CrowdFlow: Integrating Machine Learning with Mechanical Turk for Speed-Cost-Quality Flexibility

Alexander J. Quinn[1], Benjamin B. Bederson[1,2,3], Tom Yeh[3], Jimmy Lin[1,2]
University of Maryland
Human-Computer Interaction Lab
Department of Computer Science[1] || iSchool[2] || Institute for Advanced Computer Studies[3]
College Park, MD 20742 USA
aq@cs.umd.edu, bederson@cs.umd.edu, tomyeh@umiacs.umd.edu, jimmylin@umd.edu

Abstract

Humans and machines have competing strengths for tasks such as natural language processing and image understanding. Whereas humans do these things naturally with potentially high accuracy, machines offer greater speed and flexibility. CrowdFlow is our toolkit for a model for blending the two in order to attain tighter control over the inherent tradeoffs in speed, cost and quality. With CrowdFlow, humans and machines work together to do a set of tasks at a user-specified point in the tradeoff space. They work symbiotically, with the humans providing training data to the machine while the machine provides first cut results to the humans to save effort in cases where the machine's answer was already correct. The CrowdFlow toolkit can be considered as a generalization of our other domain-specific efforts aimed at enabling cloud computing services using a variety of computational resources to achieve various tradeoff points.

1. Introduction

There is a large set of problems that can be solved either by human computation or machine learning. These include recognizing the faces of missing children in surveillance videos, translating documents between languages, or summarizing the opinions of blogs relating to a particular topic, and many others from the realms of natural language processing (NLP),

Generally, humans can solve these problems with higher accuracy than machines alone could do, though human efforts tend to be costly and time-consuming. Online labor markets such as Amazon Mechanical Turk (AMT) [13] give more flexibility than traditional in-person human labor. However, for most tasks, even this approach is significantly slower and more expensive than using machines. The advantage of machines is that they are inexpensive and fast, although they tend to have lower accuracy, especially when problems are difficult.

Human computation – alternately referred to as crowdsourcing [5][7], distributed human computation [16], or collective intelligence [14] - is the practice of distributing work, typically computation-like tasks, to a large number of humans via the web. Most crowdsourcing applications represent specific points in the speed-cost-quality spectrum. For example, games with a purpose (GWAPs) [22][23] typically keep human labor costs to a minimum and to use several redundant annotations to ensure the quality of the final results, all at a cost of time. MonoTrans, another current project of ours, uses monolingual human participants in collaboration with machine translation systems to translate books - resulting in translations which are between pure machine and pure expert (bilingual) human solutions [2][9]. Although MonoTrans is not a GWAP, it is similar in that it yields relatively high quality and low cost, at the expense of time. Naive uses of AMT provide faster turnaround time (and a greater range of potential applications) at the expense of cost and quality.

The principal goal of this research is to discover strategies that combine the strengths of humans and machines, in order to move past these two rigid extremes. We would like to give developers the flexibility to choose any point in the speed-cost-quality spectrum, and automatically combine human and machine resources to achieve the desired balance. To do so in a general way is an ambitious goal.

We are developing a framework for blending capabilities of humans and machines in order to enable much greater flexibility with respect to speed-cost-quality. The strategy blends the flexibility of Amazon Mechanical Turk with machine learning (or other automated methods).

To demonstrate the framework, we built the CrowdFlow toolkit, a programming library for distributing tasks between AMT and an automated system. The toolkit is currently implemented in Python and should be available for download sometime in summer 2010. The main advantages of this toolkit are that it manages the allocation of tasks to humans and machine(s) in order to achieve the desired speed, cost, or quality level.

An especially powerful element of this framework is that the humans and machines benefit from one another. Machines benefit from receiving human-created training data that helps boost their accuracy. In turn, the human workers may benefit from having the machine results as a starting point. When the machine results are correct, the human worker need only verify that it is correct, rather than doing the task from scratch.

2. Related Work

The idea of a symbiotic relationship between humans and computers goes back to the 1960s, when Licklider theorized about the future potential for such systems [11]. More recently, the idea of complementary computing has been applied to corporate telephone systems, where callers may be automatically transferred to a human operator, as opposed to the automated menu, but only when there was a need and the immediate

