“Building New Software With Artificial Artificial Intelligence”

Jeff Barr
Web Services Evangelist, Amazon.com
1200 12th Avenue South
Seattle, WA 98144
United States of America
jbarr@amazon.com

ABSTRACT

In 2005, Amazon.com introduced the concept of “Artificial Artificial Intelligence,” through its Amazon Mechanical Turk (AMT) offering. AMT provides a web services API (Application Programming Interface) which allows software developers to easily and economically build programs that tap into a world-wide, internet-scale human workforce on an incremental, as-needed basis. This paper reviews Amazon’s motivation for building AMT, describes the current set of offerings, and demonstrates several real-world applications.

General Terms

# Economics

# Experimentation

# Human Factors

Keywords

Artificial intelligence, API, human intelligence, software, user interface, Web services.

1. INTRODUCTION

In the 50 years since John McCarthy coined the term “artificial intelligence,” much progress has been made toward identifying, understanding, and then automating many classes of symbolic and computational problems that were once the exclusive domain of human intelligence. However, much work remains to be done in the field and sometimes it appears that applying good old human brainpower is still the best way to get the job done.

Over 300 years ago, Wolfgang von Kempelen built an automaton that defeated many human opponents at a chess board. Known as “The Turk,” the wooden mannequin toured the United States and Europe for many years, defeating such famous challengers as Benjamin Franklin, Napolean Bonaparte, and Edgar Allen Poe. The secret to the automaton was, of course, a human chess master hidden inside.

The Amazon Mechanical Turk[1] introduces the concept of “Artificial Artificial Intelligence,” providing a web services API (Application Programming Interface) which allows software developers to easily and economically build programs that tap into a world-wide, internet-scale human workforce on an incremental, as-needed basis. Like its namesake, Amazon’s Mechanical Turk presents a mechanical front to conceal, or abstract, the human processing power and intelligence hidden inside.

2. GENESIS

Several years ago, Amazon identified a number of internal tasks that would be amenable to high-volume processing by a workforce composed of human beings with particular skills. Some of the initial tasks included:

Data Improvement: Thousands of merchants load data for millions of products into Amazon’s catalog, often leading to conflicting, missing, or erroneous product information. Human processing is the best possible arbiter of conflicts after all conceivable automatic checking has been done.

Japanese Text Orientation: Japanese text can be written left to right or top to bottom. As part of Amazon’s effort to create searchable indices from scanned images of book content, the text recognition system must be informed of the text direction. Anyone fluent in written Japanese can quickly and efficiently glance at a scanned image and identify the text direction.

Image Selection: Amazon’s BlockView image technology aggregates millions of street-side images to create a scrollable panoramic street view in the context of an online business directory. After automatic processing has chosen several candidate images, human intelligence is used to choose the best possible image to represent each street address and business.

These tasks, along with many others not listed, shared a number of common attributes:

High Business Value: Each task made a contribution to Amazon’s asset base in a small yet measurable way. The

[1] www.mturk.com.

aggregate value of the completed task was high enough to have a meaningful impact on the quality of Amazon's catalog or of other digital assets.

High Volume: The amount of work to be done was high, often numbering in the millions or even tens of millions of individual work units.

Self-contained: Each task was self-contained and required little, if any, global context.

Human-centric: Each task could make good use of human skills that are either impossible or prohibitively expensive to fully automate.

Varied Demand: The amount of work to be done varied from day to day. One day there might be a surge of millions of work units. The next day there might be demand for an entirely different type of work. The varying levels and types of work ruled out simply adjusting staffing levels on a day-by-day, task-by-task basis.

Key challenges in building this system included scalability, accountability, reputation tracking, quality control, and flexibility. Let's consider each of these in turn:

Scalability was necessary because the system as envisioned would be able to manage millions or even tens of millions of in-process tasks per day. Large volumes of work could arrive at any time, and many workers could log in and address these tasks concurrently.

