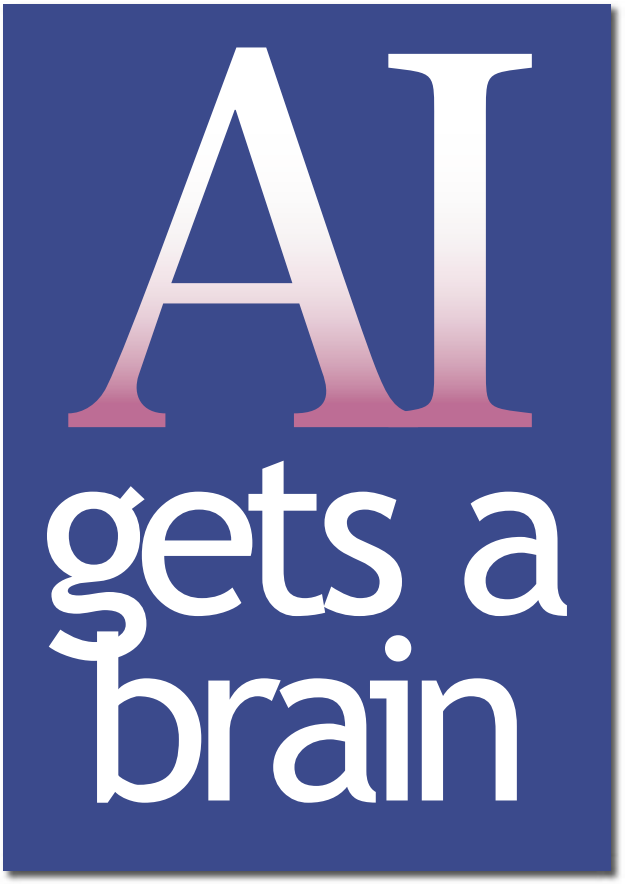
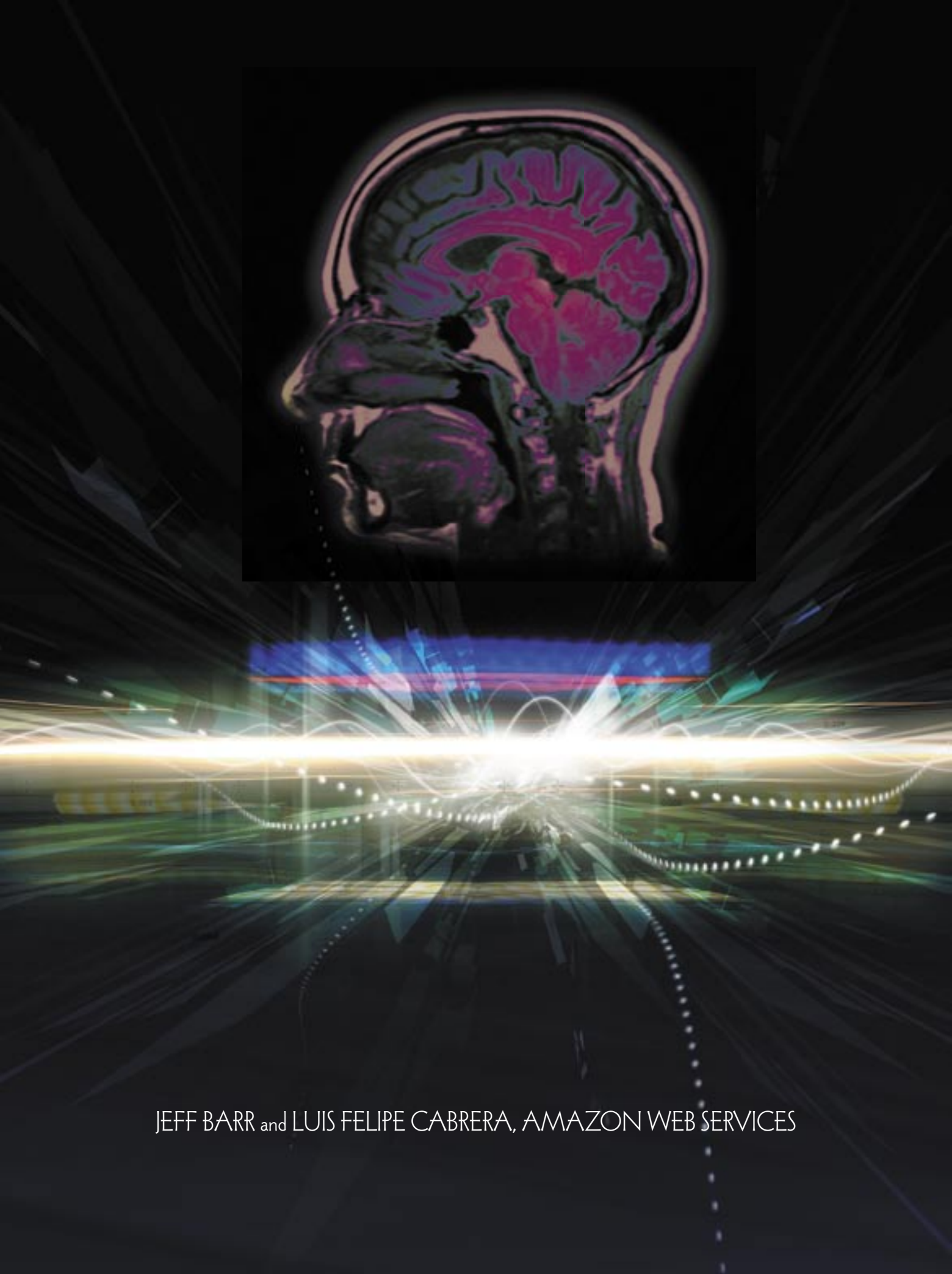
New technology allows software to tap real human intelligence.