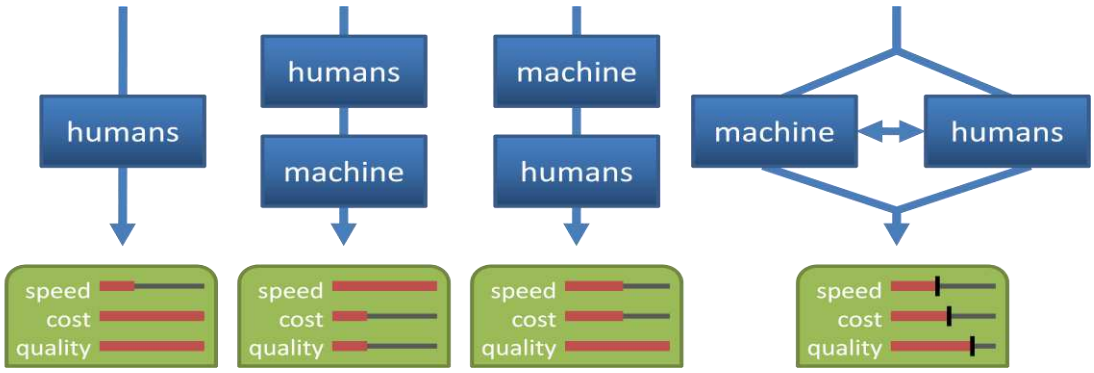

Figure 1. Tradeoffs resulting from different kinds of systems. (a) human labor only; (b) supervised learning with machine utilizing human-created training data; (c) automated machine methods with human correction; (d) CrowdFlow

availability of operators. Whereas CrowdFlow puts the power to balance the task load in the hands of the developer, this telephone system served as an automatic load-balancing mechanism [6].

The idea of a compromise between automated and human control has also been used for entertainment-oriented characters in virtual environments [4]. The AR Facade system allows participants to interact with characters that are partly AI-controlled, and partly controlled by human operators in a Wizard of Oz fashion.

Many research projects have investigated what factors influence quality in crowd sourced results. Snow found that by collecting an average of four non-expert labels on AMT for NLP tasks, they were able to approximate the level of quality ordinarily provided by expert annotators [19]. Vijayanarasimhan and Grauman used automated image processing algorithms to try to approximate the expected cost of procuring image annotations from human workers based on the visual complexity of the image [21].

Pricing online work entails the complexities of labor economics compounded by the specific constraints imposed by AMT or whatever system is being used. Thus, most research to date has been limited in scope. In the context of paid question-answer services, it has been established that offering higher incentives prices tends to increase the participation by widening the set of potential respondents [8][10]. However, in most cases, higher prices do not necessarily yield higher-quality.

Enhancing the capabilities of AMT with richer programming models was demonstrated by the TurkIt toolkit, which enables programmers to use human workers in iterative algorithms [12]. The CrowdFlow toolkit is different from TurkIt in that CrowdFlow specifically ties into machine learning systems and actively facilitates the pursuit of specific speed-cost-quality targets.

Some commercial systems such as CrowdFlow*er* (unrelated to CrowdFlow) [1] utilize well-tuned algorithms for controlling cheating and enhancing quality. CrowdFlow also supports active monitoring for cheating, although that is only a secondary purpose of this work.

3. The CrowdFlow Concept

With CrowdFlow, the user of the system will specify the desired speed-cost-quality tradeoff. The system will then allocate tasks to humans and machines in a way that will attempts to fulfill the user's specification. By estimating system performance, we can describe a tradeoff space gamut within which it is possible for the human manager to manipulate the system.

Speed may be expressed as a time limit for completing the job. Similarly, cost is the maximum the user is willing to pay to Turkers and/or for any cloud-based computational resources. (The latter is planned but not currently implemented in our toolkit.) Quality is measured relative to some satisfaction criteria the user provides. It could be a heuristic that combines multiple criteria.

Making CrowdFlow work entails several challenges. First, we must be able to estimate, within some confidence interval, the accuracy of the human workers, even if there is no ground truth. This is doubly important when you consider that without the humans' judgment, it is impossible to estimate the machine's accuracy. Also, the system must keep updating the accuracy estimates and automatically adjust the allocation of tasks as needed. Second, we need to specify a generic architecture that is both easily usable by developers of specific problems while still providing significant value. Third, we need to understand and support the wide range of human capabilities in this context. Finally, a stretch goal is to go beyond independent actions and support true interactive collaboration between human and machine participants.

Our goal is to keep CrowdFlow as general and broadly applicable as possible. Specifically, we expect it to be applicable to problems with these properties:

- Solvable by humans but at a speed or cost that makes scaling up impractical.
- Solvable by computers (including but not limited to machine learning algorithms), but with quality less than humans and less than what the situation requires.
- Divisible into small, independent subtasks.

In particular, we expect it to be especially useful for tasks with this additional optional property:

- Cannot be solved by machine learning algorithms previously trained for other problems due to domain-specific challenges.

CrowdFlow uses Turkers in two primary roles:

- **Worker** looks at the question (or other stimulus) and creates an answer.
- **Fixer** takes an answer believed to be wrong but easy to fix, and changes it to make it correct.