Reputation tracking was necessary in order to provide a long-term incentive for workers to do the best possible job. The initial tasks would pay a modest amount, sometimes 1 or 2 cents (US) per task. Without a system to track and control reputations of individual workers, the system would quickly degenerate into chaos.

Accountability is closely akin to reputation. Individual workers must have an identity within the system, and they need to recognize that their work is of value to the requesting organization. On the other side of the transaction, the requesting organization must be accountable to the workers, managing quality control and payments on a fair, equitable, and timely basis.

Quality control was needed in order to ensure that requesting organizations received work of acceptable accuracy for their money.

Finally, **flexibility** would be paramount. This was to be a general-purpose system for use by Amazon and by others. All potential users of the system would want to efficiently create and process tasks of many different types.

3. THE AMAZON MECHANICAL TURK

Amazon's Mechanical Turk provides the interface for computers to make requests of human beings. The system provides SOAP[2] and REST[3] interfaces for the creating and management of work units, also known as HITS or Human Intelligence Tasks. Software applications make calls to Mechanical Turk's web services interface to request that human beings perform tasks best suited to human intelligence. These tasks include those listed above, and many others, including translating paragraphs of text from one language to another, describing a photograph, or identifying a sound. Human beings capable of performing those tasks find, accept, and complete them, and then register the results with the Mechanical Turk. The requesting application is then notified when the tasks are complete and results are available. Each task includes payment information, and the human being is paid as soon as the work is found to be acceptable by the requesting organization.

[2] www.w3.org/TR/soap/.

[3] www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.

The Mechanical Turk system manages task submission, assignment, and completion, matches qualified people with tasks that require particular skills, provides a feedback mechanism to encourage quality work, stores task details and results, all behind a web services interface.

The five key Mechanical Turk concepts are Workers, HITS, Qualifications, Assignments, and Requesters.

Workers are human beings who want to earn money by working on HITS. Each worker is presumed to have some skills that are of potential applicability to Mechanical Turk HITS.

Qualifications are tests or assertions used to ensure that only properly qualified Workers have access to certain HITS. Qualifications can verify that a particular Worker has a particular skill, such as the ability to read French. They can also verify the Worker's ability to perform other HITS at a desired rate of success. Qualifications can be machine-graded against an answer key, or they can be manually graded by the Requester.

HITS are Human Intelligence Tasks, or individual work units. Each HIT is a fine-grained task, such as "Is there a dog in this picture?" or "Is this Japanese text vertical or horizontal?" Each HIT can have any number (zero or more) associated Qualifications. HITS are specified using the Question Language and are ultimately rendered as part of a web-based user interface. HITS can present text and graphical data to the Worker and can accept the Worker's responses using standard HTML form elements such as text input fields, radio buttons, dropdown menus, and check boxes.

Assignments represent mappings of HITS to Workers. When a worker decides to perform a particular HIT, the HIT is said to be *assigned* to the Worker. The Requester is able to specify the desired number of Assignments for each HIT. This feature can be used to implement a quality control system using plurality (see sidebar, Plurality). Of importance is the fact that any given HIT will never be assigned to the same worker more than once, regardless of the number of Assignments that the Requester has specified for the HIT.

Requesters are individuals or organizations with work to be done. The requesters typically use a software application to submit tasks on their behalf. This application uses the Mechanical Turk's web service interface to load the tasks and qualifications, approve completed work, and to retrieve results. Requesters must also deposit payment information into their Amazon.com account prior to loading tasks.

4. SYSTEM WORKFLOW

The Requesters, the Qualifications, HITS, and the Workers all interact at the Mechanical Turk web site (http://www.mturk.com). Let's take a step-by-step look at how all of this comes together.

Preparation: The Requester starts by identifying some work to be done and designing the HIT. Good-quality HITS are self-contained, context-free, and expressible using the system's Question Language (see sidebar). The Requester also defines the Qualifications, also expressed using the Question Language, and decides on the payment (price per Assignment) to be paid to the Workers. The Requester can set the price that they are willing to pay to any value from 1 US cent on up.

