

A Comprehensive Human Computation Framework – With Application to Image Labeling

Yang Yang¹, Bin B. Zhu², Rui Guo³, Linjun Yang², Shipeng Li², Nenghai Yu¹

¹ University of Science and Technology of China, Hefei, Anhui, 230027, China

² Microsoft Research Asia, Beijing, 100080, China

³ Beihang University, Beijing, 100080, China

¹ {wdscxsj, ynh}@ustc.edu.cn

² {binzhu, linjuny, spli}@microsoft.com

³ imguorui@gmail.com

ABSTRACT

Image and video labeling is important for computers to understand images and videos and for image and video search. Manual labeling is tedious and costly. Automatically image and video labeling is yet a dream. In this paper, we adopt a Web 2.0 approach to labeling images and videos efficiently: Internet users around the world are mobilized to apply their “common sense” to solve problems that are hard for today’s computers, such as labeling images and videos. We first propose a general human computation framework that binds problem providers, Web sites, and Internet users together to solve large-scale common sense problems efficiently and economically. The framework addresses the technical challenges such as preventing a malicious party from attacking others, removing answers from bots, and distilling human answers to produce high-quality solutions to the problems. The framework is then applied to labeling images. Three incremental refinement stages are applied. The first stage collects candidate labels of objects in an image. The second stage refines the candidate labels using multiple choices. Synonymic labels are also correlated in this stage. To prevent bots and lazy humans from selecting all the choices, trap labels are generated automatically and intermixed with the candidate labels. Semantic distance is used to ensure that the selected trap labels would be different enough from the candidate labels so that no human users would mistakenly select the trap labels. The last stage is to ask users to locate an object given a label from a segmented image. The experimental results are also reported in this paper. They indicate that our proposed schemes can successfully remove spurious answers from bots and distill human answers to produce high-quality image labels.¹

Categories and Subject Descriptors

I.2.6 [Learning]: Knowledge Acquisition. H.3.3 [Information Search and Retrieval]: Retrieval models. H.5.3 [Group and Organization Interfaces]: Collaborative computing.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

MM’08, October 26–31, 2008, Vancouver, British Columbia, Canada.
Copyright 2008 ACM 978-1-60558-303-7/08/10...\$5.00.

General Terms

Design, Security, Human Factors

Keywords

Distributed knowledge acquisition, human computation, image labeling, common sense problems, HumanSense.

1. INTRODUCTION

Given an image or video, what does it contain? What does an object do in the video? How are objects related to each other? Such questions are hard to answer for even today’s most powerful computers, but can be easily answered by humans. There exist many similar tasks in our lives. For example, characters scanned from a book that cannot be recognized by Optical Character Recognition (OCR) can be easily recognized by humans. A person can be instantly identified from a group of people by humans, but the same task is hard for computers. This type of problem that requires only human’s common sense knowledge to solve is referred to as a *common sense problem* in this paper.

As early as in the 1980s the importance of common sense knowledge for computers was recognized [1][2]. Labeling images or videos is a common sense problem. The labeling results are very useful in helping computers understand images or videos, and in improving their image or video search results. Humans are typically hired to label images or videos nowadays. It is a labor-intensive work. It would be impractical or simply infeasible to hire enough number of dedicated people to label images or videos in a large scale, such as those crawled from the Internet. Labeling crawled images and videos would dramatically improve image and video search accuracy and relevancy. A new approach is needed to meet the demand of labeling large-scale images and videos. The Internet and Web 2.0 can be exploited to achieve the goal. There are hundreds of millions of Internet users, and hundreds of thousands of small Web sites. These Internet users and Web sites can be mobilized to help solve such common sense problems with an appropriate system, making the Internet very suitable for solving large-scale common sense problems.

There are several proposals that exploit the power of the Internet to solve common sense problems. Open Mind [3] is a world-wide collaborative effort initiated by the MIT Media Lab to collect common sense knowledge from people to develop intelligent

¹ This work was done when Yang Yang was an intern at Microsoft Research Asia.

software. Regular Internet users are recruited to answer simple questions on its Web sites, and the collected answers are processed by machine learning algorithms. Open Mind relies solely on contributions from online volunteers. It costs very little, mainly for the maintenance of the specific Web sites. But counting on online volunteers would make Open Mind unable to scale up to meet the demand of labeling crawled images and videos. Open Mind has no mechanism to prevent spurious answers from bots. The results may be tainted and become unreliable and inaccurate when bots are employed to answer questions in Open Mind.

By exploiting the fact that many Internet users are willing to play entertaining online games, von Ahn et al. proposed to use *human algorithm games* to harness their time and energy to solve common sense problems [4][5][6][7][8][9]. The idea is to convert plain, tedious questions into vivid, intriguing online games. Web users, viewed as distributed processors connected by the Internet, are attracted to play these games, and valuable common sense knowledge comes as a by-product. For example, the ESP game [4] displays a same image to a pair of concurrent players randomly selected and asks them to guess what the partner would think about the image. Each player enters possible descriptions of the image as quickly as possible, and once both players give a same description, they “win” on this image and move on to play with the next image. This method is as cost efficient as Open Mind. In addition, it has a strong probabilistic guarantee that the data collected has a high-quality. The reason is twofold. First, people are driven by the desire to win the game, and the only way to win on an image is that both parties have to give a same description. Since the two players are selected randomly, it is hard to win the game unless both parties give a valid answer. This would effectively screen out incorrect answers or spurious answers by bots. Google brought online a commercialized version of the ESP game, the Google Image Labeler [10], in 2006.

