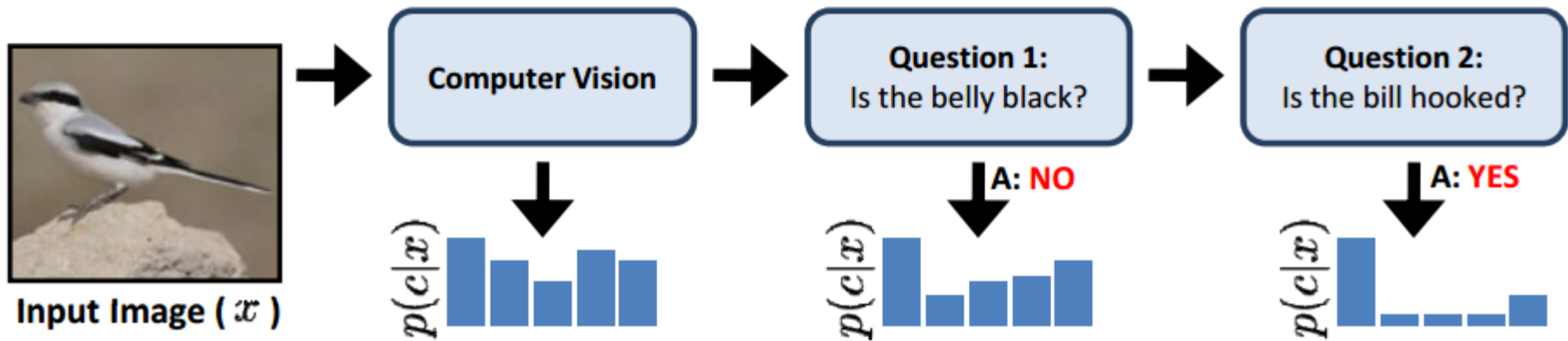
- Check the responses from multiple turkers against each other.
- Averaging multiple labels, etc.

Cost Effectiveness

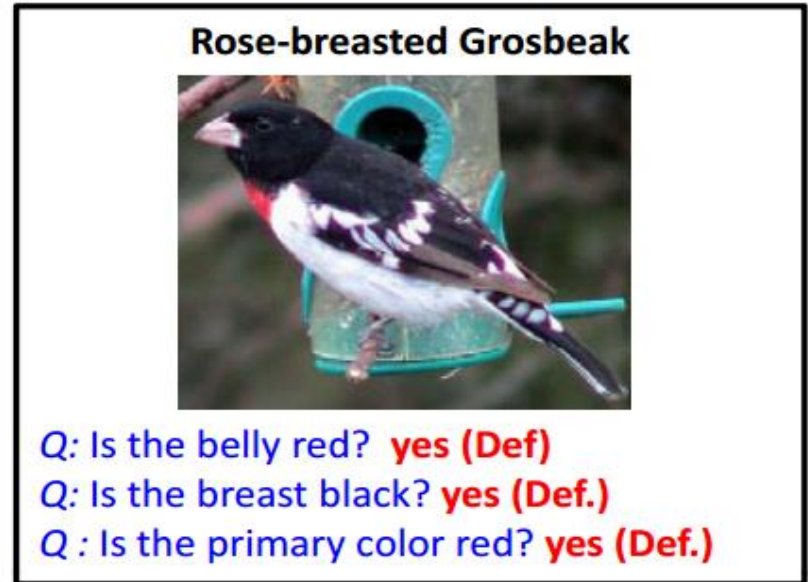
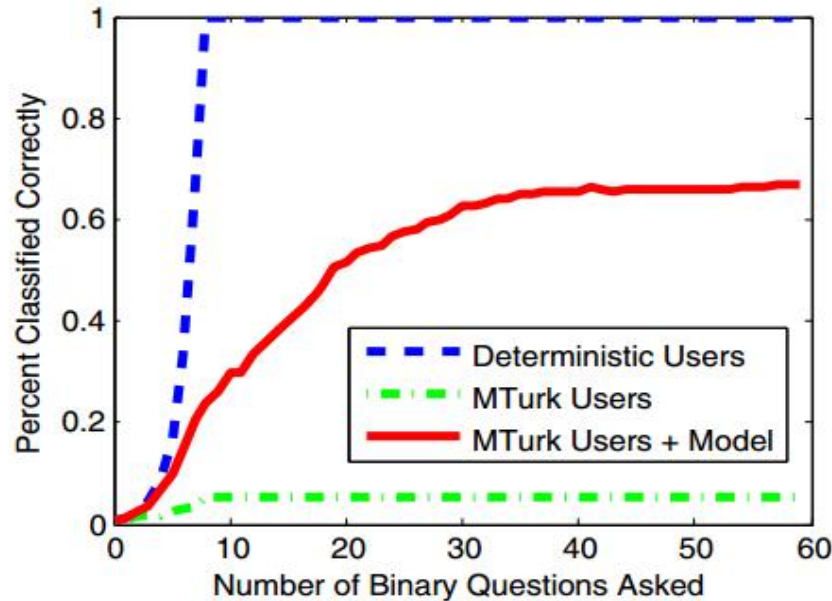
- [Welinder, et. al.] Estimation of annotator reliabilities
 - ▣ Use the reliability of the annotator to determine how many additional labels are needed to correctly label the image.