Which role is used depends on the specifics of the domain and the characteristics of the machine learning system. If the cognitive cost of fixing an incorrect result is low, and/or if the accuracy of the machine is relatively high, then the fixer role is preferred. It allows Turkers to benefit from the machine results, thus reducing their effort, and potentially reducing the required price and/or time required to get the work done. On the other hand, if there is an especially high cognitive load associated with looking at an incorrect result and transforming it into a correct result, and/or if most of the machines results are wrong, then it may be easier for Turkers to just do the tasks from scratch. In our toolkit, the user of the system makes the decision. Ultimately, we envision that the valve will decide automatically, or use some combination of each, if needed.

- **Validator** looks at an answer from a human or machine, and simply reports if it is correct or not.
- **Appraiser** takes an answer believed to be wrong and decides whether it would be easier to fix it or replace it with a new answer.
- **Corrector** takes an answer which may or may not be correct, and either validates that it is correct, fixes it to make it correct, or replaces it with a new correct answer.

A CrowdFlow implementation will include a fully automated system, a network of human contributors, and a valve. The **valve** is the unit that maintains accuracy estimates and calculates the optimal allocation of work between the machine and the various human roles. The conceptual model of CrowdFlow is shown in Figure 2.

In addition to the task allocation, the valve will also make use of human results to train the machine (if it is a learning system in need of further training data). Depending on the problem, it may also feed the lower quality machine results to humans (i.e. correctors, validators, etc.) as a starting point, to make the task easier than simply answering from scratch.

4. Sample applications

We built two sample applications, which demonstrate a subset of the full model. One of the applications uses human participants as Workers. The other uses Workers and Fixers. These domains were chosen because they present different challenges and thus illustrate the generality of the CrowdFlow approach. In this section, we describe the context of these applications.

4.1 Text processing

Natural language processing (NLP) has made great strides in developing algorithms and systems capable of analyzing textual language, but despite researchers' best efforts, automated solutions still do not achieve human-level performance in most tasks. We believe that text processing in general is a promising application area for CrowdFlow.

One specific application is large-scale sentiment analysis, also called opinion mining [15]. The goal is to automatically analyze how people "feel" with respect to a particular target, extracting

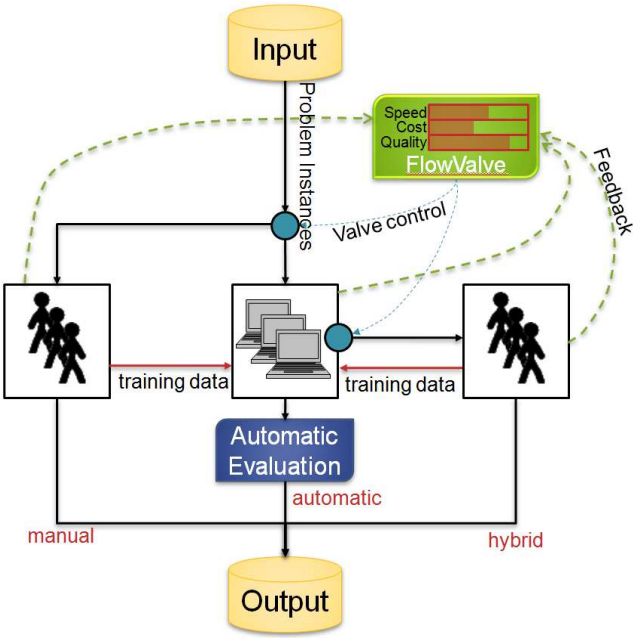

Figure 2. CrowdFlow conceptual model.

such features as polarity (positive or negative) and intensity, as well as finer-grained structure such as topic facet, rationale, or precise identification of the opinion holder. It is easy to see the applications of such technology in areas from marketing to counter-terrorism to political campaigning. Clearly, paying human editors to read and summarize all of the opinions is expensive and not scalable, while automated machines today aren't good enough. Fortunately, each review can be analyzed independently and then aggregated in a straightforward way, making this problem ideally suited to CrowdFlow.

Another important application in NLP is machine translation, or algorithms and systems for automatically translating text from one language into another. In today's increasingly multi-lingual world, the ability to transcend language barriers is important to many organizations ranging from multinational corporations to intelligence agencies. Although for certain language pairs, automated techniques can produce quite understandable output, the output is far from perfect and is often garbled. Human translators, on the other hand, are highly-trained and highly-skilled, making manual translations of large amounts of text prohibitively expensive. Finally, for the most part, translation can be parallelized given some appropriate level of granularity such as a document or individual message. Thus, translation between human languages could be performed using CrowdFlow.