In the 50 years since John McCarthy coined the term *artificial intelligence*, much progress has been made toward identifying, understanding, and automating many classes of symbolic and computational problems that were once the exclusive domain of human intelligence. Much work remains in the field because humans still significantly outperform the most powerful computers at completing such simple tasks as identifying objects in photographs—something children can do even before they learn to speak.

Software developers with innovative ideas for businesses and technologies are constrained by the limits of artificial intelligence. In today's business landscape where companies are more cost-conscious than ever, projects that require a vast network of humans are scrutinized with a fine-tooth comb and often scrapped because the cost of establishing and managing a network of skilled people to do the work outweighs the value of completing it. If software developers could programmatically access and incorporate human intelligence into their applications, a whole new class of innovative businesses and applications would be possible. This is the goal of Amazon Mechanical Turk:[1] to give software developers and businesses the power to use human intelligence as a core component of their applications and businesses. With Amazon Mechanical Turk, people are freer to innovate because they can now imbue software with real human intelligence.

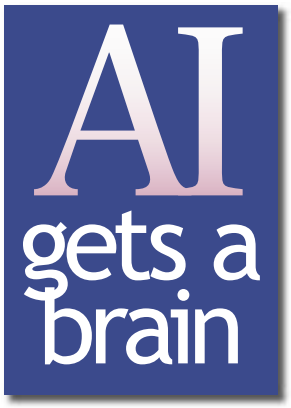
In 1769, Wolfgang von Kempelen built an automaton that defeated many human opponents at chess. Known as "The Turk," the wooden mannequin toured the United States and Europe for many years, defeating such famous challengers as Benjamin Franklin, Napolean Bonaparte, and Edgar Allen Poe.[2] The secret to the automaton was, of course, a human chess master hidden inside. Like its namesake, Amazon's Mechanical Turk presents a mechanical front to conceal, or abstract, the human processing power and intelligence hidden inside. Developers can use the Amazon Mechanical Turk Web services API to submit tasks to the Amazon Mechanical Turk Web site, approve completed tasks, and incorporate the answers into their software applications. To the application, the transaction looks very much like any remote procedure call: The application sends the request, and the service returns the results. In reality, a network of humans fuels this "artificial artificial intelligence" by coming to the Web site, searching for and completing tasks, and receiving payment for their work. This allows software developers to easily and economically build programs that tap into a worldwide, massively parallel, Internet-scale human workforce on an incremental, as-needed basis.