Funding: The Requester makes a deposit in their Amazon account. This deposit must be sufficient to pay the Workers for all of the work to be loaded. Reliance on a deposit in advance of the work protects the Workers against unscrupulous Requesters who could otherwise get work done without the wherewithal to pay for it. The Requester must also deposit an additional 10% over and above what they will pay the Workers; this represents Amazon's fee for operating the Mechanical Turk service

Initialization: The Requester's application makes a series of web service calls to load the Qualifications and the HITS into the Mechanical Turk. As part of the response data from each web service call, the system returns identifiers for the Requester to use as part of the approval process. The HITS are available immediately for Workers to act upon.

Work: Workers periodically visit the Mechanical Turk site to check for work to be done. Publicity for new types of HITS is also generated within the Worker community using a number of blogs[4] and online discussion forums. Workers look for HITS that are of interest to them, and for which they can meet any requisite Qualifications. The Workers then endeavor to do the work, accepting HITS and returning completed results back to the system for approval. The system tracks a multitude of statistics for each Worker and for each Requester.

Approval: As soon as the Requester has loaded a batch of HITS into the system, it will begin polling for reviewable HITS – HITS where the requested number of Assignments have taken place and been submitted by Workers. Each polling cycle will return all such Assignments to the Requester. The Requester then performs any final checking or other quality control measures (perhaps using the Plurality model described in the sidebar), and then approves each Assignment that was found to be acceptable

Finalization: As soon as Assignments are approved by the Requester, the corresponding payments are released to the Workers. The Requester is able to aggregate results from their entire batch of HITS for use within their own processing.

5. SYSTEM INTERFACE

As noted previously, the Requesters interact with the Mechanical Turk by means of its web services interface. Requesters typically build application programs using popular languages such as C++, C#, Java, or PHP. The application built by the Requester effectively serves as a bridge and a coordinator between the Requester's internal data and systems and the Mechanical Turk. For example, if the Requester were to use the Mechanical Turk to process a number of graphical images owned by the Requester, the application would be responsible for copying those images from the Requester's private storage into the HITS as well as for copying the answers or other information provided by the Workers back into other storage managed or owned by the Requester.

Each web service request is signed using the HMAC[5] algorithm. HMAC is a cryptographic hashing function used here to authenticate the request. By insisting on signed requests our system is able to know with a high degree of confidence that the Requester is making requests on their own behalf rather than on someone else's. Amazon supplies each registered software developer with access to the private and public keys that are needed to sign the message. For efficiency reasons (the HMAC is computationally expensive) we expect only certain fields of each request to be signed.

Here are some of the more important web service calls:

CreateQualificationType: Creates a Qualification that can be subsequently attached to any number of HITS.

CreateHIT: Creates a new HIT given a title, description, question data, and qualification list.

GetReviewableHITs: Returns the list of HITs that are ready to be reviewed.

GetAssignmentsForHIT: Returns the list of completed Assignments for a given HIT. This call is typically used in conjunction with GetReviewableHITs in order to process all Assignments for all reviewable HITS.

ApproveAssignment: Signifies approval of a HIT Assignment performed by a Worker and also releases payment to the Worker.

GetHIT: Returns the data which describes the HIT.

DisposeHIT: Destroy all memory of a HIT.

GrantQualification: Attach a particular type of Qualification to a Worker, signifying that they have successfully demonstrated that they possess a particular skill.

NotifyWorker: Send a notification message from Requester to Worker via the Mechanical Turk system.

Potential Requesters are able to use the web services interface to connect their application and their business logic to the Mechanical Turk, making it an integral part of their business workflow.