4.2 Human detection

Another computing task our framework is well suited to is the retrospective analysis of surveillance videos. For example, one might need to find occurrences of a specific type of event in a long video sequence. Since events of interest often involve humans, it is necessary to first detect humans in videos before their activities can be analyzed. Human detection is a problem that can be solved by human workers more accurately than by

machines. Conversely, machines can solve it more cheaply. When human workers do this kind of work, a video is typically partitioned and distributed to several workers who can work in parallel. They may use a specialized GUI to draw bounding boxes around each human in every frame. On the other hand, to solve purely by machines, a human detector must be pre-trained using images of humans obtained from other videos before it can be applied to the current video. Our hybrid computing framework makes it possible for human and machine workers to collaborate in a seamless manner. Human images detected by the machine workers can be validated and/or fixed by human workers. Similarly, results obtained from human workers can be used as additional domain-specific training examples to retrain and improve the accuracy of the human detector. Most importantly, at all times the automated valve will attempt to maintain the right mix of human and machine work to yield the desired speed, cost, and quality characteristics.

5. Implementation

Our toolkit is implemented in Python and uses Amazon Mechanical Turk (AMT). Our goal is for the system to ultimately help developers blend answers from machine learning classifiers and Turkers to yield results within preset speed-cost-quality settings. The core CrowdFlow implementation is structured as an object-oriented Python library that can be used by programmers developing solutions to a wide variety of problems. It takes care of the valve's balancing logic as well as the low level details of dealing with the AMT web service. The CrowdFlow library exposes the following domain-independent components (implemented as Python classes):

- **Valve:** The controller responsible for allocating tasks to the machine(s) and human(s), submitting the needed HITs to AMT, and overseeing the process to ensure that the given accuracy/cost/time constraints are fulfilled.
- **AMT**: Handles the mechanics of interfacing with AMT, including submitting HITs, qualifications (i.e. for informed consent forms), etc.
- **Machine**: An abstract class representing a generic machine learning classifier with train(question,answer), evaluate(question), init(), and terminate() methods. This will be overridden at the user of the CrowdFlow library.
- **Task**: An abstract representation of a task that, when overridden, contains the specifics of each task instance. This could contain a URL, a string of text, the full data of an image, or anything else.
- **HITSpecification**: Instantiated by the user of the library to encapsulate details about the HIT, such as the price, required qualifications, and user interface that the Turkers will use to complete the HITs.

Since the CrowdFlow framework is intended to be general, we allow the user to specify the user interface using AMT's native XML schema, which provides maximum flexibility to support HITs containing any combination of text, HTML, images, video, and even dynamic content such as Flash and Java. For example, supporting the human detection tasks type requires an interface for drawing bounding boxes over a photograph. Since the exact XML will usually depend on the specifics of each task, the user of the library provides a function taking a list of Task instances as the argument.

The CrowdFlow library also accommodates a wide variety of machine algorithms by allowing the user to wrap any executable as a subclass of the Machine class. This means that the service can be distributed across multiple machines.

To use CrowdFlow to perform a set of tasks, the user of the library sets up an account with AMT and writes the following code:

- subclass of the Machine class, providing the necessary glue code to interface with the user's machine learning system
- subclass of the Task class, including logic for interpreting and scoring answers
- function to generate the XML description of the HIT user interface given a list of tasks
- main function which invokes the valve using the desired constraints and an instance of the HITSpecification class

In addition, if training data is available, CrowdFlow will use it to start training the machine. The training data, if available, is also used to establish the estimated reliability of Turkers.

5.1 Running the job

When the user's program instructs the valve to run the job, it starts by training the machine with the initial training data (if any) and establishing the initial accuracy of the machine by performing cross-validation with slices of the training data. If no training data was provided, then the user must provide an accuracy estimate for the machine.

After initializing the machine, the valve uses the given constraints to calculate an initial split, the proportion of the tasks to be allocated to humans. Then, it submits the HITs to AMT and evaluates the rest of the tasks on the machine. It polls periodically to monitor progress. Incoming results are used to further train the machine. It also notifies the user's program of progress via a callback mechanism.

Once enough HITs have been completed to fulfill the given constraints, the valve cancels any outstanding HITs. The results are sent to the calling program via a callback mechanism, and stored in a database (currently implemented with SQLite). The database is accessible via the CrowdFlow API.

5.2 Cheating

The CrowdFlow toolkit integrates a mechanism for discouraging cheating and mitigating its effects. The valve intersperses tasks with known ground truth (from the training data) along with the tasks for which answers are being sought. As it monitors the incoming results, it keeps a running accuracy estimate for each Turker. Once the Turker has completed a given number of HITs, if the Turker's accuracy is below some given threshold, all results from that Turker are invalidated and the corresponding tasks are resubmitted so they can be done by another Turker. At that point, the system sends an e-mail warning to the Turker that if the quality does not improve, they will be blocked from doing these HITs. If the Turker continues to send low-quality results, they are in fact blocked. The thresholds are supplied by the user via the API.