JEFF BARR and LUIS FELIPE CABRERA, AMAZON WEB SERVICES

GENESIS

Amazon identified a number of internal tasks that would be amenable to high-volume processing by a workforce composed of individuals with particular skills. Some of the initial tasks included the following:



Data improvement. Thousands of merchants load data for millions of products into Amazon's catalog. This can lead 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. A9's BlockView image technology aggregates millions of street-level images to create a scroll able 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, 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 aggregate value of the completed task was high enough to have a meaningful impact on the quality of Amazon's catalog or 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 automate fully.

Varied demand. The amount of work to be done varied from day to day. There might be a surge of millions of work units on a particular day. The next day might see 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.

After examining these tasks (along with many others), we realized several things. From the start there was definitely an abundance of Amazon-centric tasks to be done. After some initial discussions with potential industry partners, it also became evident that many other organizations had similar needs. There was room for an automated system to mediate between the computer and the people doing the work: accepting requests and payment information; finding and managing the workforce; tracking progress, payment, and reputation information; performing quality control; and returning results to the requesting organization. Thus, Amazon Mechanical Turk was born out of real-world requirements from Amazon and other potential users of the system.

Key challenges in building this system included scalability, reputation tracking, accountability, quality control, and flexibility.

Scalability. The system as envisioned would 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. Scalability was a necessity.

Reputation tracking. Without a system to track and control reputations of individual workers, the system would provide no framework to recognize and reward good workers. Reputation tracking was necessary to provide a long-term incentive for workers to do the best possible job.

Accountability. This 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. This would ensure that requesting organizations received work of acceptable accuracy for their money.

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

AMAZON MECHANICAL TURK

Amazon Mechanical Turk provides the interface for computers to make requests of human beings. The system provides SOAP[3] and REST[4] interfaces for creating and managing work units, also known as HITs (human

intelligence tasks). Software applications make calls to Amazon Mechanical Turk's Web service interface to request that human beings perform tasks best suited to human intelligence. These tasks include those just listed and many others, such as translating paragraphs of text from one language to another, describing a photograph, transcribing podcasts, or identifying a sound. Human beings capable of performing those tasks find, accept, and complete them, and then register the results. 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.

The Amazon 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, and stores task details and results, all behind a Web services interface.

The five key Mechanical Turk concepts are HITs, workers, qualifications, assignments, and requesters.

HITs. 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) of 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, drop-down menus, and check boxes.

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

Qualifications. These 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. Each HIT can have any number (zero or more) of associated qualifications. 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 manually graded by the requester.

Assignments. 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*. Note that any given HIT will never be assigned to the same worker more than once, regardless of the number of assignments the requester has specified for the HIT.

Plurality is an important quality control mechanism in the Amazon 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 Amazon Mechanical Turk to get a high-quality answer to this question using plurality, 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, three out of five) 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.

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

SYSTEM WORKFLOW

The requesters, qualifications, HITs, and workers all interact at the Amazon 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. The Question Language allows for the following types of elements in the questions: text; bulleted list; binary data with associated MIME type; radio button; drop-down list; check box; and multiple choice.

The requester defines the qualifications, also expressed using the Question Language, and decides on the payment (price per assignment) for the workers. Requesters can set the price to any value from 1 cent (US) on up.

Funding. The requester makes a deposit in an 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 paying for it. Requesters must also deposit an additional 10 percent over 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 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 Amazon 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[5] and online discussion forums. Workers look for HITs that are of interest to them, and for which they can meet any qualifications. The workers then endeavor to do the work, accepting HITs and returning results to the system for approval. The system tracks a multitude of statistics for each worker and each requester.

Approval. As soon as the requester has loaded a batch of HITs into the system, it will begin polling for reviewable HITs—those 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, who then performs any final checking or other quality control measures (perhaps using the plurality model) and approves each acceptable assignment.