Augmenting Computer Vision

- Using humans to improve performance



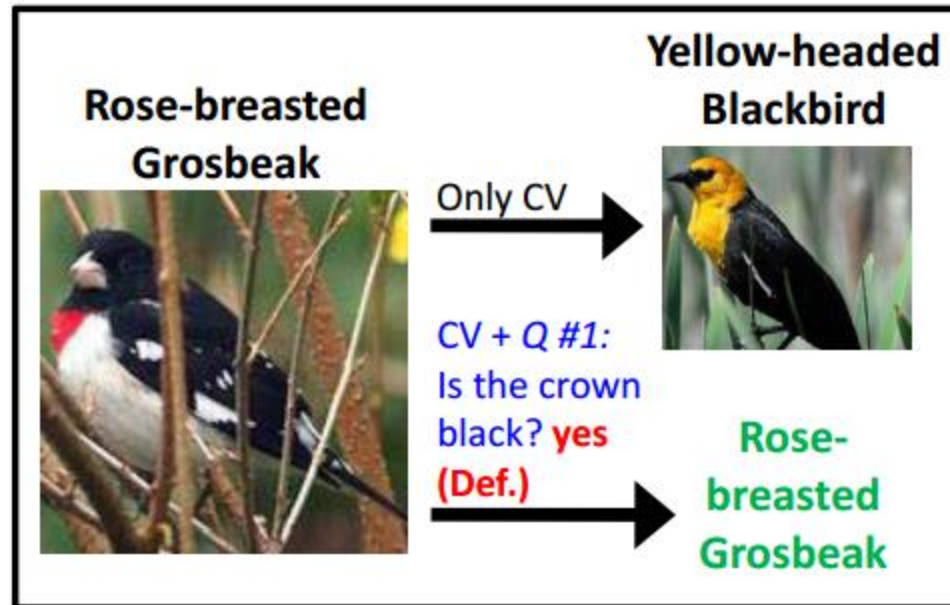
Augmenting Computer Vision



- Deterministic Users: assumed perfect users
- Turkers: subjective answers degrade performance (brown vs buff)



Augmenting Computer Vision



- Human answer corrects computer vision's initial prediction

TurKit



- Toolkit for prototyping and exploring algorithmic human computation

TurKit Script

- extension of JavaScript
- wrapper for MTurk API

```
ideas = []
for (var i = 0; i < 5; i++) {
  idea = mturk.prompt(
    "What's fun to see in New York City?
    Ideas so far: " + ideas.join(", "))
  ideas.push(idea)
}

ideas.sort(function (a, b) {
  v = mturk.vote("Which is better?", [a, b])
  return v == a ? -1 : 1
})
```

Generates ideas for things to see from 5 different workers and getting workers to sort the list

Crash-and-rerun programming

- ❑ Script is executed until it crashes
- ❑ Every line that is successfully run is stored in a database
- ❑ If script needs to be rerun, cost of rerunning human computation task is avoided by looking up the previous result (**use keyword once**)
- ❑ waitForHIT function that crashes unless results are ready

TurKit: Quicksort

```
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) A
        left.add(x)
      else
        right.add(x)
    quicksort(left)
    quicksort(right)
    A.set(left + pivot + right)

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

Use **once** if function is:

- deterministic
 - once `Math.random()` would result in the same value every run
- high cost
- has side-effects
 - ex: approving results from a HIT multiple times causes errors

TurKit: Parallelism

```
fork(function () {  
    a = createHITAndWait()    // HIT A  
    b = createHITAndWait(...a...) // HIT B  
})  
fork(function () {  
    c = createHITAndWait()    // HIT C  
})
```

- If HIT A doesn't finish, crash that fork and the next fork creates HIT C
- Subsequent runs will check each HIT to see if it's done
- join() to ensure previous forks were successful
 - ▣ if previous forks unsuccessful, join crashes current path

TurKit IDE



Turker Forum and Browser Plugin

- Turkoption: (Union 2.0) shows reviews of requestors on Amazon MTurk
- TurkerNation
- Helpful Blogs for Requestors:
 - [\[Tips for Requestors\]](#)
 - [\[The Mechanical Turk Blog\]](#)