Currently, this requires that the user provide some initial training data. In the future, we hope to develop more sophisticated methods that can detect cheating and estimate accuracy, even without any initial training data.

A significant downside of this mechanism is that these tasks with known ground truth are included in the HITs. Thus, it incurs monetary cost and makes the entire job take longer. For this reason, in our early trials of the system, we did not use such a mechanism. In that trial, we had Turkers do 2000 tasks based on the sentiment polarity domain. Cheating was very prevalent. Presumably, the Turkers were using scripts to automatically click on answers at random.

Our analysis of that preliminary data showed a clear distinction between the cheating and the honest work. The 2,000 HITs were completed by 15 unique Turkers who did between 3 and 568 HITs each. The 9 cheaters (defined as Turkers whose answers agreed with expert judgments less than 55% of the time) accounted for 91% of the HITs. The 6 non-cheaters each had agreement with expert judgments more than 75% of the time.

We knew beforehand that cheating might occur and we knew of several tactics for mitigating it. However, we opted to do that initial exploration without their use. Since many such strategies add complication to the HITs or limit which Turkers can do them, they confound measurements of the speed at which HITs are accepted and performed. Unfortunately, with cheaters accounting for 91% of the results, that goal was unattainable.

Several alternatives exist. Delores Labs uses a similar system in conjunction with a multilevel approach whereby Turkers who do good work are given special qualifications that give them access to more HITs reserved specifically for Turkers known to do good quality work. Amazon also provides facilities for blocking Turkers whose previous work was marked as unacceptable some given percent of the time. Sheng proposed using redundant annotations to reduce noise from cheating and other sources of wrong answers [18], a strategy which is widely employed. Von Ahn built games where two participants unknown to each other answer the same question simultaneously, and they only get points if their answers agree, thus incenting participants to submit answers that would generally be considered correct [22]. Although that work was specific to games, the idea of having users validate each other's work explicitly has been proposed for use on AMT [20]. Chen used statistics to identify which work had been performed by cheaters [3].

The policy of AMT is that Requesters (those who submit HITs) may refuse payment for work deemed to be of low quality. Thus, some requesters review the work and refuse payment for low quality results. Naturally, this approach is unpopular with Turkers. The web site Turker Nation serves as a de facto blacklist of Requesters alleged to have used tactics that Turkers consider unfair. Thus, refusing payment could potentially lead to negative repercussions. Furthermore, it requires checking the work for correctness, which necessitates additional labor.

6. Analysis

We wanted to better understand CrowdFlow's ability to flexibly target a specific point in the speed-cost-quality tradeoff space. Running CrowdFlow repeatedly for every possible point in the space would consume time and money needlessly. Instead, we ran the HITs once and used simulation to explore the space.

In all cases, we used CrowdFlow's anti-cheating mechanism as above, sending a warning after 10 poor results and banning/discounting the Turkers' work after 20 poor results, with the threshold set as a program parameter. Only one Turker was banned/discounted. This may be because we restricted these HITs to only Turkers with an established approval rating of at least 80%.

To run the HITs for all of the tasks, we used CrowdFlow and set the allowable error threshold to 0% (100% accuracy). This way, the valve directed all of the tasks to Turkers. The result was a set of human-provided answers to all of the task instances for which we had ground truth data available. In all cases, the ground truth data was provided by the authors of the papers describing the respective machine learning algorithm.

6.1 Human Detection in Photographs

For the human detection domain, we used the fixer role. Turkers were given a photograph and instructed to ensure that each human was covered tightly by a bounding box. The images

Figure 3. Human detection domain user interface on Amazon Mechanical Turk.

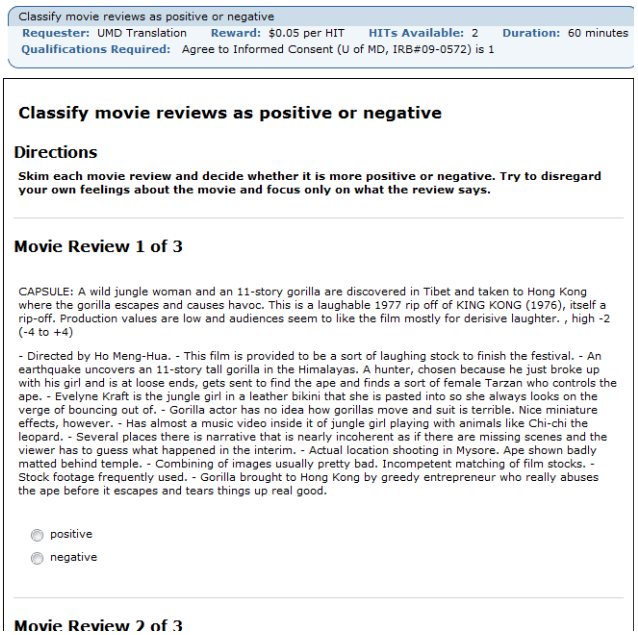

Figure 4. Sentiment polarity analysis user interface on Amazon Mechanical Turk.