Finalization. As soon as the requester approves assignments, the corresponding payments are released to the workers. The requester is able to aggregate results from the entire batch of HITs for use within the requester's own processing.

SYSTEM INTERFACE

As noted previously, the requesters interact with the Amazon 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 Amazon Mechanical Turk. For example, if the requester were to use the Amazon Mechanical Turk to process a number of graphical images, 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 (keyed-hash message authentication code) algorithm. HMAC is a cryptographic hashing function used to authenticate the request. By insisting on signed requests, the Amazon system is able to know with a high degree of confidence that requesters are 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 (HMAC is computationally expensive), Amazon expects 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 to process all assignments for all reviewable HITs.
- ApproveAssignment. Signifies approval of a HIT assignment performed by a worker and releases payment to the worker.
- GetHIT. Returns the data that describes the HIT.
- DisposeHIT. Destroys all memory of a HIT.
- GrantQualification. Attaches a particular type of qualification to a worker, signifying that the worker has successfully demonstrated a particular skill.
- NotifyWorker. Sends a notification message from requester to worker via the Amazon Mechanical Turk.

Potential requesters are able to use the Web services interface to connect their applications and business logic to the Amazon Mechanical Turk, making it an integral part of their business workflow.

CHECKS AND BALANCES

Integral to the success of the Mechanical Turk concept is a set of checks and balances that protect the system from intentional or accidental misuse by workers or requesters. A principal defensive tactic is the use of statistical measures. Statistics are kept per worker 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.

POSSIBLE USES FOR THE MECHANICAL TURK

As a simple, efficient way to access an Internet-scale workforce, the Amazon Mechanical Turk can be used in an almost infinite number of different ways. Here's a 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 http://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 prequalified workers. The work in each HIT consists of listening to a single podcast episode and generating a high-quality text transcript of the conversation.

Language translation. The system has been used for English-to-French and French-to-English translation, and can 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 its product catalog.

Data gathering. Several requesters are now using the Amazon 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 that characterize the image.

Web site review. Given a link to a Web site, the task is to review the site and answer a series of multiple choice questions about it.

Marketing survey. The task is to answer a series of qualifying questions and then take a marketing survey.

Sound verification. The task is to listen to a sound and verify that it matches the description.

Facial image verification. The task is to compare two facial images and decide if they depict the same person.

An important variant on most of these 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 examine the work to verify the accuracy of the translation. These two HITs actually require slightly different skill sets; many people who can read and verify the accuracy of a translation are not necessarily qualified to create a translation.

LOOKING TOWARD THE FUTURE

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

REFERENCES

1. The Mechanical Turk can be accessed at http://www.mturk.com or aws.amazon.com/mturk.
2. Standage, T. 2002. *The Turk: The Life and Times of the Famous Eighteenth-Century Chess-Playing Machine*. Walker and Company.
3. http://www.w3.org/TR/soap/.
4. http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.
5. http://www.mechturkblog.com is one of the many blogs that appeared in the days and weeks following the beta launch of the system.

LOVE IT, HATE IT? LET US KNOW

feedback@acmqueue.com or www.acmqueue.com/forums

JEFF BARR, as an evangelist for Amazon Web Services, is focused on furthering awareness of the service among software developers. Barr meets regularly with developers in the U.S. and internationally to introduce Amazon Web Services and to help them build businesses and applications with the program's services. He joined Amazon in 2002 as a senior software developer.

LUIS FELIPE CABRERA is vice president of software development for Amazon Web Services. Prior to joining Amazon Web Services, Cabrera held several positions at Microsoft and at IBM. He started his tenure at Microsoft as the architect for storage management in the Windows Base Group. He went on to become an early member in the Microsoft Web Services Architecture group, where he developed a number of distributed systems technologies for Web services. His last assignment at Microsoft was in the SQL Server team as part of the database mirroring feature. Cabrera spent 12 years at IBM where he became a member of the IBM Academy of Technology. Among the projects that he participated in are QuickSilver, Starburst, and ADSM. Cabrera earned his doctorate from the University of California at Berkeley where he was also a professor of computer science.