CAPTCHA [11] is a computer generated challenge-response test to distinguish humans from computers using a common sense problem. reCAPTCHA [12] is a novel CAPTCHA that produces valuable common sense knowledge to improve the OCR quality in digitizing books for the Internet Archive [13]. It works by combining two words that OCR cannot read into a challenge. One word is already identified and serves as a conventional CAPTCHA, while the other word is not identified yet. If a user recognizes the identified word, the answer to the unidentified word is assumed to be correct, and is collected to identify the unidentified word. The OCR result is therefore improved.

Using games or CAPTCHA to collect common sense knowledge, as described above, are innovative ideas. They are very suitable for large scale applications such as labeling images for a portal of user-uploaded images and the Internet Archive project [13]. It’s estimated that “5,000 people playing the ESP game 24 hours a day would label all images on Google (425,000,000 images) in 31 days” [4]. Although very useful, they are restricted to collecting certain types of common knowledge. As admitted by the author of human algorithm games, turning a common sense problem into a human algorithm game is “nontrivial” [9] because games are required to be enjoyable. It is also a challenge to preserve user’s enthusiasm in playing a human algorithm game either. Like human algorithm games, converting a common sense problem into a CAPTCHA is not trivial, either. In addition, a human algorithm game typically requires two Internet users to play live

against each other². This would raise the bar for the game hosting Web sites, requiring them to have enough number of concurrent online users. The manpower of smaller Web sites is thus wasted.

A computational process that involves humans in performing certain steps is called *human-based computation* [14], or simply *human computation*³. It leverages differences in abilities and costs between humans and computers to achieve symbiotic human-computer interaction. In this paper, we propose a framework that employs human computation to solve general common sense problems efficiently. The framework supports a range of viable business models, and can scale up to meet the demand of a large amount of common sense problems. A hosting Web site can be either large with heavy traffic or small with limited visitors so that every Internet user can contribute. Our system can be deployed at the entrance to Web-based services such as Web email services, software downloading services, etc. It also supports a profit sharing ecosystem similar to Google’s AdSense [15] that motivates Internet users to offer their solutions to our problems in exchange for shared profit.

The framework proposed in this paper aims to address the problems associated with the existing approaches mentioned above. With our framework, there is no need to convert a common sense problem into a CAPTCHA or an entertaining game. Such a conversion is nontrivial and challenging, and needs to be designed case by case. Many human computation examples may not be able to convert to human algorithm games or CAPTCHA but can easily fit into our framework. These examples include human-based genetic algorithm [16] which utilizes humans for evaluation and three types of innovation (contributing new content, mutation, and recombination), and interactive evolutionary computation that uses humans for evaluation of computer graphics [17] or of video labeling by active-learning engines [18]. Therefore a much broader spectrum of common sense problems can be solved with our framework. Attracting enough human contributors is a key factor in using human computation to solve large scale common sense problems. As we have mentioned, preserving user’s enthusiasm is a great challenge for a human algorithm game. With the application scenarios to be described in Section 2.3, our framework can maintain a constant or even increasing number of human contributors. Like Google’s AdSense, Web sites with small traffic can be employed in our framework to reach more human contributors.

The framework brings several technical challenges not seen in the aforementioned human computation schemes, including:

1. How to design the system to support solving general common sense problems without knowing what questions are asked? In other words, the system should be problem-agnostic.

² In [9] single player games are also described. They are variations of the two-player games by using a bot to emulate a human player. They are irrelevant to the problems addressed in this paper since we have assumed that the problems cannot be solved by computers. If a server-side bot can emulate a human player, there is no need to collect more answers from humans since the collected information is enough.

³ Human computation amusingly brings computing back to its original meaning. In 1940s, a computer, i.e., a person, used a calculator to contribute to a larger problem such as computing missile trajectories.

2. How to screen out answers from bots? In order to support solving general common sense problems and to enable a single Internet user to contribute (i.e., any Web sites, large or small, can solicit their users to help solve common sense problems), our system may not be able to tie a question to a CAPTCHA or to use other means such as two users playing against each other to tell whether an answer is from a bot or human. This requires that our system should have a mechanism to screen out spurious answers from bots.
3. How to distill the answers collected from individual human users to ensure the quality of the solutions our system produces?
4. How to prevent malicious players of the system from attacking others or contaminating the solutions produced by the system? How to prevent users from making money without contributing any answers?

These technical challenges are addressed in the system we propose in this paper. Our system uses `<iframe>`'s isolation guaranteed by the Same Origin Policy (SOP) [19] to prevent a malicious problem provider from attacking participating Web sites, and uses fragment identifiers for secure client-side cross-domain communications. Random problem ID is used to prevent attacks launched by colluding participating Web sites and users. We propose a statistical method to effectively remove spurious answers from bots, and a majority voting scheme to combining individual human answers to produce high-quality solutions to common sense problems.

After presenting our general human computation framework, we apply it to image labeling, which is a typical common sense problem. We propose three incremental refinement stages to produce high-quality image labels. The first stage collects candidate labels from users. The second stage refines the candidate labels and identifies synonymic labels. Experimental results will also be reported. They will show the effectiveness of our schemes in removing spurious answers from bots and distilling individual human answers to achieve high-quality solutions.

The remaining of the paper is organized as follows. The threat model and an ecosystem based on our proposed system are presented in Section 2. Our human computation framework is described in detail in Section 3. The framework applied to image labeling is described in Section 4. Experimental results are reported in Section 5. The paper concludes in Section 6.