were taken from the INRIA person data set, which is popular among image understanding researchers and includes ground truth for every image. A web interface (Figure 3) for drawing and editing bounding boxes was provided. It was pre-populated with results generated by the freely available DetectorPLS human detection software [17]. Some of the images contained multiple humans. Results were scored using an algorithm based on the overlap between the Turker's bounding boxes and the ground truth bounding boxes. Since we wanted to gain experience with the fixer role, we used only images for which the machine detected at least one human - about half of the total set. Among this subset of 120 images, the machine's accuracy is 60% using the same scoring algorithm.

It took 3 hours 42 minutes to get the results, including the time to rerun some HITs due to cheating. The average accuracy of the results from the Turkers was 90%. Payment was rejected for the one Turker whose work was rejected and resubmitted due to significant cheating. Thus, the total cost was $2.40.

Based on these results, we can estimate the work split required to achieve any other point in the tradeoff space.

Consider a few possible scenarios, supposing we had 1000 of these tasks and the general cost, time, and accuracy per task remained the same.

6.1.1 Time constraint

If time is the chosen constraint, then the human load L_H can be estimated dividing the time target T by 111 seconds, the observed time per hit (start to finish, including the additional time at it by anti-cheating measures).

$$L_H = \frac{T}{111 \text{ seconds}}$$

Then, the cost C and combined accuracy A are found as

$$C = \$0.02 \bullet L_H$$

$$A = 90\% \bullet L_H + 60\% \bullet (1000 - L_H)$$

For example, if we limited the time to 10 hours, we would have collected 324 judgments at 90% accuracy, with the remaining 676 tasks done by the machine at 60% accuracy. Thus, we would have an overall accuracy of 70%, costing $6.48, and taking 10 hours.

6.1.2 Cost constraint

Calculating based on a cost constraint is done similarly. We divide the cost limit C by the price per HIT to get the human load.

$$L_H = \frac{C}{\$0.02}$$

Then, the combine accuracy A is found as above and the time T as follows:

$$T = L_H \bullet 111 \text{ seconds}$$

For example, doing the same 1,000 tasks while limiting the cost to $5, we would have had 250 human judgments and 750 machine judgments, for an overall accuracy of 68%, costing $5, and taking 7 hours 42 minutes.

6.1.3 Accuracy constraint

Calculating the human load for an accuracy constraint A, We start with the equation for A.

$$A = \frac{90\% \bullet L_H + 60\% \bullet L_M}{1000} = \frac{90\% \bullet L_H + 60\% \bullet (1000 - L_H)}{1000}$$

Then, we solve for L_H.

$$L_H = 1000 \bullet \frac{A - 60\%}{90\% - 60\%}$$

Cost and time are calculated the same way as above.

For example, if we fixed the accuracy A for those same 1,000 tasks at 70%, we would have needed to give 333 tasks to Turkers, for a total accuracy of 70%, costing $6.66, and taking 10 hours 16 minutes.

6.2 Sentiment Polarity of Movie Reviews

The sentiment clarity analysis task consisted of deciding if a movie review is more positive or negative. The classification code is provided with the LingPipe natural language processing toolkit for Java (http://alias i.com/lingpipe/). It uses the algorithm and movie review data and ground truth described by Pang & Lee [15]. The data set consists of 2,000 movie reviews taken from the IMDB discussion group. The average length of a review is 612 words. The Turkers agreed to an informed consent form via AMT's qualification mechanism. Each HIT (Figure) presents three movie reviews at a time and uses radio buttons to let the Turker enter a judgment of "positive" or "negative". Turkers were paid US$0.05 per HIT, with each HIT containing 3 movie reviews

Over a period of 8 hours 7 minutes, the Turkers judged a total of 1083 movie reviews (361 HITs) for a total cost of $54.35. No Turker was discounted due to cheating. The overall human accuracy was 91.0%. In comparison, the machine classifier can