6. CHECKS AND BALANCES

Integral to the success of the Mechanical Turk concept is the presence of a set of checks and balances. These checks and balances protect the system from intentional or accidental misuse by Workers or by Requesters. A principal defensive tactic is the use of a number of statistical measures. Statistics are kept on a per-Worker basis for such values as:

- Total number of HITs attempted
- Total number of HITs completed
- Total number of HITs accepted by the Requesters
- Total number of HITs abandoned

Additional statistics are tracked for each type of HIT processed by each Worker.

Similar statistics are kept for Requesters, although they are not currently made available for external use.

[4] www.turking.com and www.mechturkblog.com are just two of the many blogs that sprung up in the days and weeks following the beta launch of the system.

[5] http://en.wikipedia.org/wiki/HMAC.

7. POSSIBLE USES FOR THE MECHANICAL TURK

As a simple, efficient way to access an internet-scale workforce, the Mechanical Turk can be used in an almost infinite number of different ways. Here's a small sampling of some that we have collected to date. Some of these are actual, finished applications, others are ideas ripe for the picking.

Podcast Transcription: This has been implemented at www.castingwords.com. The site handles the process of accepting the podcast, selecting the episode(s) to be transcribed, and accepting payment instructions. The selected episodes are then mapped to HITs where they are accessible to pre-qualified Workers. The work in each HIT consists of listening to a single podcast episode and then generating a high-quality text transcript of the conversation found therein.

Language Translation: The system has been used for English to French and French to English translation, and can (of course) be used for any possible combination of natural languages.

Catalog Data Improvement: Amazon has used the system to perform a number of data improvement and validation processes on our product catalog.

Data Gathering: Several Requesters are now using the Mechanical Turk to collect and evaluate lists of "Top 3" items (restaurants, theaters, and so forth) on a city-by-city basis.

Image Tagging: Given an image, the task is to enter a small number of descriptive tags which characterize the image.

Web Site Review: Given a link to a web site, review the site and answer a series of multiple choice questions about the site.

Marketing Survey: Answer a series of qualifying questions and then take a marketing survey.

Sound Verification: Listen to a sound and verify that the sound matches the description.

Facial Image Verification: Compare two facial images and decide if they are of the same person or not.

An important variant on most of the above HITs is the use of a secondary HIT to verify the first. For example, high-quality translation of French to English has been implemented using a pair of HITs. The first HIT is to translate the text; the second is to verify the accuracy of the translation.

8. LOOKING TOWARD THE FUTURE

Existing businesses, as well as those now in the formative stages, should look to the Mechanical Turk model as an infrastructure component that will give them the ability to tap in to an on-demand, Internet-scale workforce. We look forward to seeing the creative applications and business models that will be built around the system.

9. SIDEBAR: PLURALITY

Plurality is an important quality control mechanism in the Mechanical Turk universe. Using plurality, Requesters can detect and protect themselves from low-quality workers. Let's say that the HIT contains an image, and the question put to the Worker is "Is there a dog in this picture?" To use Mechanical Turk to get a high-quality answer to this question using a plurality system it is loaded into the system with a maximum assignment count of 5. The system ensures that any particular Worker can see the HIT at most one time. As soon as a majority of the Workers (in this case, 3 out of 5) agree on the result, the Requester can accept that result as the answer and proceed. If no plurality emerges, this often means that the HIT is ambiguous

10. SIDEBAR: QUESTION LANGUAGE

The Mechanical Turk Question Language allows for the following types of elements in the questions:

- Text
- Bulleted list
- Binary data with associated MIME type
- Radio button
- Dropdown list
- Checkbox
- Multiple choice

11. REFERENCES

[1] Amazon Mechanical Turk. Retrieved February 9, 2006. www.mturk.com.

[2] Friedling, T.R. Architectural Styles and the Design of Network-based Software Architectures. Retrieved February 9, 2006 from University of California, Irvine. www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.

[3] HMAC. Retrieved February 9, 2006 from Wikipedia. http://en.wikipedia.org/wiki/HMAC.

[4] Latest SOAP Versions. Retrieved February 9, 2006 from World Wide Web Consortium. www.w3.org/TR/soap.