2. THREAT MODEL AND ECOSYSTEM

2.1 Players

Our human computation system is called *HumanSense*. There are four players connected by the Internet in it, as shown in Figure 1: problem providers, HumanSense server, participating Web sites, and Internet users. The role of each player is explained as follows:

- Problem providers. A problem provider provides common sense problems that need to be solved with HumanSense. Resulting answers from the HumanSense server are also sent back to the problem provider. Therefore a problem provider is also called a solution seeker. A problem provider typically offers money, souvenirs, free services, or anything else valuable to compensate the other parties of the system for their contribution to solving the problems.

- HumanSense server. It selects problems and sends to participating Web sites, fetches Internet users' answers, analyzes them to produce solutions to the problems, and sends these answers to the problem provider. It is the major player of HumanSense, and will be described in detail in the subsequent sections.
- Participating Web sites. A participating Web site receives a problem each time from the HumanSense server, and presents it to Internet users for their answers. It may offer certain services such as Web email or software downloading service to Internet users in exchange for their answers to the problem.
- Internet users. They contribute their answers to the problem in exchange for money, rewards, services, or simply fun. An Internet user might be a bot.

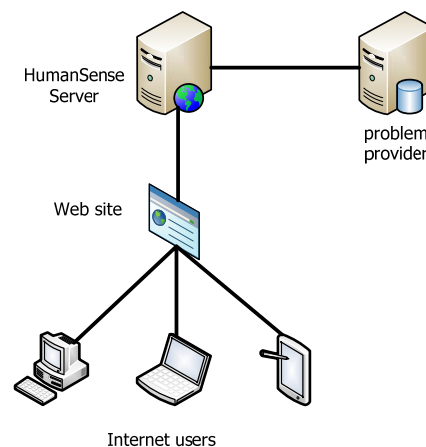


Figure 1: Four players in HumanSense.

2.2 Threat Model and Assumptions

In HumanSense, we assume that only the HumanSense server is trusted. A problem provider may be anyone who seeks solutions to her common sense problems through HumanSense. A problem provider may be malicious, targeting at attacking participating Web sites or tricking Internet users into clicking a malicious link to go to a malicious Web site or download a malicious file. An Internet user is untrusted too. It may be a bot that provides arbitrary answers to the problems it encounters, or a malicious human who want to gain disproportionate compensation. A participating Web site may also be malicious. It may collude with other participating Web sites or Internet users to make more money disproportional to their contributions to the problem solutions.

We further assume that most Internet users are well-behaved humans who want to contribute in exchange for compensation or free services. A human may sometimes be careless enough to provide incorrect answers. Therefore human users collectively provide correct solutions to common sense problems. This is the foundation of our framework to be used to solve common sense problems.

2.3 Ecosystem

As we have mentioned previously, HumanSense supports a range of viable business models. We are particularly interested in two application sceneries. Both would be able to attract a large number of human contributors to solve common sense problems, an essential factor to make an application of HumanSense successful. The first scenario is to place our human computation at an entrance to free services provided by a large Web site with a steady flow of visitors, such as Windows Live Hotmail which provides free Web email services, or www.download.com which provides free software to download. These Web sites are compensated with the fees paid by problem providers for hosting common sense problems. Just like online advertisement, our system would encourage a Web site to improve its services to attract more users since the more the Internet users, the more the Web site can make from HumanSense. When a user wants to enter her Hotmail account or download free software, she is asked to provide answers to a common sense problem before receiving the service she asks for. Scores or awards may be used to encourage people to provide high-quality answers.

It is also possible to combine a common sense problem with a CAPTCHA challenge for certain types of common sense problems, e.g., reCAPTCHA. In this case, a user answers the CAPTCHA and contributes an answer to the common sense problem. If the answer to the CAPTCHA is correct, the user is assumed to be human, and is allowed to receive the desired service, and the answer to the common sense problem is recorded by the HumanSense server. Since the CAPTCHA has already screened out bots from entering answers to our problems, all the collected answers are from humans. There is no further need to try to apply any mechanism to remove answers from bots in this case.

Another application scenario is distributed human computation in which HumanSense pushes common sense problems to the Web pages of participating Web sites, large or small. Like AdSense [15], HumanSense collects common sense problems from problem providers who seek solutions through HumanSense, and pushes them to participating Web pages. Relevancy or other criteria can be used to select a common sense problem to be presented to Internet users in a certain Web page. HumanSense would charge the problem providers for the solutions HumanSense provides. The profit would be divided among the participating Web sites and the Internet users who provide correct answers. All the answers provided by a user are recorded by HumanSense, but only the answers contributing to the final solution generated by HumanSense would be credited for compensation of contributing answers. HumanSense brings benefits to all the parties involved. Participating Web sites makes money by hosting common sense problems to reach their visitors. They help HumanSense reach a huge number of Internet users, visitors of both large and small Web sites, in a short time. Internet users make money by contributing correct answers in an easy and convenient manner. This is a key difference between HumanSense and the human algorithm games mentioned previously. We don't rely on Internet user's enthusiasm in playing games – they may get bored after playing for a while, and it is a nontrivial task to convert a common sense problem into a game. Instead, HumanSense presents a common sense problem as it is to users and allures them to contribute by compensating their correct answers with money. Solution seekers also benefit from HumanSense since they find a cheap and quick method to solve large-scale common sense problems such as labeling images and

videos crawled from the Internet for search. HumanSense makes money by operating the system.