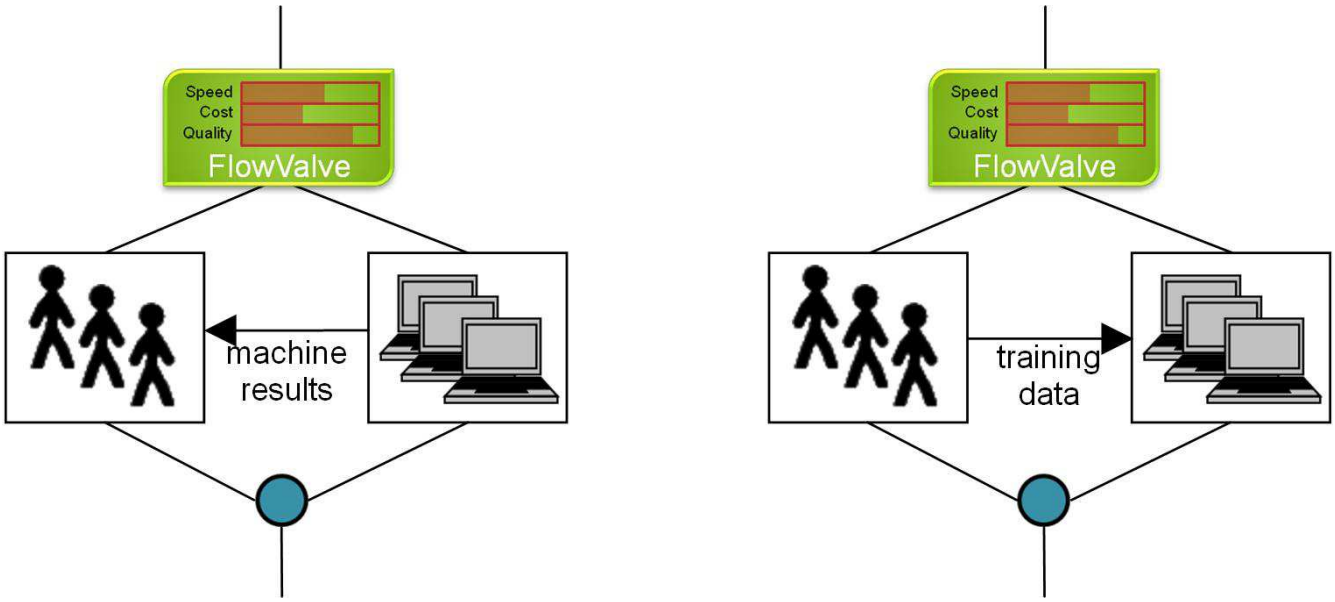

Figure 5. The human detection example (left) illustrates one direction of feedback, with computers helping Turkers. The sentiment analysis domain (right) illustrates the other direction, with the Turkers providing training data.

do the same tasks with only 83.5% accuracy. A similar extrapolation could be done with this domain.

7. Discussion and future work

This toolkit is the first step in this line of research. It does not use all of the roles and possibilities we envision for CrowdFlow, nor does it resolve all of the difficult questions that would be necessary to fully realize the vision. However, the flexibility of CrowdFlow to provide a spectrum of speed-cost-quality tradeoffs is evident from the simulations.

The experiments we have run to date exercise the two directions of benefit, humans providing training data to machines and machines providing imperfect results to save the humans work. However, as Figure 5 illustrates, neither of these domains exercise both direction at the same time. Ultimately, that will be the goal.

Looking forward, we would like to know what factors affect the speed at which Turkers will accept HITs, and to what extent the current AMT design is influencing the economics. This would be useful for maximizing the benefits of AMT, but also to inform the design of any future online on-demand labor market place.

Regarding the roles we described above, we plan to do more evaluations to understand how those can be used to optimize the output with CrowdFlow. To do that, we will need to know how cognitively expensive it is for a human to judge whether a machine generated answer is good enough to correct versus clearing the answer and starting over. Perhaps research methods from economics could guide such an exportation by estimating the perceived abstract cost of performing different kinds of tasks. Similarly, we would like to be able to model the best price to pay for a given HIT based on the work involved.

Even with the best models, practical aspects of the AMT site design might affect the ability of CrowdFlow to deliver the promised results. Thus, we would like to better understand how the performance of CrowdFlow in the wild differs from our predictions based on models.

8. References

[1] Barrett, V. Dolores Labs Vets Web Sites On The Cheap - Forbes.com. *Forbes Magazine*, 2009. http://www.forbes.com/forbes/2009/0330/048-you-know-it.html.

[2] Bederson, B. B., Hu, C., & Resnik, P. (2010) Translation by Interactive Collaboration between Monolingual Users, *Proceedings of Graphics Interface* (GI 2010), (in press).

[3] Chen, K., Wu, C., Chang, Y., and Lei, C. A crowdsourceable QoE evaluation framework for multimedia content. *Proceedings of the seventeen ACM international conference on Multimedia*, ACM (2009), 491-500.