3. OUR HUMAN COMPUTATION FRAMEWORK

3.1 Problem Representation

In HumanSense, a problem is represented by the problem resource files (such as images, videos, etc.) plus a manifest file which contains the following information about the problem:

- **problem:** Required. This is the root element, indicating the file is a problem-describing manifest.
- **id:** Required. Unique global ID of the problem.
- **resources:** Required. This part indicates resource files associated with the problem. Each resource file is represented as a nested element, such as **image**, **video**, etc.
- **priority:** Optional. An integer value indicating how often the problem shall be presented to the users. The default value is 1.
- **value:** Optional. An integer value indicating how much value the problem is worth. Combined with other factors, this value is used to calculate a “score” or monetary award for a correct answer to the current problem. The default value is 1.
- **type:** Required. The type of the problem. This part indicates to the HumanSense server how to process the answers to the problem. The proposed framework allows a plug-in to support a new method associated with a new type to process the collected answers.

```
<problem>
  <id>13089</id>
  <resources>
    <image>13089.jpg</image>
  </resources>
  <priority>3</priority>
  <value>2</value>
  <type>ImageLabeling</type>
  <stage>MultipleChoices</stage>
  <labels>
    <label>tiger</label>
    <label>claw</label>
    <label>tail</label>
  </labels>
</problem>
```

Figure 2: An example of manifest file for an image in the second stage of image labeling.

The manifest file may also contain problem-type specific elements. An example of manifest file is shown in Figure 2. It is a manifest file for an image in the second stage for image labeling described in Section 4. Three labels, tiger, claw, and tail, are contained in the manifest. They are displayed as multiple choices for users to select appropriate ones from in the second stage of image labeling. They are problem-type specific elements. A manifest file is expressed in

XML for easy extension. Note that some items such as `priority` in a manifest file may be mutable during the computation.

3.2 Constituent Modules and Interactions

Figure 3 shows major constituent modules of HumanSense. The whole process consists of the following steps:

1. An Internet user visits a participating Web site for a service such as accessing Web emails. The Web site requests a common sense problem from the HumanSense server. The HumanSense server selects an appropriate problem from the problem database. To prevent malicious Web site from tracking or logging the (problem, answer) pair, the HumanSense server generates a random ID unique to the current session, maps it to the ID of the selected problem, and sends it to the Web site. The HumanSense server maintains the association of the random ID and the real problem ID for the current session.

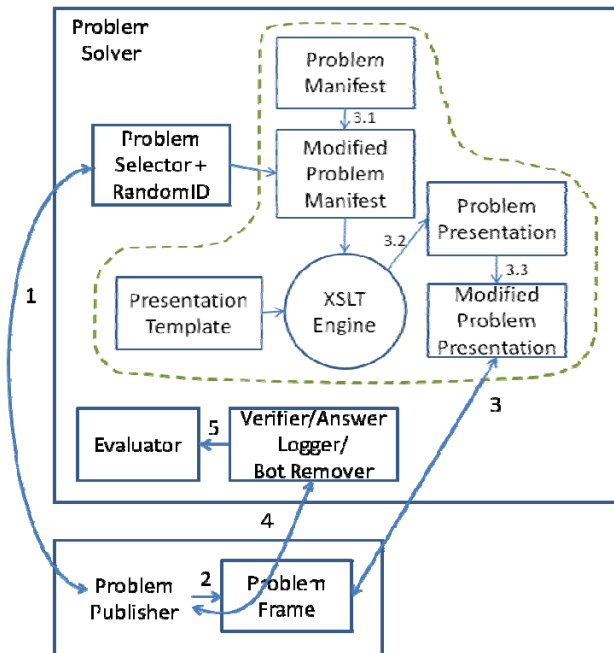


Figure 3: Constituent modules.

2. The Web site creates an `<iframe>` to aggregate the problem presentation into the participating Web page. The `<iframe>` will load the problem with pre-fetched `randomId` by using URL like the following: `http://HumanSenseServer/problem?id=randomId`. A malicious problem provider may launch phishing attacks against a user by tricking the user to believe that the problem frame is from the Web site, and inputs private data such as a password into the frame, resulting in the private data being secretly sent back to the malicious problem provider through embedded scripts. To prevent such phishing attacks, the Web site should wrap the problem frame in a different display style to differentiate the problem frame from the Web site's content, and also add a clear note on top of the problem frame to warn users that the frame is used to answer common sense problems rather than input private data.

3. The HumanSense server generates the problem page and sends to the problem frame.

3.1 The problem manifest file is modified to remove the information that is not needed by Internet users, such as the problem ID, priority, value, etc. All the direct resource references (such as `13089.jpg` in Figure 2) are replaced with some URL like `http://HumanSenseServer/resource?id=randomId&index=0`, where the index parameter indicates the order of the resource in the problem manifest that the URL refers to. Since the HumanSense server maintains the association of the random ID with the actual problem ID, correct resources can be retrieved by the HumanSense server. Web sites or Internet users, on the other hand, cannot tell from the resources or the random ID if the problem has been answered or not. Therefore they cannot launch an attack to repeat an answer to the same problem in order to let the answer be in the solution produced by HumanSense. Note that users would be awarded for contributing correct answers.

3.2 A problem provider is allowed to choose a presentation template for each problem it provides. The presentation template can be selected from a library of templates or created if no suitable template is available. A new template will be saved in the template library for future selection. A presentation template is defined in XSLT or CSS, and is applied to the modified problem manifest file to convert the modified manifest into a Web page containing normal HTML and JavaScript to provide UI for presenting the problem and inputting answers, as well as a special JavaScript function called `"$collectAnswer"` to designate how to collect answers from the generated UI. Since the problem is represented in an `<iframe>` whose domain is different from the Web site, the Same Origin Policy (SOP) [19] guarantees that the content in the problem frame would not introduce any cross-site scripting (XSS) attacks to the Web site.

3.3 The modified problem representation page is sent back as the content of the problem `<iframe>` in the Web page. The page is modified for appending scripts to support cross-domain communication, which will be used to transmit a token from the problem frame to the Web page of the participating Web site.

4. If a common sense problem is used as a CAPTCHA to tell humans from bots, the Web site needs to know whether the problem is answered by a human or by a bot in order to make a decision on navigating to the logic of the next step or not. In this case, a CAPTCHA is added to a common sense problem by the HumanSense server, such as the case of reCAPTCHA in which a CAPTCHA and a problem are displayed side by side. The HumanSense server should select a problem with known answers from a random problem provider as a CAPTCHA. This would prevent a problem provider from using the known answers to its problems to solve the CAPTCHA automatically. When a correct answer to the CAPTCHA is received, the HumanSense server would send a token to the problem frame. Cross-

domain communications between the problem frame and the frame of the Web site are needed in generating the token and passing the CAPTCHA result included in the token to the Web site. The current Web standards don't support cross-domain communications. The next Web standards, HTML 5 [20], as well as other proposals [21][22] would support secure client-side cross-domain communications. These proposals require modifications on a current browser. In order to allow client-side cross-domain communications without any modifications to the current browsers, the subspace scheme proposed in [23] is adopted in our system. To generate the token, a nonce is first generated by the server of the Web site, and passed to the problem frame via fragment identifiers for the current problem. The nonce is then passed to the HumanSense server with the answers to the CAPTCHA and the common sense problem. If the answer to the CAPTCHA is correct, the HumanSense server uses its private key to encrypt the nonce together with other information in the token. The token is passed to the problem frame, and then to the Web site's page via fragment identifiers. The Web site's page will then submit the token to its server. Upon receiving the token, the Web server uses the public key of the HumanSense server to decrypt the token and verifies the decrypted nonce matches the nonce it generates for the problem. If they match, it concludes that the token was indeed from the HumanSense server for the specific CAPTCHA and the user is human. The logic of the next step is then executed.

The two modules of the HumanSense server, one that removes spurious answers by bots and the other that distills individual human answers, will be described in Sections 3.4 and 3.5, respectively.

3.3 Selection of a Common Sense Problem

A problem is selected to be presented at a participating Web page based on the following criteria: problem's type, value, and priority. By default, a participating Web page is assumed to be able to present any type of problems for users to answer. A participating Web site can optionally ask the HumanSense server to send a specific type of problems to present to its visitors. Problems of default value are typically presented on a participating Web page. Internet users can ask for problems of higher or lower values. Correct answers to problems of higher values would earn more money, higher scores, or better services. As a result, users may be motivated to choose problems of higher values to answer. We note here that a problem of higher value means that a user usually needs to spend more time or effort to answer. More compensation should be offered for correct answers to problems of higher values. Problems of higher priority are selected to be pushed to participating Web pages more frequently. Solutions to problems of higher values or priorities would cost more to the problem provider.

In our system, only the solution provider and Internet users know if an answer is correct or not. Problems and answers are agnostic to both the HumanSense server and Web sites except the types of the problems and answers. The HumanSense server uses statistical methods to analyze the collected answers from users and produces problem solutions to send to the problem providers. To avoid producing incorrect answers due to corrupted data collected from

users, problems should be selected randomly so that it would be unlikely for any fixed group of users to receive the same problem more than once. This mechanism targets mainly at collusion of bots, as we have assumed in Section 2.2 that human users are benign. This can be easily achieved with a large pool of common sense problems that our system is designed for.

When the pool of problems is small, it is inevitable that some problems might be repeated despite the fact that a problem is randomly selected. HumanSense has built-in security mechanisms to protect against colluding attacks by bots and Web sites unless content of a displayed problem is analyzed to extract its features which are compared with those of previously displayed problems to detect if two problems are the same or not, as in the image labeling case described in Section 4. Note that from Section 3.2, the Web content sent to a participating Web site does not contain any information about the problem. Web sites or Internet users cannot tell if two problems they saw are the same or two from the Web content the problem <iframe> receives. In addition, multiple copies of a problem can be generated, with each copy being slightly different. For example, for an image to be labeled in the system described in Section 4, the content of each copy is slightly modified without changing its semantic meaning. Hence the hash values of these copies are different, and it is impossible to use hash values to identify if two problems are the same or not. The only way to find out if two images are the same or not is to use content analysis.

3.4 Removing Spurious Answers from Bots

When CAPTCHA is not used with common sense problems, the collected answers from users may contain spurious answers provided by bots. These spurious answers must be removed from the collected answers to ensure the quality of the solutions produced by HumanSense. Since the common sense problems cannot be answered by computers (otherwise there is no need to use human computation to find answers), and it is highly unlikely that a fixed group of users would be able to see a same problem appearing more than once, we can assume that an answer provided by a bot is random with uniform distribution, and each answer from bots is assumed to be independently and identically distributed (i.i.d.). Therefore the answers from bots can be modeled as an i.i.d. uniform distribution.