[4] Dow, S. Mehta, M., MacIntyre, B., Mateas, M.. Eliza meets the Wizard-of-Oz: Blending Machine and Human Control of Embodied Characters. In CHI 2010, 2010.

[5] Hoffmann, L. *Crowd control*. Communications of the ACM 52, 3 (2009), 16-17.

[6] Horvitz, E., Paek, T.. Complementary computing: policies for transferring callers from dialog systems to human receptionists, User Modeling and User-Adapted Interaction, 2007.

[7] Howe, Jeff, *The Rise of Crowdsourcing*. Wired Magazine, Issue 14.06, June 2006.

[8] Hsieh, G., Kraut, R. E., Hudson, S. E.. Why Pay?: Exploring How Financial Incentives are Used for Question & Answer. In CHI 2010.

[9] Hu,, C. (2009) Collaborative translation by monolingual users, In *Proceedings of the 27th international conference extended abstracts on Human factors in computing systems* (CHI 2009), pp. 3105-3108, Boston, MA, USA: ACM, 2009.

[10] Jeon GYJ, Kim YM, Chen Y. Re-examining price as a predictor of answer quality in an online Q&A site. In CHI 2010.

[11] Licklider, J. C. R. Man-computer symbiosis. In *IRE transactions on human factors in electronics*, 1960.

[12] Little, G., Chilton, L. B., Goldman, M., and Miller, R. C. 2009. TurkIt: tools for iterative tasks on mechanical Turk. In Proceedings of the ACM SIGKDD Workshop on Human Computation (Paris, France, June 28 - 28, 2009). P. Bennett, R. Chandrasekar, M. Chickering, P. Ipeirotis, E. Law, A. Mityagin, F. Provost, and L. von Ahn, Eds. HCOMP '09. ACM, New York, NY, 29-30. DOI=http://doi.acm.org/10.1145/1600150.1600159

[13] Mechanical Turk. http://mturk.com.

[14] Malone, T.W., Laubacher, R., and Dellarocas, C. Harnessing Crowds: Mapping the Genome of Collective Intelligence. (February 3, 2009). MIT Sloan Research Paper No. 4732-09.

[15] Pang, B., Lee, L., and Vaithyanathan, S. 2002. Thumbs up?: sentiment classification using machine learning techniques. In Proceedings of the Acl-02 Conference on Empirical Methods in Natural Language Processing - Volume 10 Annual Meeting of the ACL. Association for Computational Linguistics, Morristown, NJ, 79-86. DOI= http://dx.doi.org/10.3115/1118693.1118704

[16] Quinn, A. J., Bederson, B. (2009) A Taxonomy Of Distributed Human Computation. Technical Report, University of Maryland, Human-Computer Interaction Lab, HCIL-2009-23.

[17] Schwartz, W. R., Kembhavi, A., Harwood, D., Davis, L. S.. Human Detection Using Partial Least Squares Analysis. In *Proceedings of the ICCV*. Kyoto, Japan, 2009

[18] Sheng, V. S., Provost, F., and Ipeirotis, P. G. 2008. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceeding of the 14th ACM SIGKDD international Conference on Knowledge Discovery and Data Mining* (Las Vegas, Nevada, USA, August 24 - 27, 2008). KDD '08. ACM, New York, NY, 614-622.

[19] Snow, R., O'Connor, B., Jurafsky, D., and Ng, A.Y. Cheap and fast---but is it good?: evaluating non-expert annotations for natural language tasks. *Proceedings of the Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics* (EMNLP 2008), 254-263.

[20] Sorokin, A., Forsyth, D. Utility Data annotation with Amazon Mechanical Turk. In *InterNet08*, pages 1-8, 2008.

[21] Vijayanarasimhan, S. and Grauman, K. What's It Going to Cost You?: Predicting Effort vs. Informativeness for Multi-Label Image Annotations. CVPR, pp.2262-2269, In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009.

[22] von Ahn, L. and Dabbish, L. 2004. Labeling images with a computer game. In Proceedings of the SIGCHI Co,nference on Human Factors in Computing Systems (Vienna, Austria, April 24 – 29, 2004). CHI '04. ACM, New York, NY, 319-326. DOI= http://doi.acm.org/10.1145/985692.985733

[23] von Ahn, L. v. 2006. Games with a Purpose. Computer 39, 6 (Jun. 2006), 92-94. DOI=http://dx.doi.org/10.1109/MC.2006.196

[24] von Ahn, L., Graham, M., I., Dabbish, L., Kitchin, D., Blum, L. Solving Hard AI Problems With Computer Games. CMU Technical Report CMU-CS-02-191, Nov. 2002