Suppose the i -th answer to a problem P provided by users is a_i . Let DA be the set of distinct answers collected for problem P , and the j -th member of DA is denoted as A_j . The frequency C_{A_j} that answer A_j appears in the collected answers for problem P is then $C_{A_j} = \sum_i b_{i,j}$, where

$$b_{i,j} = \begin{cases} 1, & \text{if } a_i = A_j; \\ 0, & \text{otherwise.} \end{cases}$$

C_{A_j} consists of two parts: contribution from humans $C_{A_j}^h$ and contribution from bots $C_{A_j}^b$: $C_{A_j} = C_{A_j}^h + C_{A_j}^b$. Let's consider the distribution of $C_{A_j}^b$. Suppose that the total number of answers and the number of distinct answers are T and N , respectively. Note

that $T \geq N$. It is easy to deduce the average and standard deviation of $C_{A_j}^b$ for i.i.d. uniform distribution:

$$\langle C_{A_j}^b \rangle = T / N, \quad (1)$$

$$\sigma_{C_{A_j}^b} = \sqrt{\frac{T}{N} - \frac{T^2}{N^2}} \approx \sqrt{\frac{T}{N}} \quad (2)$$

The following recursive procedure is applied to remove spurious answers from bots when the HumanSense server has collected enough number of answers to problem P :

1. If N is smaller than a threshold and T/N is larger than a threshold, terminates; otherwise initialize the set of answers from bots, S_{bot} , to be the set of all the answers collected for problem P .
2. Calculate the average and standard deviation of the answers provided by users in S_{bot} by using Eqs. (1) and (2).
3. Any frequency $C_{A_j} > k \sigma_{C_{A_j}^b} + \langle C_{A_j}^b \rangle$ is considered as

human contribution and removed from S_{bot} , where k is a threshold parameter. If there is no human contribution, this process is terminated. Otherwise go back to Step 2.

In Step 1, the procedure checks how likely the collected answers contain ones from bots by checking both the number of distinct answers and the ratio of the total answers to the number of distinct answers. The remaining steps are applied only if it finds out that it is likely that the collected answers contain ones from bots. All the answers in the resulting S_{bot} of the above procedure are considered as answers from bots and are therefore removed from the collected answers.

3.5 Evaluation of Human Answers

Recall that we have assumed that human users are benign but may sometimes be careless enough to provide erroneous answers. This module is applied to human answers, i.e., the collected answers if CAPTCHA is used with common sense problems or the surviving answers after the procedure described in Section 3.4 is applied to remove spurious answers from bots, to deduce a final answer, considered as the solution, to the problem. Like the procedure described in Section 3.4 to remove answers from bots, this procedure is applied only when the number of human answers to a problem is larger than a threshold.

Simple majority voting is used to combine individual human answers since according to Gentry et al. [24], majority voting is better than other methods such as Bayesian inference. Human answers to a problem are listed from high to low according to their frequencies. The slope, i.e., the relative difference of the neighboring frequencies is calculated. The slope at each answer is compared with the slope of the neighboring answer, starting with the answer of the highest frequency. If there is a substantial increase in slope at an answer, that answer is the separation point. All the answers with frequencies higher than the separation point are considered as the final answer, while the remaining answers are discarded.

4. EXAMPLE: IMAGE LABELING


In this section we use image labeling as a concrete common sense problem to demonstrate how HumanSense works. Assume that it is used with an email login page. Our image labeling task is

divided into three incremental refinement stages. The first stage is to ask users to describe the objects in an image. The candidate labels from the first stage are then refined with multiple choices at the second stage. Synonymic labels are also identified at this stage. In the third stage, users are asked to locate the object corresponding to a given label in a segmented image. We assume that at the start of the first stage, there is no prior knowledge of the images. We also assume that CAPTCHA is not needed for the email service that hosts the image labeling.

The first stage is to collect raw descriptions of objects in an image and turn them into candidate labels for the 2nd stage. At the start, all the images are put into this stage's pool of images. There is no prior knowledge of the objects in an image. Users are required to give some descriptions of objects in the images they see. Figure 4 shows an image labeling task in the 1st stage. As more data are collected, the scheme described in Section 3.4 is applied to remove spurious answers from bots, and the scheme described in Section 3.5 is applied to distill human answers to produce candidate labels. When candidate labels emerge, entering more of these candidate labels would not gain any further knowledge about the image. To prevent users from entering these candidate labels, the candidate labels are put into a "taboo phrase list". The "taboo phrase list" is inserted in the problem manifest file of the image to be displayed with the image. Users are forbidden from entering labels in the "taboo phrase list". The labels in the "taboo phrase list" are displayed as red words in Figure 4. The reload button on the right of the "OK" button allows a user to skip to the next image if the current one is too hard to describe. With more labels put into the "taboo phrase list", the value of the problem is also increased, and correct answers would be awarded more. The score of a correct label of the image as well as the total score of the current user are also shown in the problem frame in Figure 4. When the HumanSense server finds that there is an enough number of labels in the "taboo phrase list" or users would skip labeling an image which has labels in its "taboo phrase list", the HumanSense server concludes that it has collected enough answers for the image. The image is then removed from the pool of the first stage images and put into the pool of the second stage images.

User Name:

Password:



Taboo phrases: girl, T-shirt

Please describe objects in this image (separate them with "," and avoid taboo phrases):

Tip: You'll get 10 points for each proper description. Your current total is 0.

**Figure 4: First stage of the image labeling:
Collecting raw descriptions of objects in an image.**

The second stage targets at refining the candidate labels acquired in the first stage. The candidate labels are displayed as multiple choices with the image. Users are asked to choose the ones that are relevant to the images, as shown in Figure 5. The purpose is to further improve the quality of the labels to achieve high-quality labels for the images. It is possible that labels collected from the first stage contain synonyms. Users are also asked to correlate the synonyms in this stage.

Like in the first stage, correct answers would be scored or awarded. Bots and lazy human users would simply choose all the labels displayed, which guarantees that correct answers would be selected, resulting in no further knowledge gained about the image in this stage despite paying certain scores, awards, or money to these users. To deal with this problem, Random “trap labels” are intermixed with the candidate labels obtained from the first stage and listed with the image. These trap labels are fake labels that would not appear in the image. Selection of any trap label by a user would result in rejection of the answer.

Trap labels should be selected by a computer automatically and should not be semantically close to any candidate labels obtained from the first stage. HumanSense meets the desired requirements by utilizing the WordNet project [25][26], which offers a large lexical database for English with distance between two words to indicate how semantically close they are. To get a proper trap label, the HumanSense server picks a word randomly from the WordNet database, and then calculates its distances to all the displayed labels consisting of the candidate labels as well as selected trap labels for a specific instance of the image displayed for a user to answer. If all distances are greater than a preset threshold, this word is considered to be different enough from all the displayed labels, and is selected as a trap label. Otherwise a new word is picked and tested. This process is applied repetitively until an enough number of trap labels are selected. For each instance of a displayed image, a fresh set of trap labels are selected. They are randomly intermixed with the candidate labels for a user to choose from.



Figure 5: Second stage of the image labeling: Multiple choices of candidate and trap labels.

The last stage is to locate the object corresponding to a given label refined at the second stage in a segmented image. The segmentation algorithm we used in our experiments was the open source software

EDISON [27]. An image overlaid with its segmentation result is displayed for a user to use the mouse to click all the segments belonging to the object represented by the given label, as shown in Figure 6. A user can turn on or off the segmentation result to view the original image or the image overlaid with segmentation result. A left-click of the mouse selects a segment and a right-click deselects it. The segmentation boundary of selected segments would no longer show to allow a better view of the selected segments for the specific object, as shown in Figure 7.

5. EXPERIMENTAL RESULTS

We have implemented the proposed framework with ASP.NET on Microsoft’s Internet Information Services (IIS). The implementation was geared to support the image labeling described in Section 4 but many components were generic to support other common sense problems. The system has been tested internally within our lab. The labels on five images were collected from about 200 people. Although the experiment scale is still small, the results we obtained are very encouraging. We are working towards testing at a much larger scale, but it would take a substantial time and effort to achieve that goal.



Figure 6: Final stage of the image labeling: Locating an object given a segmented image and a label.



Figure 7: Boundaries among selected segments are removed to have a better view of the object corresponding to “building”.

In addition to human inputs, our experiments also used bots to generate random labels to input into the system. Figure 8 shows the collected labels from both humans and bots for the image shown in Figure 9. Since a bot would unlikely label an image correctly, there is no difference for a bot’s label to be a

meaningful word or not. In order to differentiate the labels from humans and those from bots, all the labels generated by bots were meaningless words in our experiments. All the labels we collected from humans were meaningful words. Therefore we can easily check if there is any label from bots that has survived our procedure described in Section 3.4 to remove spurious labels from bots. Figure 10 shows the surviving labels after applying our procedure to the collected answers shown in Figure 8 to remove spurious answers from bots, with the parameter k set to 1. As we can see from the figure, all the surviving labels are real words. This means that all the spurious labels from bots are removed successfully by our scheme.

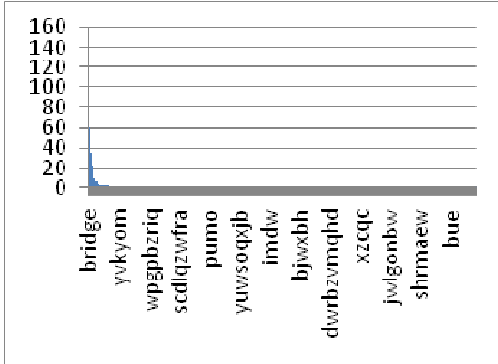


Figure 8: Collected labels from both humans and bots in the first stage for the image shown in Figure 9.



Figure 9: A test image for image labeling.

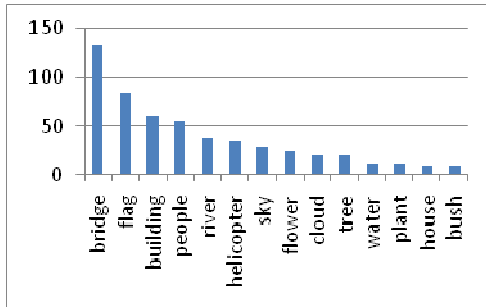


Figure 10: The labels in Figure 8 that survive our procedure to remove spurious labels from bots, with $k=1$ (see Section 3.4 for the meaning of k).

Our distilling procedure described in Section 3.5 is then applied to the surviving labels shown in Figure 10. These labels are considered as answers from humans. The result is shown in Figure 11. By comparing the labels in Figure 11 and the objects in Figure 9, we can conclude that all the major objects in the image have been labeled by our system. Therefore, our system can produce high-quality labels for images.

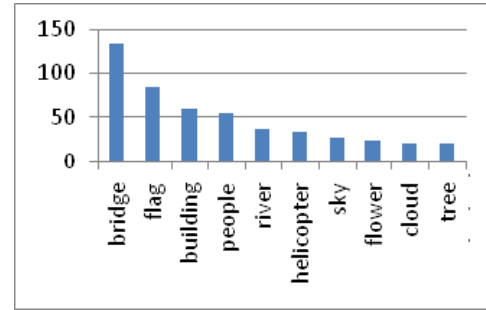


Figure 11: The resulting labels from the 1st stage for the image shown in Figure 9.

6. CONCLUSION

We have presented in this paper a human computation framework to mobilize Internet users to solve large-scale common sense problems efficiently and economically. An ecosystem based on the system is described in which all the parties would get benefits from the system. Internet users are motivated to offer correct answers to common sense problems in exchange for free services such as free Web emails, software or music downloading, or for monetary award. We described a system that can effectively prevent malicious players in the system to launch attacks against others or gain money or awards without contribution. We proposed a scheme to effectively remove answers from bots, and a majority voting scheme to distill human answers to achieve high-quality solutions to common sense problems. We then applied the general human computation framework to image labeling. Three incremental refinement stages are used to produce high-quality image labels. The first stage asks users to describe objects in an image. The candidate labels obtained from the first stage are then listed together with trap labels to ask users to refine in the 2nd stage. Synonymic labels are also correlated by users in this stage. The last stage is to ask users to locate an object corresponding to a given label in a segmented image. A user can simply use the mouse to click segments to locate an object. Experimental results were also reported. They have shown that our scheme removes spurious answers from bots effectively. That scheme together with the majority voting scheme to distill human answers can produce high-quality image labels.

7. REFERENCES

- [1] D. B. Lenat, R. V. Guha, K. Pittman, D. Pratt, and M. Shepherd. CYC: A Large-Scale Investment in Knowledge Infrastructure. *Communications of the ACM* 38(11): 33–38, Nov. 1995.
- [2] P. A. Boxer. Towards Learning Naive Physics by Visual Observation: Qualitative Spatial Representations. *Proceedings of the 14th Australian Joint Conference on Artificial intelligence: Advances in Artificial intelligence*, December 10-14, 2001. M. Stumptner, D. Corbett, and M.

- J. Brooks, Eds. Lecture Notes In Computer Science, vol. 2256. Springer-Verlag, London, 62-70.
- [3] The Open Mind project. <http://www.openmind.org/>.
- [4] Luis von Ahn and Laura Dabbish. Labeling Images with a Computer Game. ACM CHI 2004, April 2004.
- [5] Luis von Ahn, Ruoran Liu, and Manuel Blum. Peekaboom: A Game for Locating Objects in Images. ACM CHI 2006, April 2006.
- [6] Luis von Ahn, Mihir Kedia, and Manuel Blum. Verbosity: A Game for Collecting Common-Sense Facts. ACM CHI 2006, April 2006.
- [7] Luis von Ahn, Shiry Ginosar, Mihir Kedia, Ruoran Liu, and Manuel Blum. Improving Accessibility of the Web with a Computer Game. ACM CHI 2006, April 2006.
- [8] Luis von Ahn. Games with a Purpose. IEEE Computer Magazine, June 2006.
- [9] Luis von Ahn. Human Computation (PhD Thesis). CMU-CS-05-193, December 2005.
- [10] Google Image Labeler (Beta, published in August, 2006). <http://images.google.com/imagelabeler/>.
- [11] Luis von Ahn. CAPTCHA: Using Hard AI Problems For Security. Eurocrypt 2003.
- [12] reCAPTCHA. <http://recaptcha.net/>.
- [13] Internet Archive. <http://www.archive.org/index.php>.
- [14] Wikipedia item on Human-Based Computation. http://en.wikipedia.org/wiki/Human-based_computation.
- [15] Google AdSense. <https://www.google.com/adsense/>.
- [16] A. Kosorukoff. Human Based Genetic Algorithm. IEEE Int.Conf. on Systems, Man, and Cybernetics, vol. 5, pp. 3464-3469, 2001.
- [17] K. Sims. Artificial Evolution for Computer Graphics. Computer Graphics, 25(4) (SIGGRAPH'91).
- [18] X.-S. Hua and Q.-J. Qi. Online Multi-Label Active Annotation: Towards Large-Scale Content-Based Video Search. ACM Multimedia 2008.
- [19] J. Ruderman. The Same Origin Policy. <http://www.mozilla.org/projects/security/components/same-origin.html>.
- [20] Web Hypertext Application Technology Working Group. HTML 5 - Cross-document messaging. <http://www.whatwg.org/specs/web-apps/current-work/#crossDocumentMessages>.
- [21] H. J. Wang, X. Fan, C. Jackson, and J. Howell. Protection and Communication Abstractions for Web Browsers in MashupOS. 21st ACM Symposium on Operating Systems Principles (SOSP), Stevenson, WA, October 2007.
- [22] R. Guo, B. B. Zhu, M. Feng, A. Pan, and B. Zhou. CompoWeb: A Component-Oriented Web Architecture. WWW 2008.
- [23] C. Jackson and H. J. Wang. Subspace: Secure Cross-Domain Communication for Web Mashups. WWW 2007, pp. 611-619, Canada, May 2007.
- [24] C. Gentry, Z. Ramzan, and S. Stubblebine. Secure Distributed Human Computation. In Proceedings of 6th ACM Conference on Electronic Commerce, pages 155-164, New York, June 2005.
- [25] George A. Miller. WordNet: A Lexical Database for English. Communications of the ACM, Nov. 1995.
- [26] WordNet::Similarity, a Perl module for computing measures of semantic relatedness based on WordNet. <http://www.d.umn.edu/~tpederse/similarity.html>.
- [27] EDISON: the Edge Detection and Image Segmentation system. <http://www.caip.rutgers.edu/riul/research/code/EDISON/index.html>.